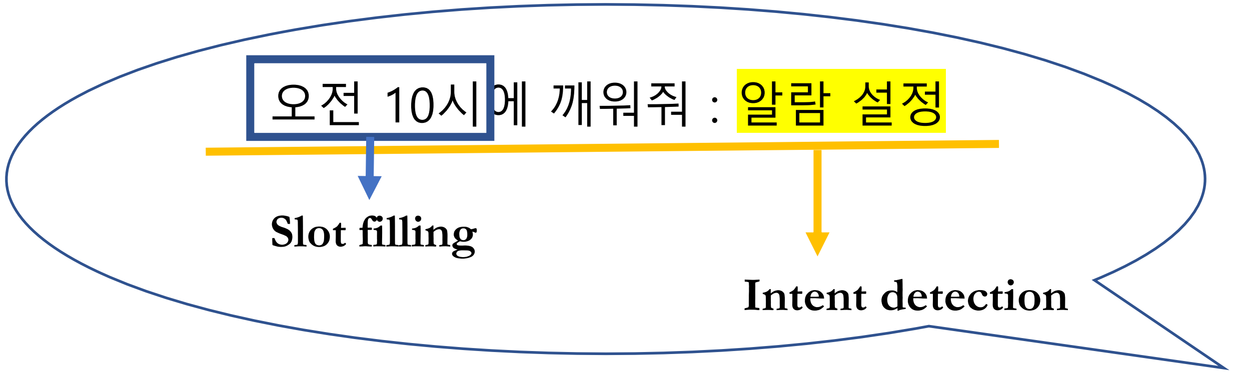
A Stack-Propagation Framework with Token-Level Intent Detection for Spoken Language Understanding

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, Ting Liu

전북대학교 IT정보공학과
202018392
박나현

Spoken language understanding (SLU)

- 대화 시스템에서 발화로부터 의미나 의도를 추론
- **Two main task**
  - **Intent detection**: 발화자의 의도 파악
  - **Slot filling**: 해당 의도를 해결하는데 slot 파악
    - Slot: 발화에 포함된 task와 관련된 의미 있는 정보



Intent detection & Slot filling

- 두 task는 원래 분리되어 실행

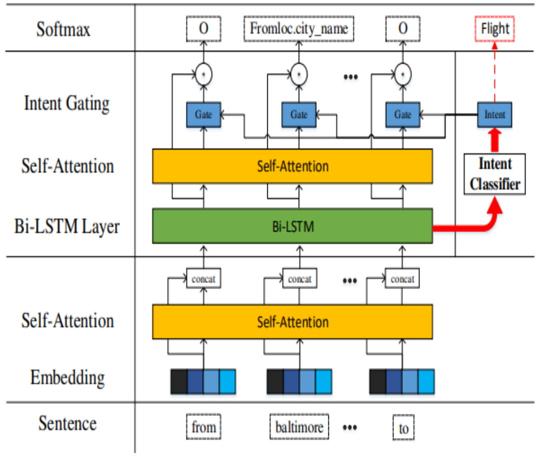
Sentence	*watch*	*action*	*movie*
Gold Slots	O	B-movie_name	I-movie_name
Gold Intent	WatchMovie		

- Slot이 intent에 의존을 자주하여 둘은 독립적이라 볼 수는 없음
  - BIO format
    - Beginning (B), inside (I) of each slot label, one for tokens outside (O) any slot label
  - Slot: movie_name > music_name (intent: watchmovie)
- Slot filling 위한 intent information 통합 바람직

기존 model

- Some joint models are proposed based on the multi-task learning framework
  - 의미론적 수준보다는 표면 수준에서 매개 변수를 공유함으로써 공동 학습만 고려
- 최근 연구
  - intent information for slot filling explicitly in joint model, gate mechanism…
  - 문제점
    - They all adopt gate vector to incorporate the intent information

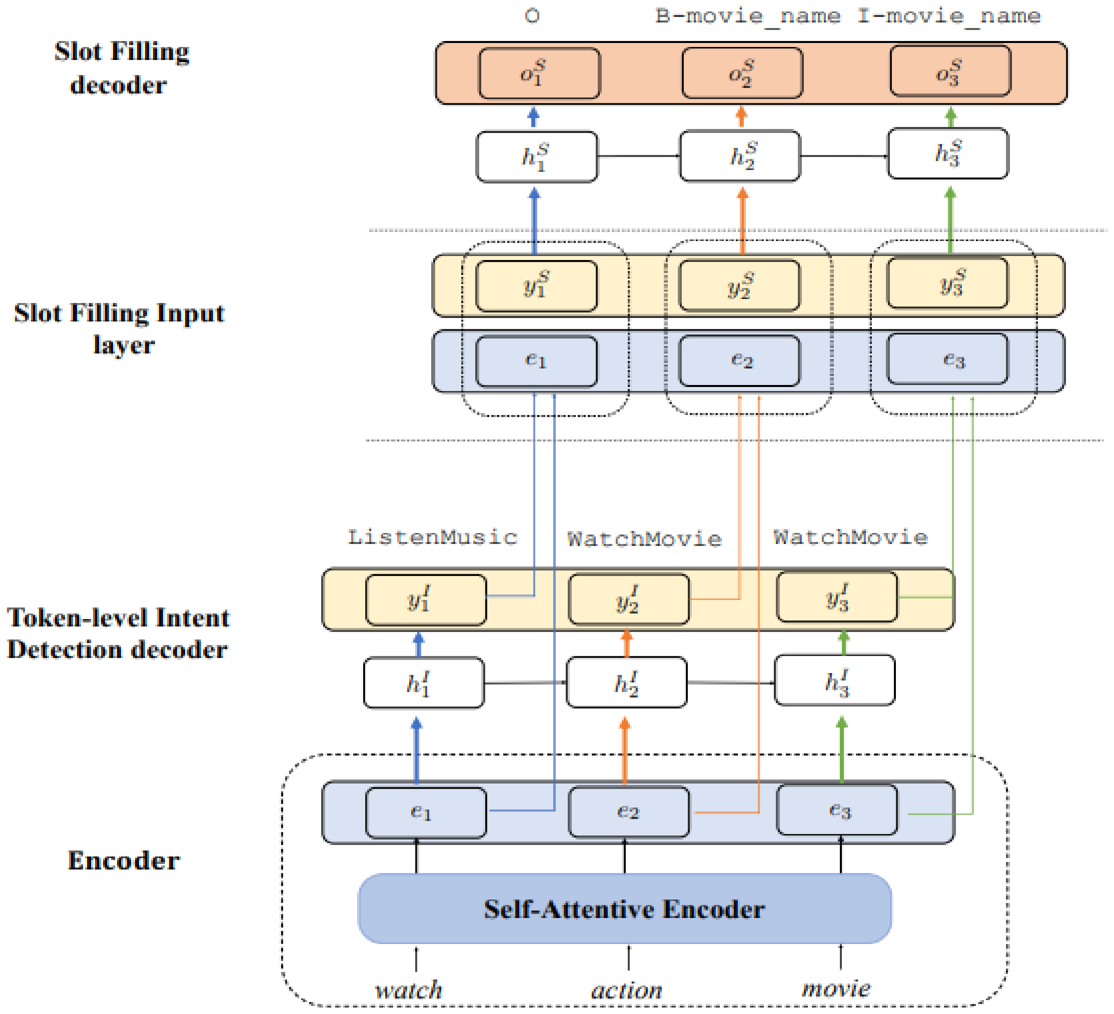
$$s_t = \text{Self-Attention}(h_t, H)$$
$$h_t^* = \text{MLP}([s_t, v^{int}])$$
$$o_t = h_t \odot h_t^*$$



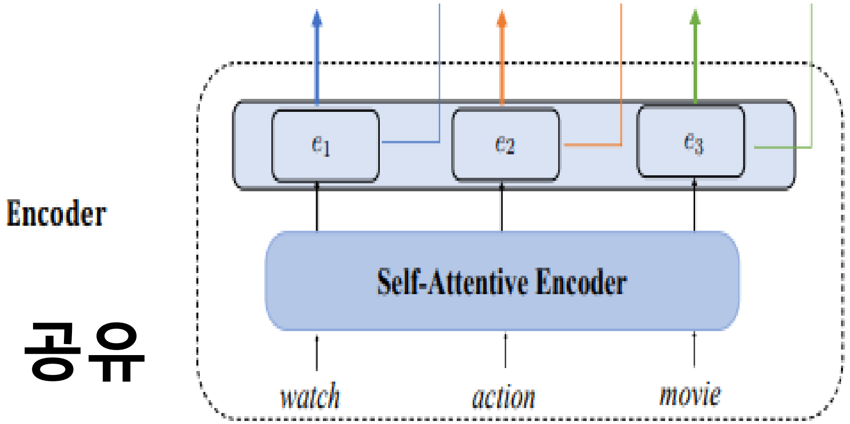
    - The utterance-level intent information they use for slot filling may mislead the prediction for all slots in an utterance if the predicted utterance-level intent is incorrect.

The contribution of this work

- Stack propagation framework in SLU task
- Token-level intent detection for stack-propagation framework
- Extensive experiments demonstrating the benefit of our proposed framework
- Explore and analyze the effect of incorporating BERT in SLU task



Self-Attentive Encoder

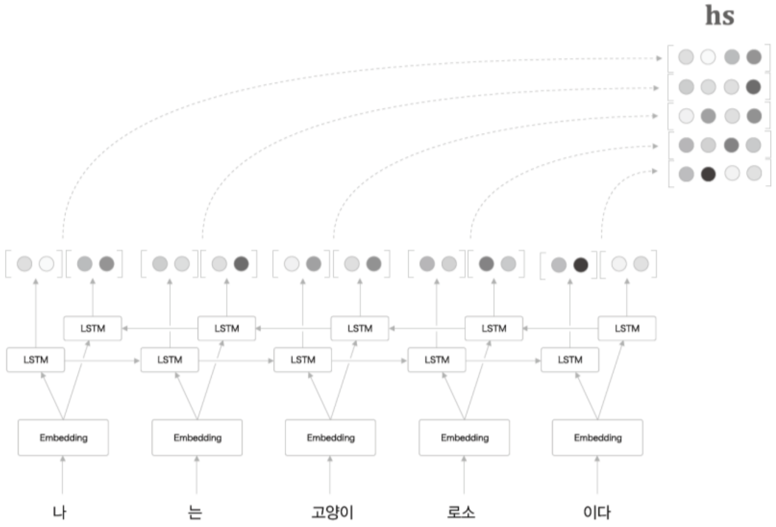


intent detection task와 slot filling task 1개 encoder를 공유

BiLSTM 사용

- Input utterance $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_T)$
- $H = (h_1, h_2, \ldots, h_T)$

  $(h_i = BiLSTM(\phi^{emb}(x_i), h_{i-1}))$



Self-Attention 사용

- Attention
  - Query-Key-Value architecture 기반
- Query, Key, Value가 동일
- Capture the contextual information for each token

$$\mathbf{C} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}.$$

$$\mathbf{E} = \mathbf{H} \oplus \mathbf{C}$$

$$\mathbf{E} = (\mathbf{e}_1, ..., \mathbf{e}_T)$$

Q: query
K:keys
V: values
C: self-attention ouput
E: final encoding representation

Token-Level intent detection decoder

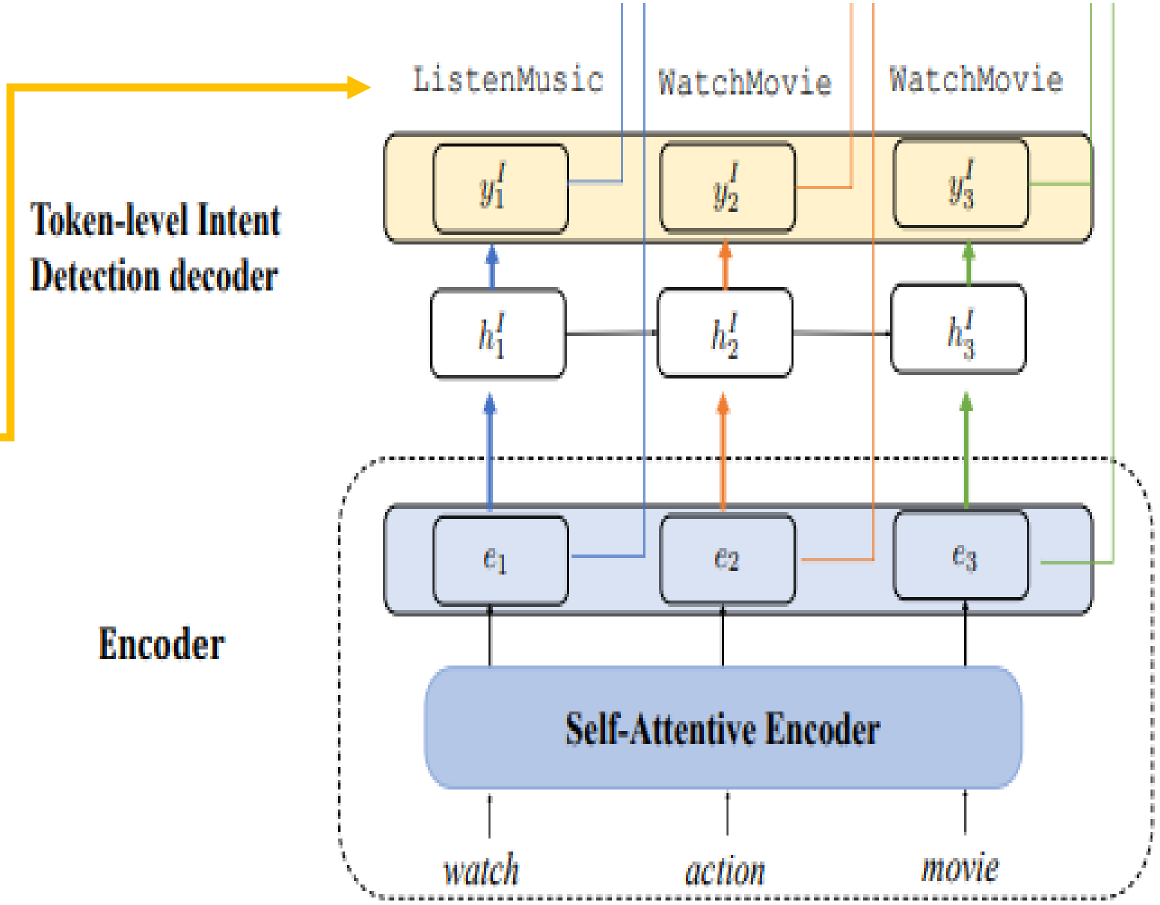
- 단방향 LSTM 사용

$$\mathbf{h}_i^I = f\left(\mathbf{h}_{i-1}^I, \mathbf{y}_{i-1}^I, \mathbf{e}_i\right)$$

$$\mathbf{y}_i^I = \mathrm{softmax}\left(\mathbf{W}_h^I \mathbf{h}_i^I\right),$$

$$o_i^I = \mathrm{argmax}(\mathbf{y}_i^I),$$

Intent label of ith token



- Final utterance result

$$o^I = \mathrm{argmax} \sum_{i=1}^{m} \sum_{j=1}^{n_I} \alpha_j \mathbb{1}[o_i^I = j],$$

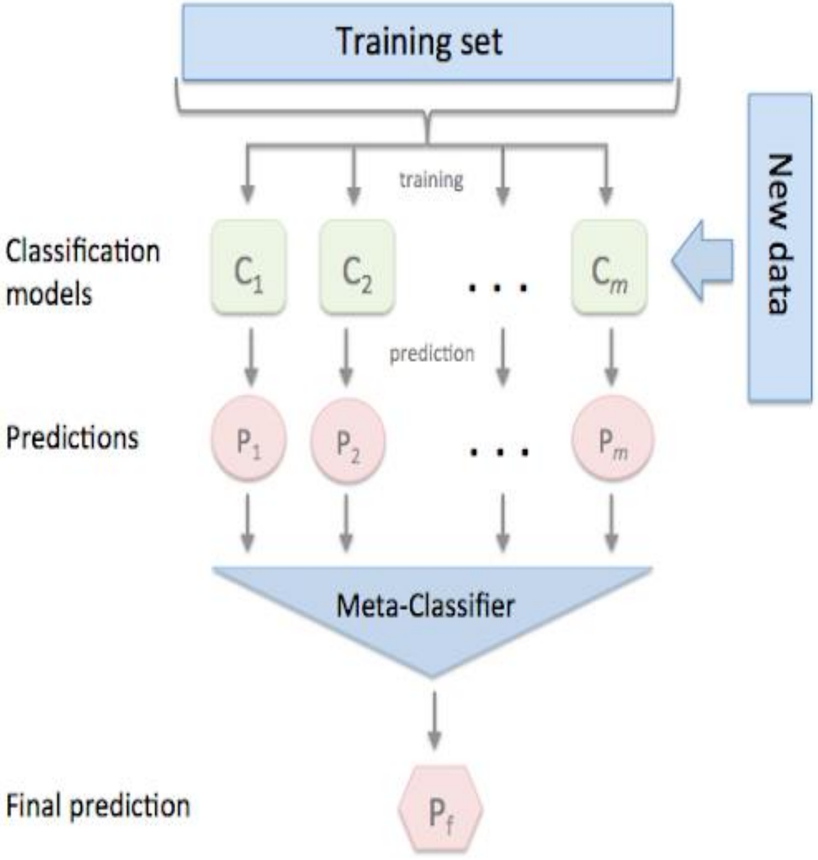
α_j: Denotes a 0-1 vector alpha of which the jth unit is one and the other s are zero.

Token-Level intent detection decoder

- 장점
  - 발화의 몇몇 token이 의도가 잘못 예측되어도 제대로 의도가 예측된 token이 slot filling할 때 여전히 유용
  - Predicted variance가 줄어들고 intent detection의 성능이 향상
    - 각 token이 전체 발화 정보를 담고 있어 각 token에서 예측된 intent를 전체 발화 intent라고 볼 수 있기 때문

Stack-propagation for Slot Filling

- **Stacking**
  - 앙상블(ensemble) 종류
    - 앙상블: 하나의 모델이 아닌 여러 학습 모델을 활용해 학습
  - 개별 알고리즘의 예측한 데이터를 기반으로 다시 예측을 수행하는 방법



Stack-propagation for Slot Filling

- **Multi-task framework**
  - 여러 task 동시에 학습
  - 각 task feature 사용 불가

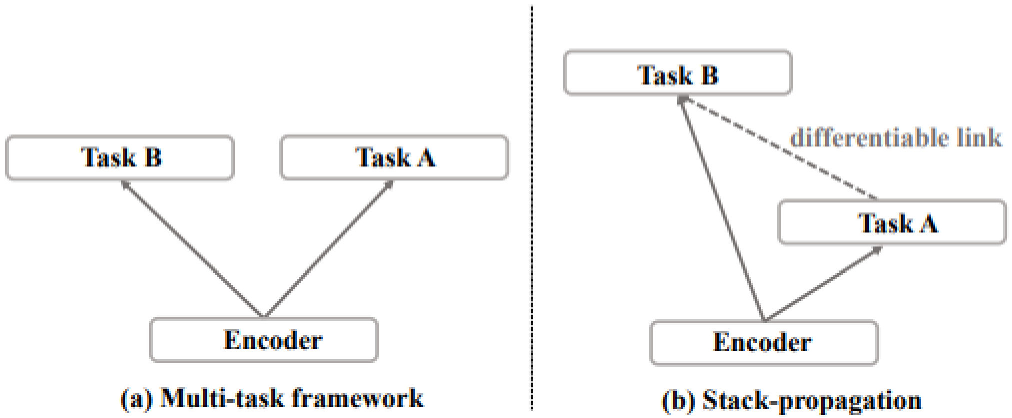
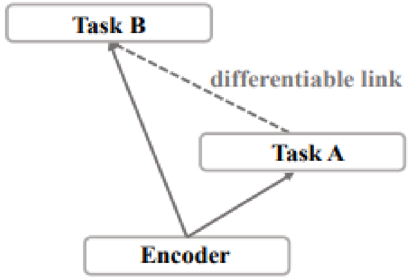


Figure 1: Multi-task framework vs. Stack-Propagation.

- **Stack-propagation**
  - A continuous from of stacking that allows for easy backpropagation down the pipeline across multiple tasks
  - Feature 공유 가능

Stack-propagation for Slot Filling

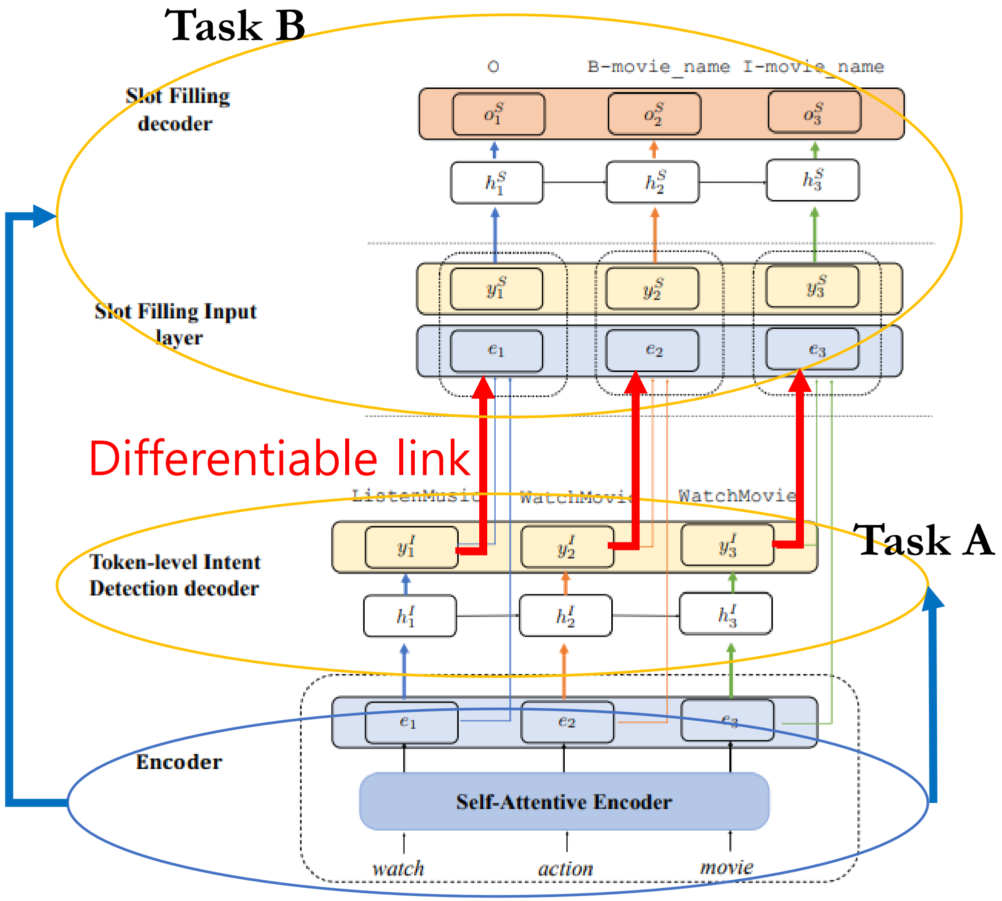


(b) Stack-propagation

- 장점
  - Explicit intent information 직접 활용
- Intent detection decoder와 다른 단방향 LSTM 사용

$$\mathbf{h}_i^S = f\left(\mathbf{h}_{i-1}^S, \mathbf{y}_{i-1}^S, \mathbf{y}_i^I \oplus \mathbf{e}_i\right)$$

$$\begin{aligned} \mathbf{y}_i^S &= \operatorname{softmax}\left(\mathbf{W}_h^S \mathbf{h}_i^S\right) \\ o_i^S &= \operatorname{argmax}(\mathbf{y}_i^S), \end{aligned}$$



Joint Training

- 여러 개의 loss들을 하나의 값으로 더해서 최종 loss로 사용하는 훈련
  - Intent detection objection $\mathcal{L}_1 \triangleq -\sum_{j=1}^{m}\sum_{i=1}^{n_I} \hat{\mathbf{y}}_j^{i,I} \log\left(\mathbf{y}_j^{i,I}\right)$
  - Slot filling task objection $\mathcal{L}_2 \triangleq -\sum_{j=1}^{m}\sum_{i=1}^{n_S} \hat{\mathbf{y}}_j^{i,S} \log\left(\mathbf{y}_j^{i,S}\right)$

Final joint objective

$$\mathcal{L}_\theta = \mathcal{L}_1 + \mathcal{L}_2$$

- Joint loss function을 통해 shared self-attentive의해 학습된 shared representation 들은 두 task 함께 고려 가능
- NLLLOSS
  - 입력값 x와 파라미터 θ가 주어졌을 때 정답 Y가 나타낼 확률
  - Softmax 를 추가하면 cross entropy

Thank you