김유진, 송경민, 문예진, 한유경, 이상민

ELMo (Embedding from Language Models)

1. Introduction

2. ELMo

3. Evaluation

4. Analysis

5. Conclusion

1. Introduction

논문 : Deep contextualized word representations

Word2Vec이나 Skip-gram 등 이전 모델

king – man + woman = queen
(1,4) - (1,0) + (4,1) = (4,5)

단점

- 각 단어가 단 한 개의 벡터로만 표현됨
  →문법구조, 다의어에 따른 뜻 변형을 적절히 반영하기에 어려움

1. Introduction

ELMo(Embeddings from Language Models)

: 사전 훈련 + 문맥 = 문맥 반영 언어 모델(deep contextualized word representations)

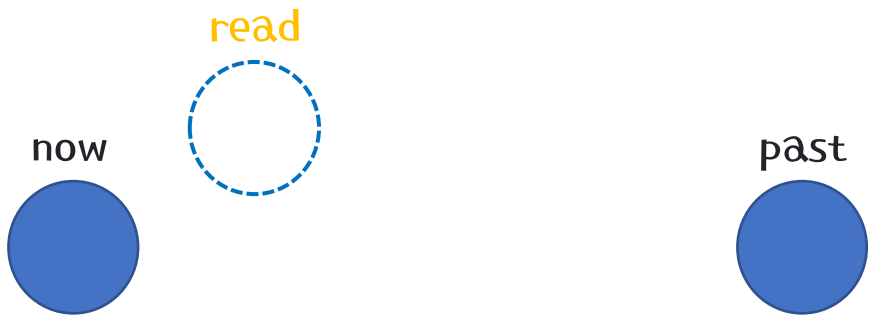
ELMo word representations are functions of the entire input sentence.

LSTM layer의 최종 layer만을 취한 것이 아닌, 모든 layer 결과를 가중합하여 얻음.

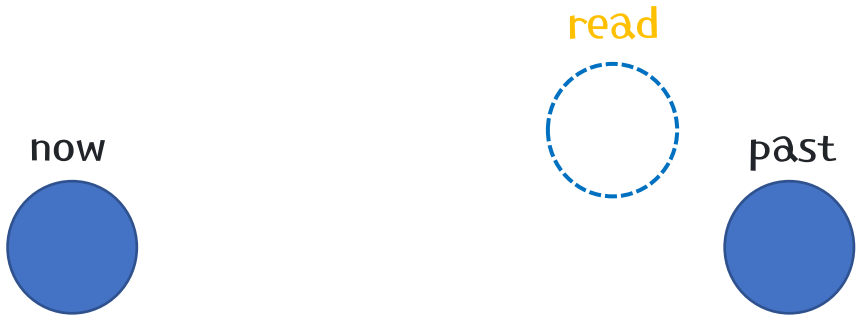
LSTM의 낮은 단계의 layer : 품사 등 문법 정보

높은 단계의 layer : 문맥 정보

I read a book.



I read a book yesterday.



2. ELMo

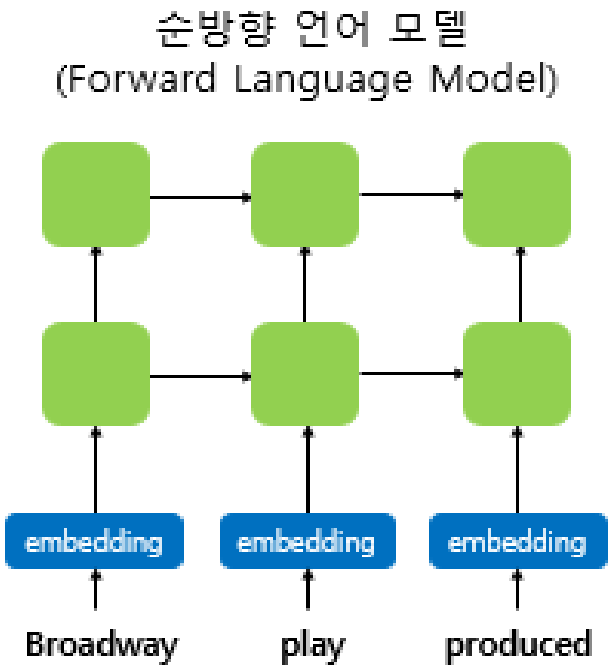
- 전체 문장을 input으로 받고, 그에 대한 각 단어들의 representation 생산
- Character convolution로부터 얻은 biLM의 가장 위 2개 layer의 가중합으로 계산
- 큰 사이즈로 biLM pretrain 시켰을 때 semi-supervised learning 가능
- 쉽게 다른 모델에 붙일 수 있음

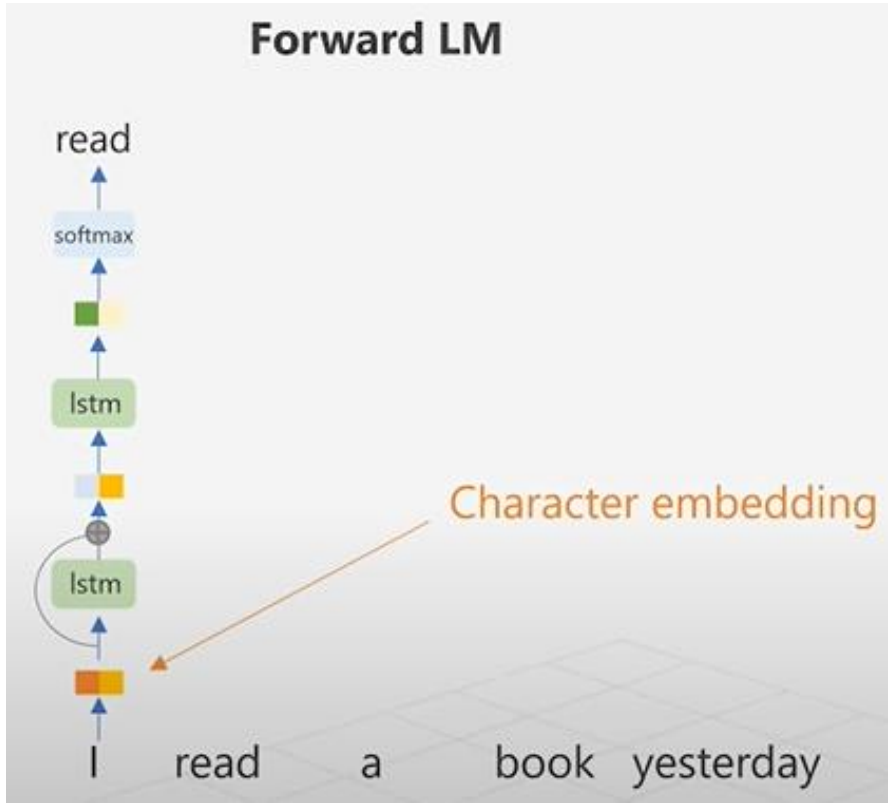
2. ELMo – biLM

biLM : 순방향 LM + 역방향 LM

N개의 token $(t_1, t_2, \ldots, t_N)$

- Forward Language Model : $(t_1, t_2, \ldots, t_{k-1})$이 주어졌을 때 token t_k가 나올 확률을 계산





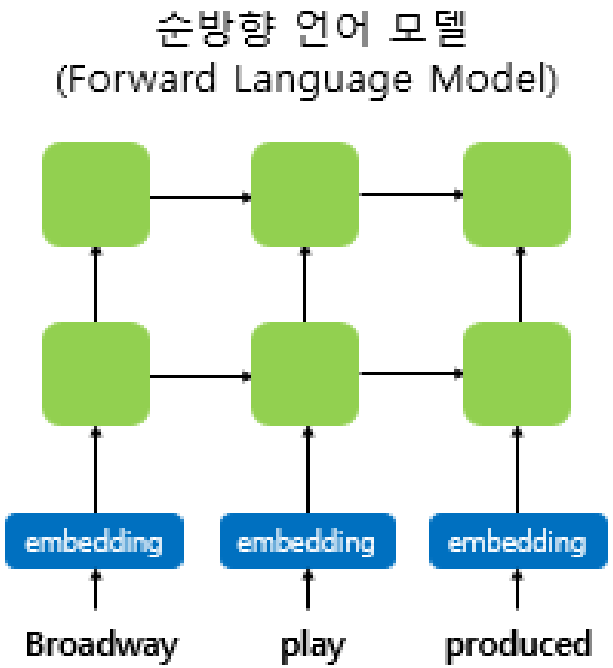
$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$$

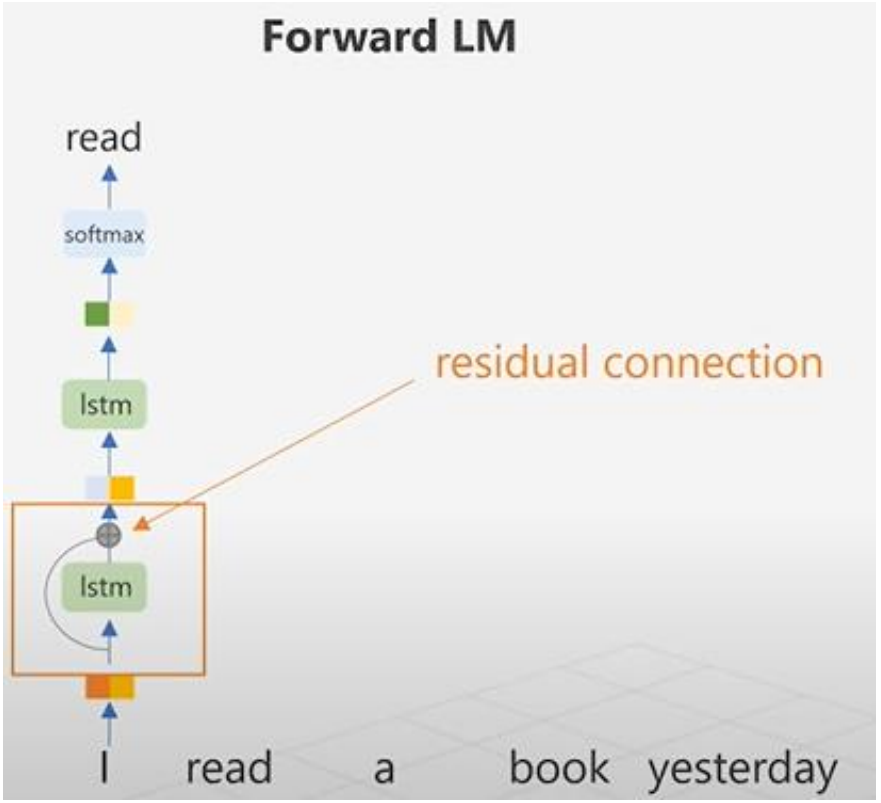
2. ELMo – biLM

biLM : 순방향 LM + 역방향 LM

N개의 token $(t_1, t_2, \ldots, t_N)$

- Forward Language Model : $(t_1, t_2, \ldots, t_{k-1})$이 주어졌을 때 token t_k가 나올 확률을 계산





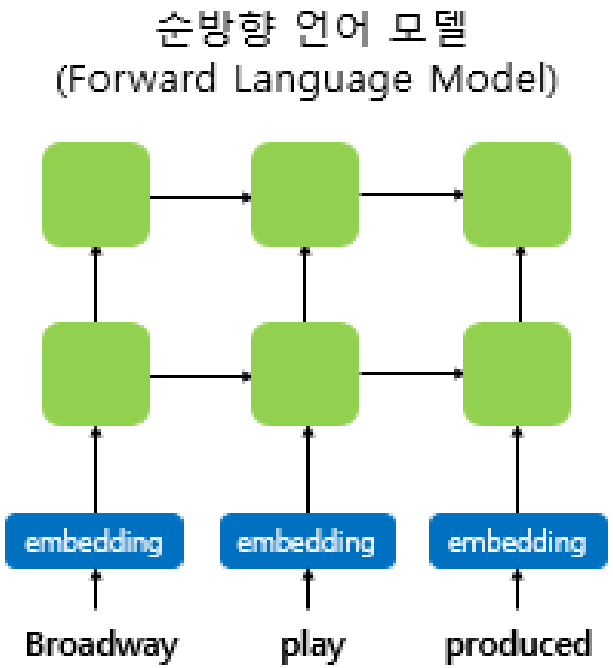
$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$$

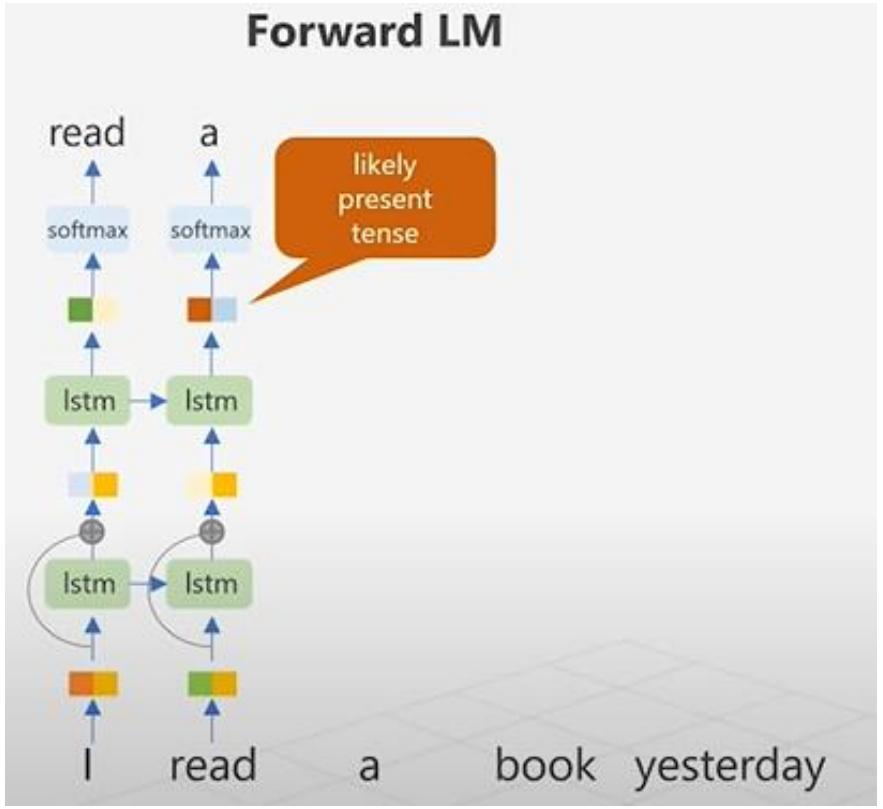
2. ELMo – biLM

biLM : 순방향 LM + 역방향 LM

N개의 token $(t_1, t_2, \ldots, t_N)$

- Forward Language Model : $(t_1, t_2, \ldots, t_{k-1})$이 주어졌을 때 token t_k가 나올 확률을 계산





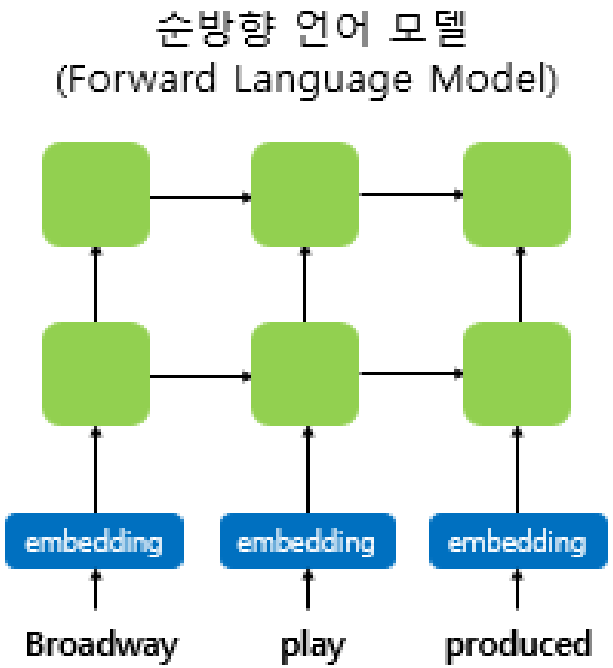
$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$$

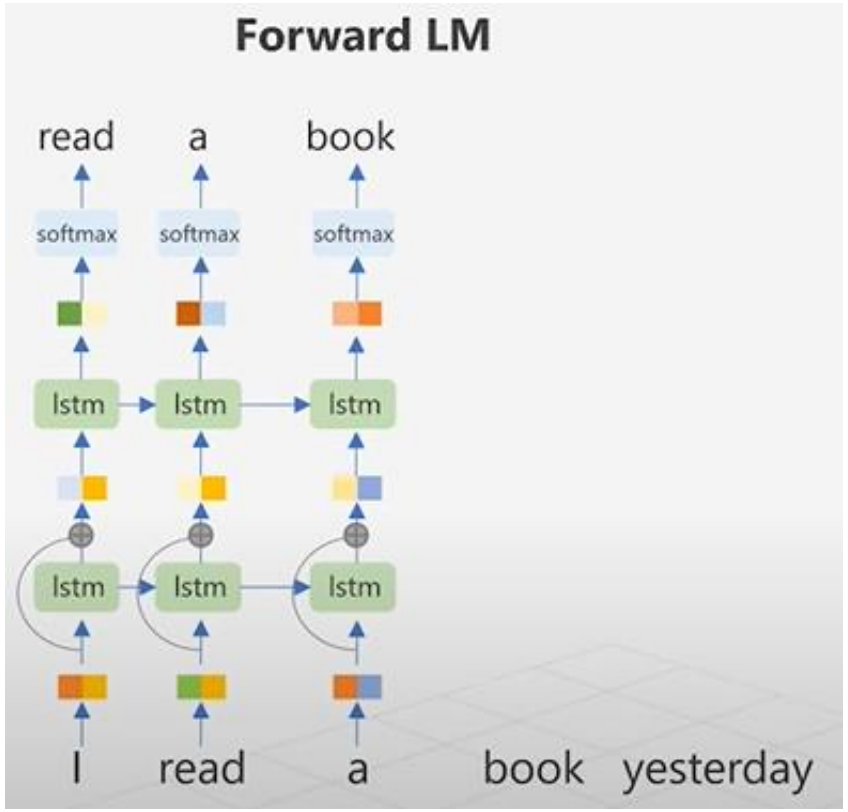
2. ELMo – biLM

biLM : 순방향 LM + 역방향 LM

N개의 token $(t_1, t_2, \ldots, t_N)$

- Forward Language Model : $(t_1, t_2, \ldots, t_{k-1})$이 주어졌을 때 token t_k가 나올 확률을 계산





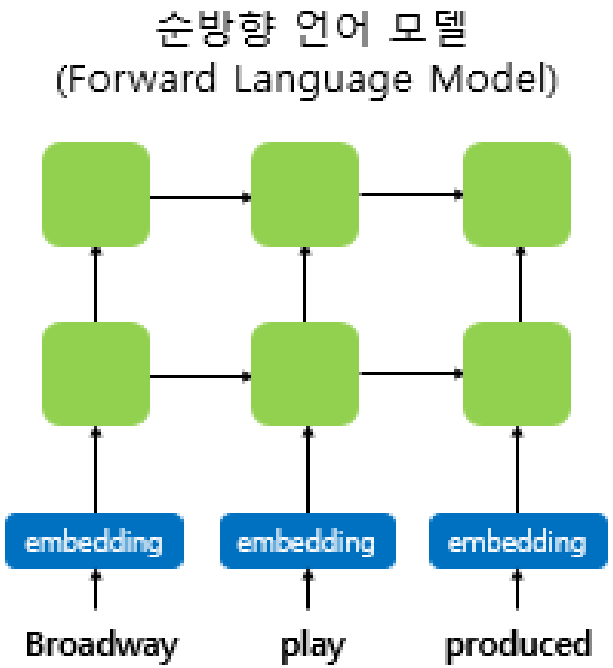
$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$$

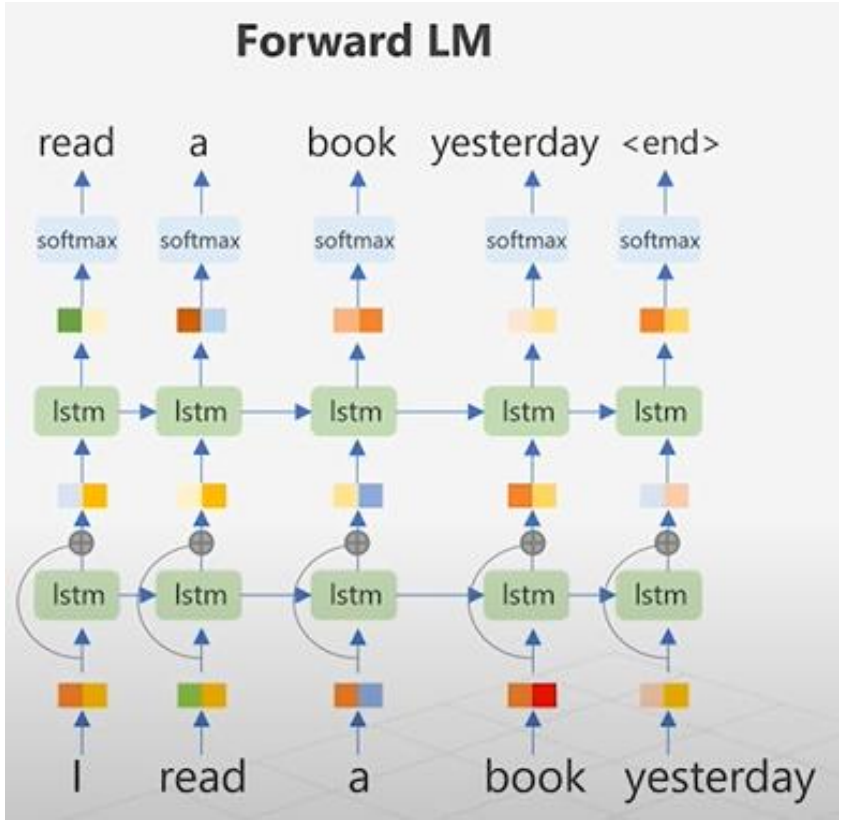
2. ELMo - biLM

biLM : 순방향 LM + 역방향 LM

N개의 token $(t_1, t_2, \ldots, t_N)$

- Forward Language Model : $(t_1, t_2, \ldots, t_{k-1})$이 주어졌을 때 token t_k가 나올 확률을 계산

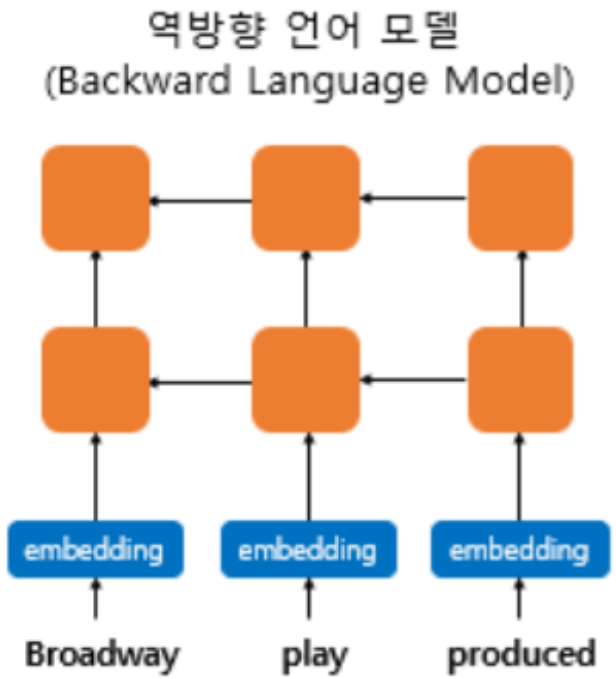




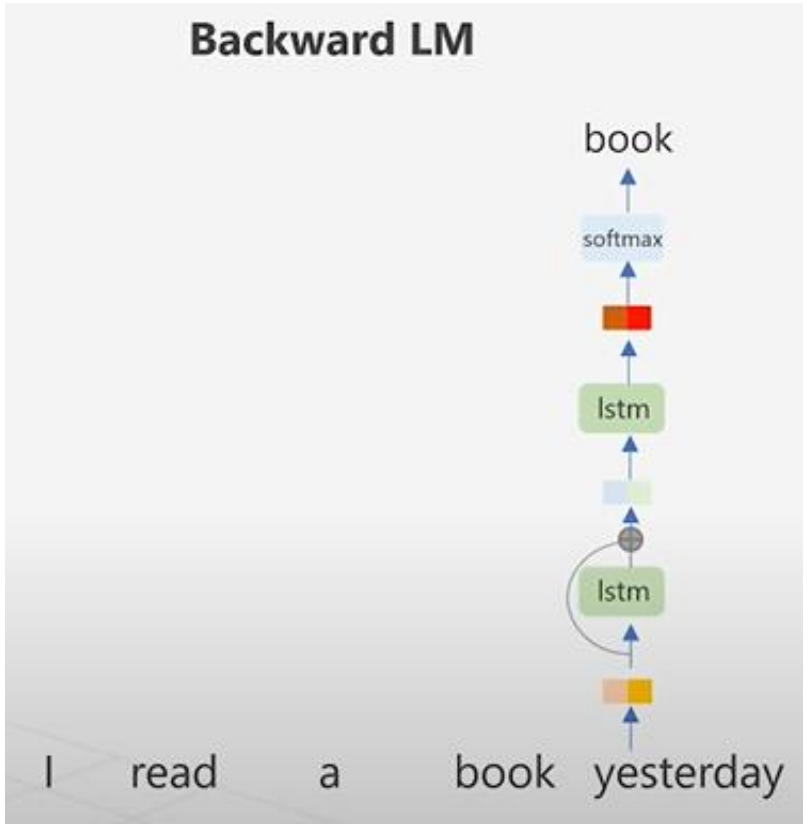
$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$$

2. ELMo - biLM

- Backward Language Model: $(t_{k+1}, t_{k+2}, \ldots, t_N)$이 주어졌을 때 token t_k가 나올 확률을 계산

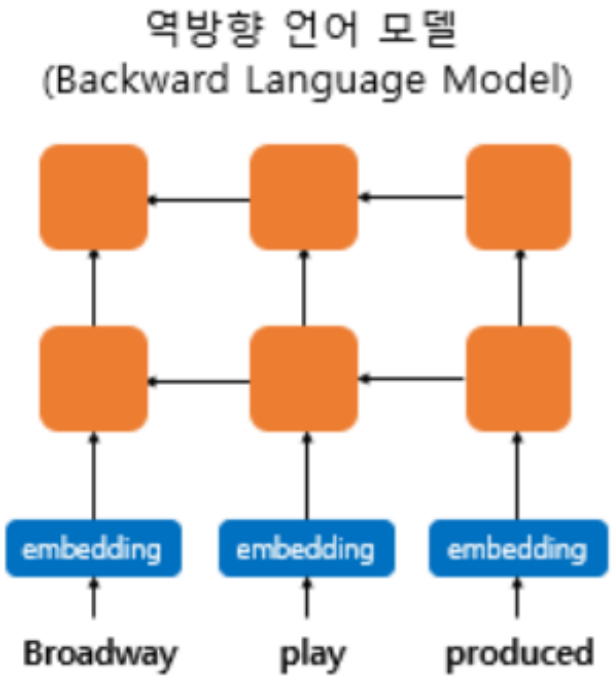


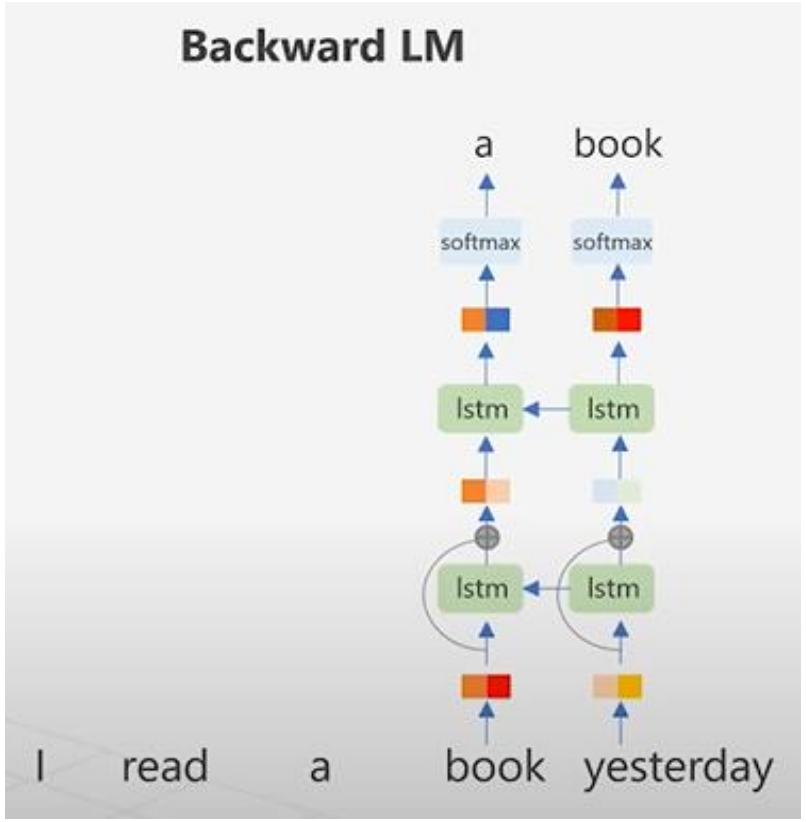
$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N)$$



2. ELMo - biLM

- Backward Language Model: $(t_{k+1}, t_{k+2}, \ldots, t_N)$이 주어졌을 때 token t_k가 나올 확률을 계산

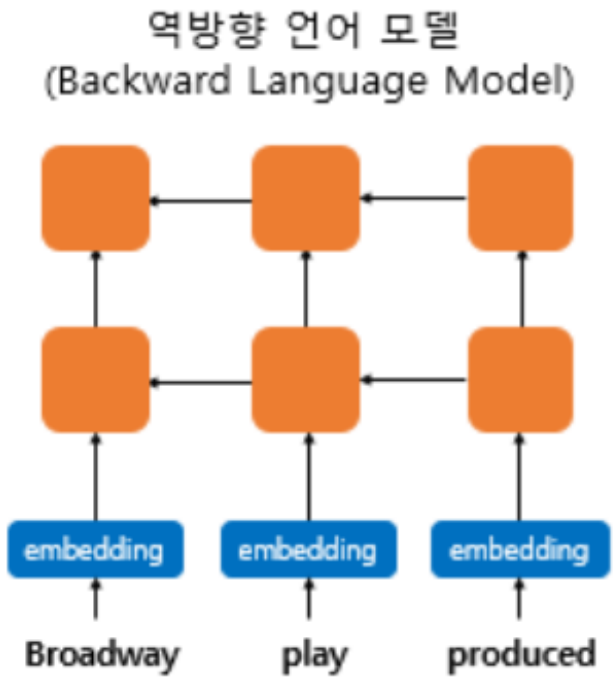


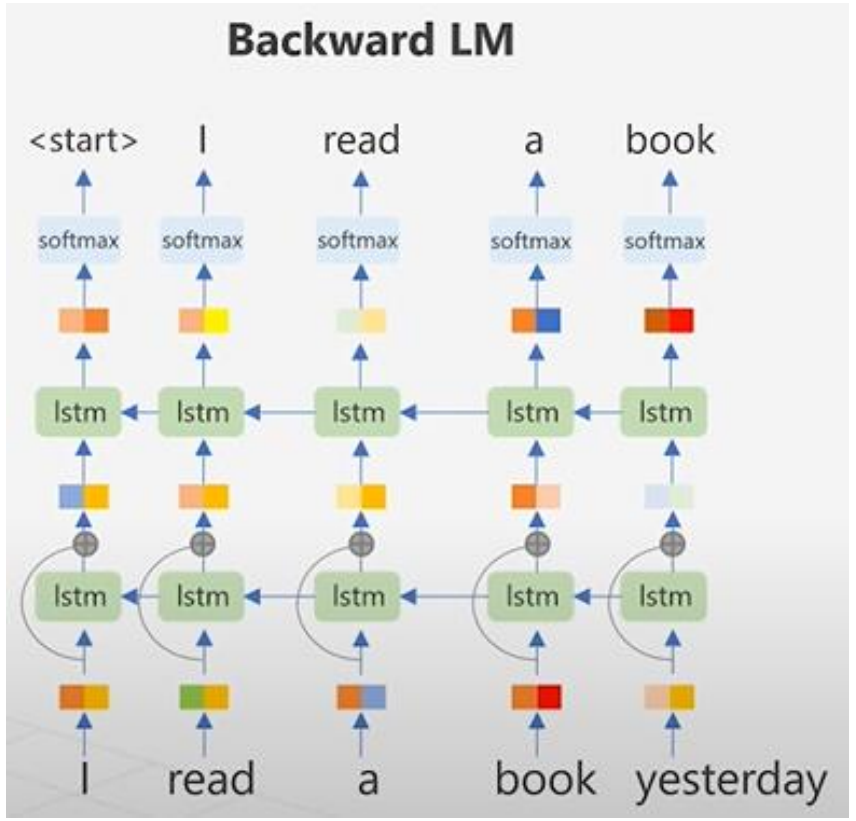


$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N)$$

2. ELMo - biLM

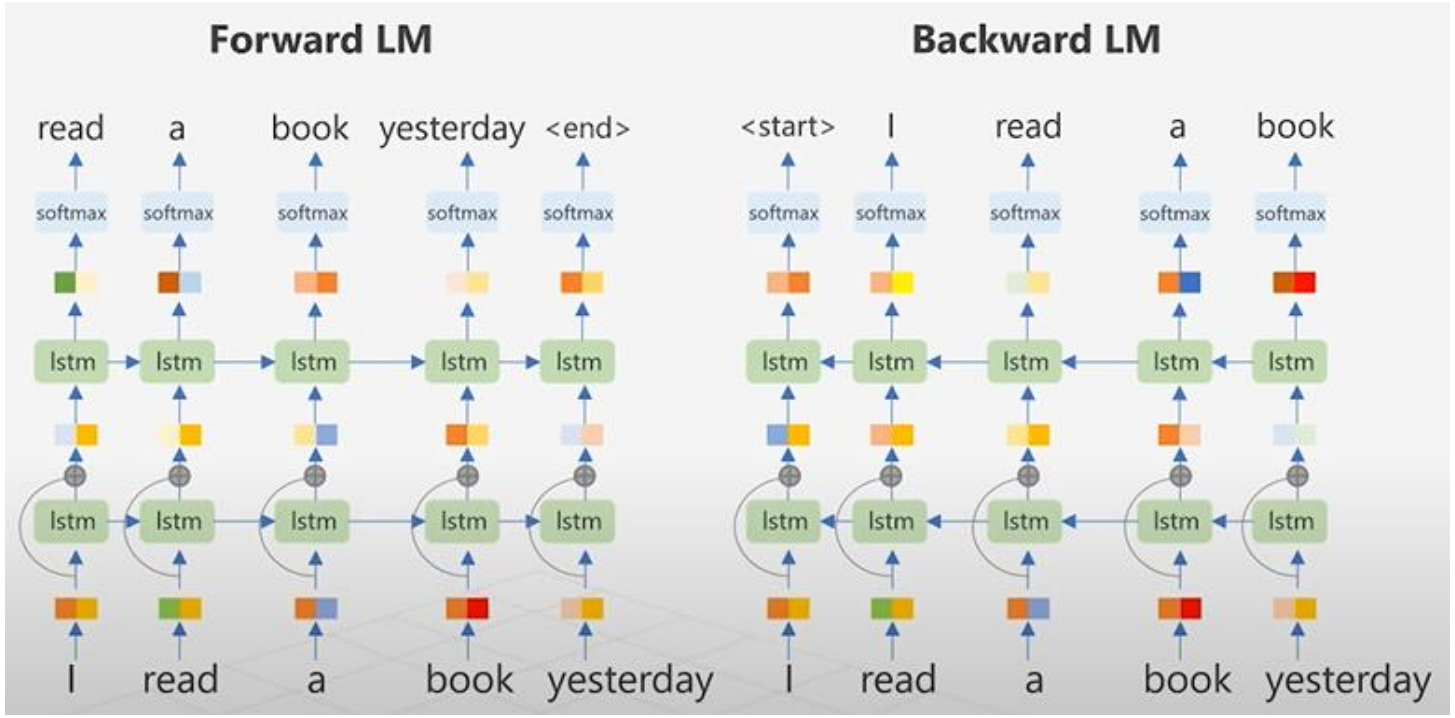
- Backward Language Model: $(t_{k+1}, t_{k+2}, \ldots, t_N)$이 주어졌을 때 token t_k가 나올 확률을 계산





$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N)$$

2. ELMo - biLM



biLM : 순방향 LM + 역방향 LM

→ 이 둘을 결합시킨 log likelihood를 최대화!

$$\sum_{k=1}^{N} \left(\log\, p(t_k|t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log\, p(t_k|t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right)$$

Θx는 token representation, Θs는 Softmax layer

2. ELMo - ELMo representation

ELMo는 biLM의 중간 layer representation을 task-specific하게 결합함.
biLM의 L개의 layer는 각 token t_k당 2L+1개의 representation을 계산

$$R_k = \{x_k^{LM}, \overrightarrow{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM} | j = 1, \ldots, L\} = \{h_{k,j}^{LM} | j = 0, \ldots, L\}$$

$h_{k,0}^{LM}$: token layer $\qquad h_{k,j}^{LM} = [\overrightarrow{h}_{k,j}^{LM}; \overleftarrow{h}_{k,j}^{LM}]$

ELMo는 R의 모든 layer를 하나의 벡터 $\text{ELMo}_k = E(R_k; \Theta_e)$ 로 압축
일반적으로, 모든 biLM layer의 task-specific한 weighting을 계산

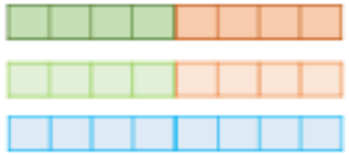
$$\text{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM}$$

s^{task} : softmax-정규화된 가중치
γ^{task}: 전체 ELMo 벡터의 크기를 조절하는 역할(scalar parameter)

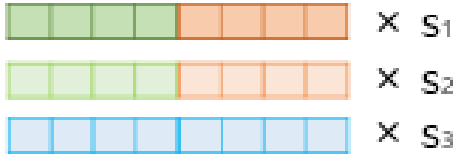
2. ELMo - ELMo representation

* read의 임베딩 구하기

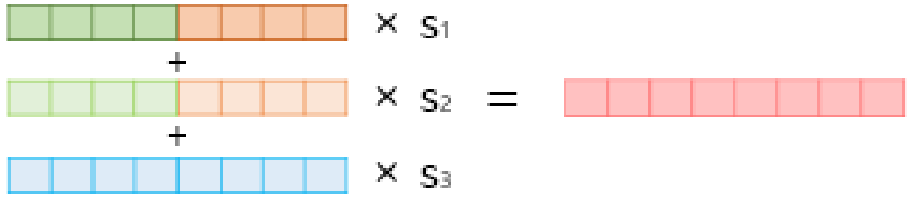
1) 각 층의 출력값을 연결(concatenate)한다.



2) 각 층의 출력값 별로 가중치를 준다.



3) 각 층의 출력값을 모두 더한다.



4) 벡터의 크기를 결정하는 스칼라 매개변수(γ)를 곱한다.

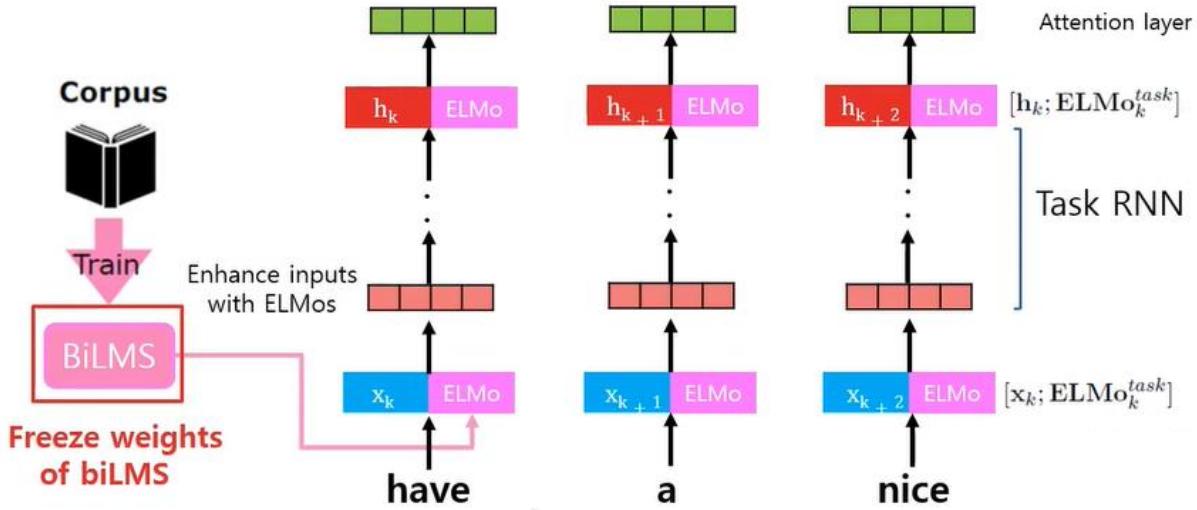
2. ELMo – 활용

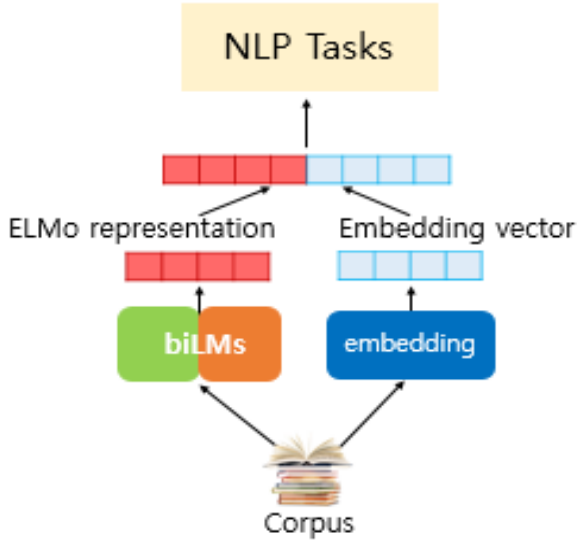
task 모델을 향상시키도록 biLM을 쓰는 과정

1. biLM을 돌리고 각 단어마다 모든 layer representation을 기록
2. 모델이 이 representation들의 선형결합을 배우도록 함
   2-1) biLM이 없는 지도 모델을 고려
   2-2) $(t_1, t_2, \ldots, t_N)$이 주어지면 pretrain된 단어 임베딩을 사용하여 각 token마다 x_k 를 만듦
   2-3) 모델이 biRNN 등을 사용해 문맥-의존적 representation h_k를 생성

ELMo를 지도 모델에 추가하려면

1. biLM의 weight 고정
2. ELMo 벡터 $ELMo_k^{task}$와 x_k 연결
3. $[x_k; ELMo_k^{task}]$를 task RNN에 전달





3. Evaluation

〈주요 task에 대한 성능 비교〉

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

SQuAD :Q&A , SNLI : Textual entailment , SRL : Semantic role labeling , Coref : Coreference resolution , NER : Named entity extraction , SST-5 : Sentiment analysis

다양한 downstream task들에서 SOTA보다 좋은 성능을 보임!

4.1. Alternate layer weighting schemes

〈λ 에 대한 성능 비교〉

Task	Baseline	Last Only	All layers	
			λ=1	λ=0.001
SQuAD	80.8	84.7	85.0	**85.2**
SNLI	88.1	89.1	89.3	**89.5**
SRL	81.6	84.1	84.6	**84.8**

- 마지막 layer만 쓰는 것보다 모든 layer를 쓰는 것이 더 좋음
- λ를 작게 하는 것이 더 좋은 성능을 나타내며 task의 종류에는 크게 영향받지 않는 것을 보임

4.2. Where to include ELMo?

〈ELMo 위치에 대한 성능 비교〉

Task	Input Only	Input & Output	Output Only
SQuAD	85.1	**85.6**	84.8
SNLI	88.9	**89.5**	88.7
SRL	**84.7**	84.3	80.9

- SQuAD, SNLI에서는 biRNN의 Output에도 ElMo를 추가하는 것이 더 좋은 성능을 보임

→ Task에 따라 적절한 architecture가 있다고 생각할 수 있음.

4.3. What information is captured by the biLM's representations?

〈Glove와 ELMo에서의 "play"〉

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

- Glove에서는 "play"에 관련된 단어들로 스포츠와 유사한 것들이 나옴
- biLM에서는 "play"와 유사한 의미로 사용되는 문장이 유사한 것으로 나옴

4.3. What information is captured by the biLM's representations?

〈Word-Sense Disambiguation – 단어 중의성 해소〉

Model	F_1
WordNet 1st Sense Baseline	65.9
Raganato et al. (2017a)	69.9
Iacobacci et al. (2016)	**70.1**
CoVe, First Layer	59.4
CoVe, Second Layer	*64.7*
biLM, First layer	67.4
biLM, Second layer	*69.0*

- BiLM이 CoVe보다 성능이 우수함
- BiLM의 first layer를 이용하는 것보다 Second layer를 이용하는 것이 성능이 우수
  → 높은 layer일수록 문맥 정보 학습

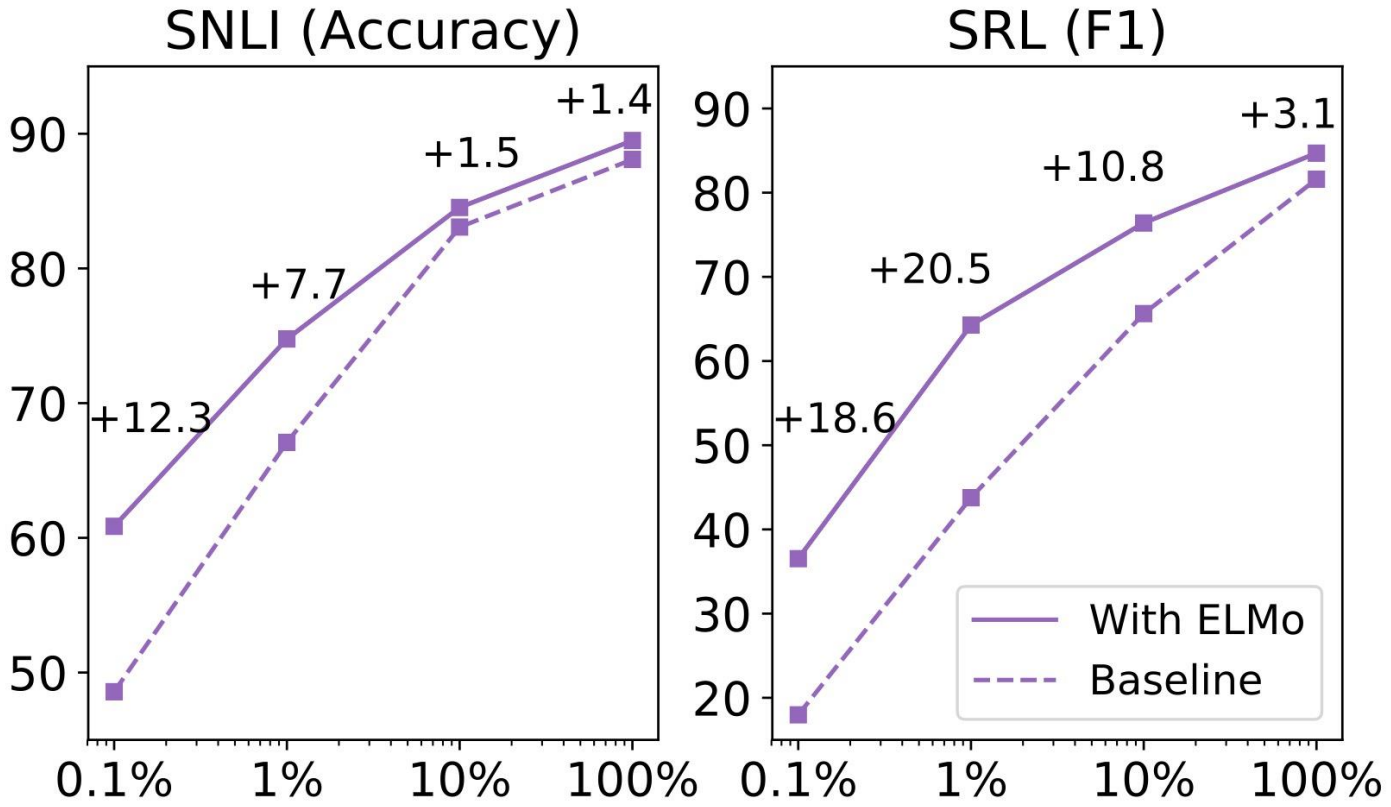
〈POS tagging-형태소 분석〉

Model	Acc.
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	**97.8**
CoVe, First Layer	*93.3*
CoVe, Second Layer	92.8
biLM, First Layer	*97.3*
biLM, Second Layer	96.8

- BiLM이 CoVe보다 성능이 우수함
- BiLM의 first layer가 second layer보다 성능이 우수
  → 낮은 layer일수록 문법 정보 학습

4.4 Sample efficiency

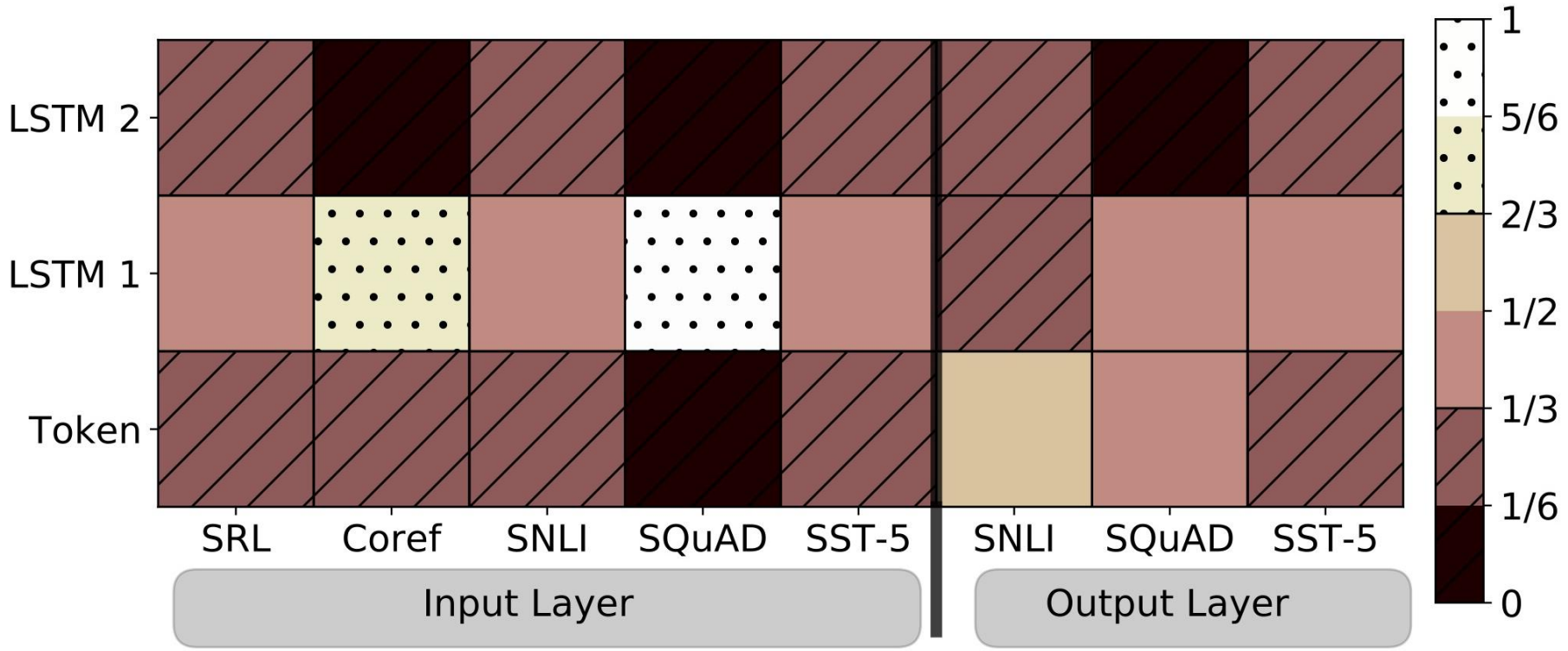
〈training set 크기에 따른 성능 비교〉



- 모델에 ELMo를 추가했을 때가 그렇지 않을 때보다 학습속도가 빠름
- 더 작은 훈련 세트를 더 효율적으로 훈련함

4.5 Visualization of learned weights

〈학습된 weights 시각화〉



- 소프트맥스 정규화된 학습 계층 가중치를 시각화한 것
- Input Layer에서는 첫번째 biLSTM layer를 선호
- Output Layer에서는 weight가 균형있게 분배되었지만, 낮은 layer를 좀 더 선호

5. Conclusion

- biLM으로부터 고품질의 깊은 문맥의존 representation을 학습하는 일반적인 접근법을 도입함.
- 광범위한 NLP작업들에서 ELMo를 적용했을 때 큰 향상이 있는 것을 볼 수 있음.
- biLM 계층이 문맥 내 단어들에 대한 다른 유형의 구문 및 의미 정보를 효율적으로 인코딩하고, 모든 계층 사용 시 전반적인 작업 향상이 있음.

Thank you for listening