Deep contextualized word representations

Sunyoung Song

1. Introduction

- 기존의 Word2Vec은 각 단어가 한 개의 벡터로만 표현됨
  -> 문법구조나 다의어에 따른 뜻 변형을 반영하기가 어려움

- **문맥에 따라 다르게** word embedding 하는 방법을 생각하게 됨
  - 단어를 임베딩하기 전에 **전체 문장을 고려해서 embedding**

  -> **문맥을 반영한 워드 임베딩** (Contextualized Word Embedding)

1. Introduction

- **ELMo** : Embeddings from Language Model
  - 사전 훈련된 언어 모델을 사용
  - 문맥을 고려하는 문맥 반영 언어 모델
  - LSTM의 마지막 layer만 사용하는 기존과는 **달리 LSTM의 모든 내부 layer를 사용**
    -> 더 많은 정보 사용 가능

    - **higher-level** LSTM : **semantic 한 정보**를 잘 표현 (output layer 에 가까운 layer)
    - **lower-level** LSTM : **syntax 한 정보를** 잘 표현 (input layer 에 가까운 layer)

  - 전체 문장을 input으로 받고, 각 단어들의 representation을 뽑음
  - 큰 사이즈로 biLM pre-train 시켰을 때, semi-supervised learning 가능
  - 쉽게 다른 모델에 붙일 수 있음

2. Bidirectional Language Model

- **bidirectional** Language model
  - **forward** language model + **backward** language model

  - N 개의 token (t_1, t_2, … ,t_N) 이 있다고 할 때
    - **forward language model :** (t_1, t_2, … , t_k-1) 로 t_k 가 나올 확률을 계산

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

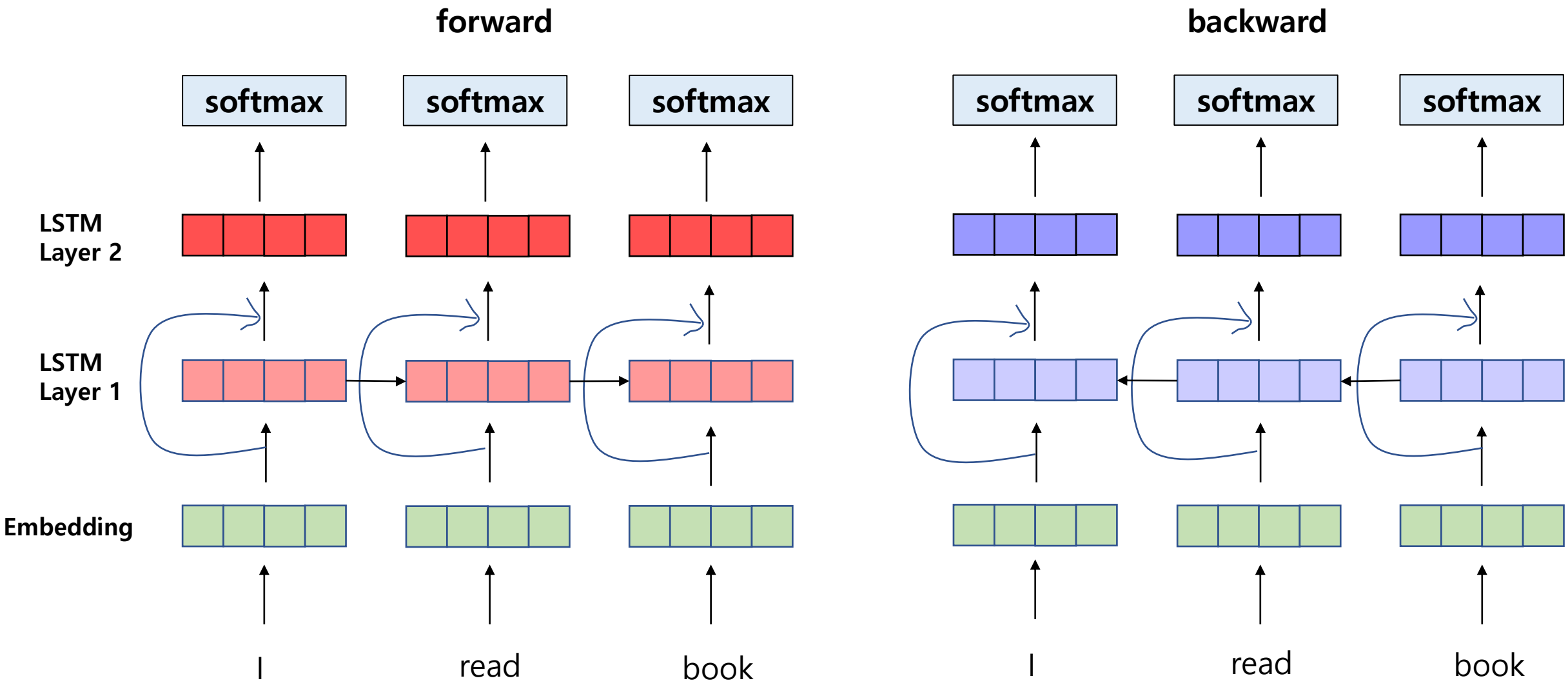
    - **backward language model :** (t_k+1, t_k+2, … , t_N) 로 t_k 가 나올 확률을 계산

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

    - 최종적으로 forward + backward, 두 방향의 log likelihood를 최대화 시키는 방향으로 학습

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

2. Bidirectional Language Model



3. ELMo

- biLM의 중간 층들의 representation 들의 결합으로 나타낼 수 있음
  - 입력 t_k에 대해 L 개의 층이 있을 때, L+L+1 (context independent representation) 으로 **2L+1 개의 representation**을 사용하게 됨

$$
\begin{aligned}
R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\
&= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},
\end{aligned}
$$

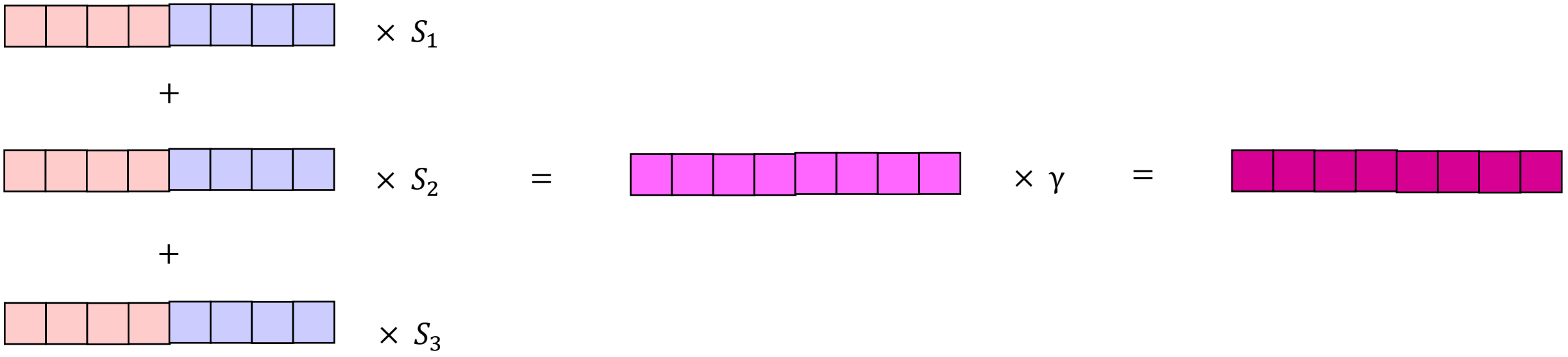
  - 최종적으로 모든 층에서 생성한 representation을 결합해 하나의 벡터를 생성

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

3. ELMo

1. 각 층의 출력값 연결
2. 각 층의 출력값 별로 가중치를 줌
3. 각 층의 출력값을 모두 더함
4. scalar parameter를 곱해 벡터의 크기 결정

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



4. Using biLMs for supervised NLP tasks

- ELMo representation 을 이용해서 기존의 NLP task의 성능을 끌어올릴 수 있음
  - x_k : 기존의 문맥을 고려하지 않는 단어 임베딩
  - h_k : x_k 임베딩에 ELMo representation을 결합/ task model의 input으로 사용됨

- **ELMo representation을 기존의 task에 함께 사용하는 방법**
  1. biLM의 가중치를 고정(freeze)
  2. 문맥을 고려하지 않은 단어 임베딩인 x_k를 ELMo representation과 연결

     $$[\mathbf{x}_k; \mathbf{ELMo}_k^{task}]$$

  3. 새로 생성한 문맥을 고려한 단어 임베딩을 task 모델의 input으로 제공

- ELMo에 dropout 적용, loss에 $\lambda \|w\|_2^2$를 더해 ELMo 가중치를 정규화하는 방법이 성능 향상에 도움이 되었음

5. Pre-trained bidirectional language model architecture

- 본 논문에서 다루는 최종 모델
  - 2개의 LSTM 층 사용 (L=2)
  - 각 층의 output은 4096 개의 unit 과 512 차원
  - 첫번째 층과 두번째 층을 residual connection으로 연결
  - 문맥을 고려하지 않는 임베딩은 2048 character n-gram convolution filter 사용, 2개의 highway layer를 추가적으로 사용, 최종적으로 512차원의 output

  -> 입력에 대해 총 3개의 representation 생성

  - 10 epoch 동안 1B Word Benchmark를 기반으로 학습
    - Backward perplexity : 39.7
    - Forward perplexity와 backward perplexity가 거의 유사하게 측정되었지만 backward perplexity가 조금 더 낮게 측정

6. Evaluation

- 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%

- ELMo representation을 추가함으로써 6개의 NLP task에서 성능 향상을 이뤄냄
- 6개의 task : **Question answering**(질의응답), **Textual entailment**(가설이 참인지 판단), **Semantic role labeling**(의미역 결정), **Coreference resolution**(상호 참조), **Named entity extraction**(개체명 인식), **Sentiment analysis**(감성 분석)

6. Evaluation

- λ에 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
  - λ 를 작게 하는 것이 더 성능이 높음

6. Evaluation

- 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, SNLI 에서는 biRNN의 output에도 ELMo를 추가하는 것이 더 성능이 높음

7. Analysis

- Glove와 biLM에서의 “play”
  - Glove에서는 “play”에 관련된 단어들로 스포츠와 유사한 것들이 나옴
  - biLM에서는 “play”와 유사한 의미로 사용되는 문장이 유사한 것으로 나옴

	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 .

7. Analysis

- Word-Sense Disambiguation (단어 모호성 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 CoVe 보다 성능이 우수함
- biLM의 first layer를 이용하는 것보다 second layer를 이용하는 것이 성능이 높음
  -> **높은 layer일수록 문맥 정보 학습**

7. Analysis

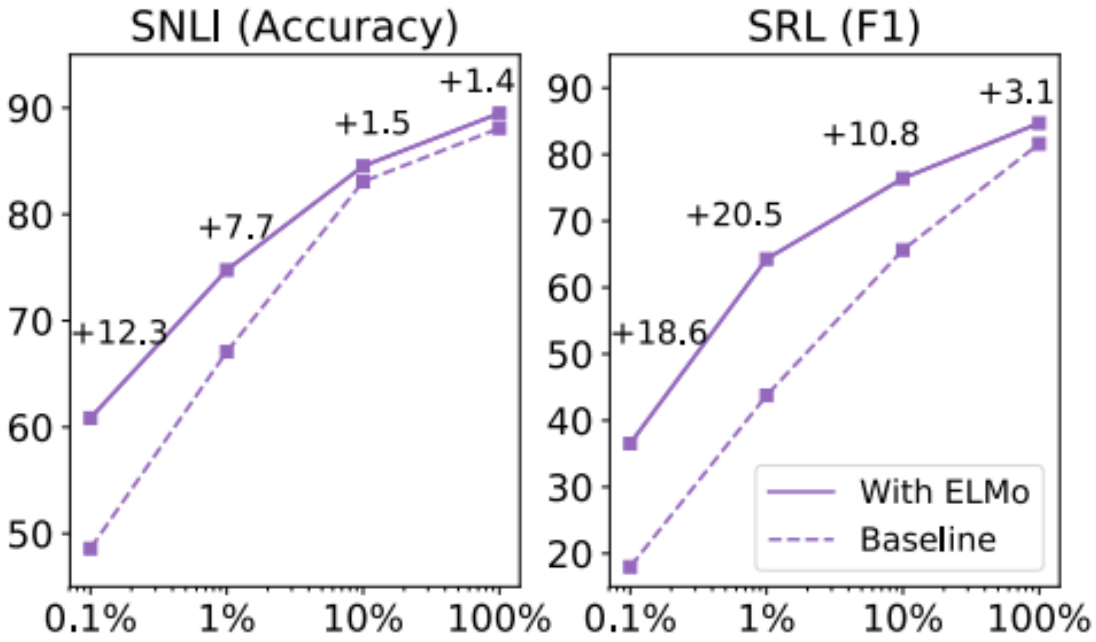
- POS tagging (형태소 분석)

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**

7. Analysis

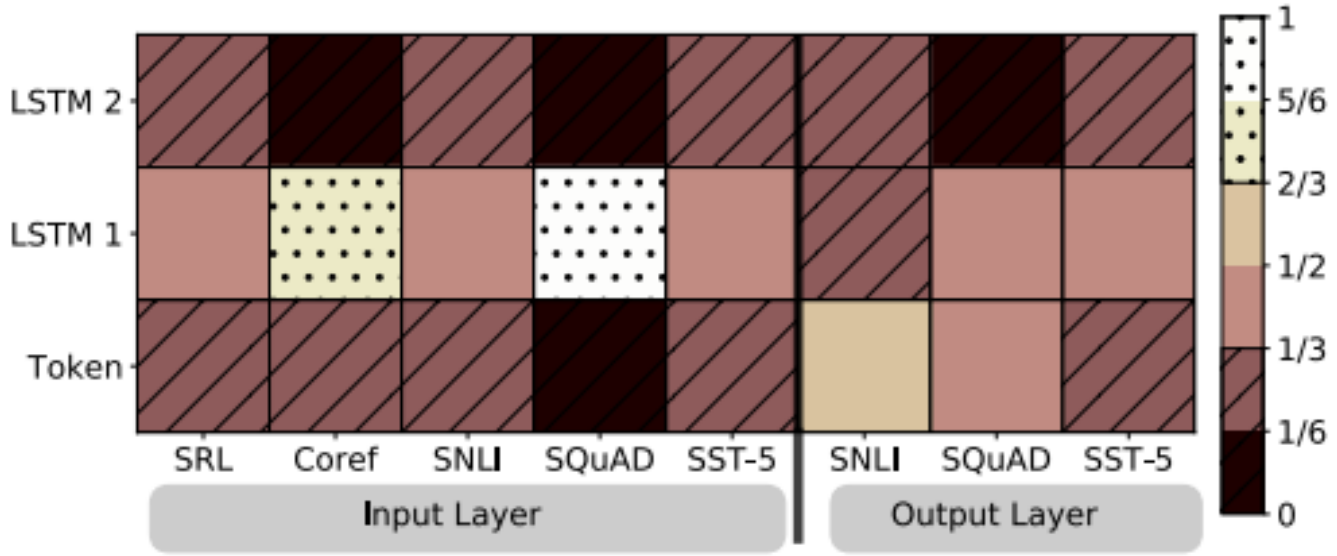
- Training set 크기에 따른 성능 비교



- 모델에 ELMo를 추가했을 때가 추가하지 않았을 때보다 학습 속도가 빠름

7. Analysis

- 학습된 weights 시각화



- Softmax-normalized 학습 계층 가중치를 시각화한 것
- Input layer에서는 첫번째 layer에 더 높은 가중치를 둠
- Output layer에서는 모든 layer에 균형있게 가중치를 둠

8. Conclusion

- biLM 으로부터 깊은 문맥 의존 representation을 학습하는 일반적인 접근법을 도입
- 광범위한 NLP task에서 ELMo 를 적용했을 때 성능 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 성능 향상이 있음

Thank You

감사합니다.