NON-AUTOREGRESSIVE DIALOG STATE TRACKING SUMMARY

Paper Link: https://arxiv.org/abs/2002.08024

Summary By: Hyungrak kim

Email: jikuaij@gmail.com

Paper Contents

ABSTRACT

Task Oriented Dialogue 를 위해서 Dialogue State Tracking(DST)는 Open-Vocabulary 와 Generation approach 로 점점 발전해왔고, 복잡한 Task 대화에서 좋은 성능을 냈다. 하지만 2 가지 문제점 이 있는데 1) 도메인과 slot 간에 잠재적인 종속성의 Signal 을 모델이 학습하는 것이 어려웠고, 2) Auto-regressive approaches 들은 multi-domain, multi-turn 에서 time cost 가 많이 든다. 이 논문에서는 2 가지 문제점을 해결하기 위해서 새로운 Non-Autoregressive Dialog State Tracking(NADST)를 제안한다.

Non-Autoregressive Dialogue State Tracking(NADST)는 잠재적으로 domain 과 slot 간에 의존성 Factor 를 최적화하여 dialogue state 에서 더 완벽한 예측을 가능하게 한다. 그리고 Non-Autoregressive 는 dialogue response generation 에서 DST 의 Latency 를 줄이기 위해서 Decoding 을 병렬로 한다. 그리고 token level 에서 slot 의존성, slot 과 domain 의존성을 모두 Detect 한다.

논문의 결과로써 MultiWOZ2.1 Corpus 의 모든 도메인에서 SOTA 를 달성했다. 그리고 Latency 부분에서 이전 dialogue history 가 확장되는 시간이 늘어날수록 다른 최신모델보다 Latency 가 줄어드는 것을 확인 할 수 있었다.

1. INTRODUCTION

Task oriented dialogues 는 사람과 Agent 대화에서 Restaurant 이나 호텔을 예약할 때 많이 쓰인다. 사람과의 대화에서 Sequence 의 Slot 이 중요한 정보이자, Goal oriented 가 명확하게 나와있다. 이런 Slot 의 정보를 잘 파악하기 위해서 각 Dialog state 를 추적하는 것이 Dialog State Tracking(DST) 이다. Dialog State 는 (Slot, Value)로 구성되어 있는데 "동네 헬스장을 가고 싶어"에서 Slot 은 장소 type, Value 는 헬스장이다. 이를 다시 구성하면 Dialog State 는 (slot, value) = ("장소 type", "헬스장")이 된다.

DST 모델은 2 가지 타입으로 나누어진다. 1) fixed-vocabulary 와 2) open-vocabulary 이다. 1) fixed-vocabulary 는 알고있는 Slot ontology 를 이용해서 각 Dialog State 의 후보를 생성한다. 2) open-vocabulary 경우 최근 많이 제안되는 방법으로 **Dialog History** 로부터 Dialog State 후보를 생성한다.

보통 Open-vocabulary DST 모델은 autoregressive[1] encoder 와 decoder 를 사용하는데, 단점으로 expensive time cost 가 많이 드는 문제가 있다. 보통 time cost 가 많이 발생하는 경우는 1) dialogue history 길이, 대화 턴 길이, 2) slot value 길이 등으로 dialogue 가 복잡할 수록 그리고 Multi domain 일 수록 time cost 가 많이 들어간다.

따라서 이 논문에서는 time cost 를 minimize 하면서 model 의 정확도를 올리는 방법을 제시한다. Fertility[2] 개념을 적용해서 non-autoregressive 모델을 만든다. 첫번째로 concatenated slot value 의 sequence 로 dialogue state 를 재구성하고, 이 결과 sequence 는 Fertility concept 을 적용할 수 있는 고유한 구조적 representation 을 가지고 있다. 그리고 이 구조는 individual slot values 의 boundary 로 정의된다.

이 모델은 2 개의 decoding process 가 존재한다. 1) input dialogue history 에서 관련된 signal 학습하고, 각 slot value 의 representation 에 대한 fertility 를 생성한다. 2) 이 predicted fertility 는 구조화된 sequence form 으로 사용되는데, sequence 는 multiple sub-sequence 와 각 represented as(slot token x slot fertility) 구성되어 있다. 그리고 결과 sequence 는 한번의 target dialogue state 에 모든 token 을 생성하기 위한 2 번째 decoder 의 input 으로 사용된다.

---

autoregressive[1]: 특정 t 에 token 을 생성하기 위해서 t 이전 발생한 token 과의 조건부 확률을 구해서 계산

Fertility[2]: non-Autoregressive Decoding 을 위해서 decode 에 들어가는 Input sequence 형태로 복사되는 단어 횟수를 말함

또한 이 모델은 slot level 과 token level 모두를 고려한 모델인데, slot 사이의 signal 을 고려하는 것이 필요하다. 하지만 보통 DST 는 그러지 못하고 있다. 예를 들면 기차 출발 장소와 기차 도착 장소는 같은 value 가 아니라는 signal 을 고려해야 한다는 것이다. 그리고 이게 정확도 측면에서 좋은 향상을 준다.

모델 평가 metric 은 Joint Accuracy 를 사용하고, state level 에서 정확도를 측정한다.

이 논문의 Contribution

1. Non-Autoregressive Dialogue State Tracking(NADST)제안하는데 이 모델은 Dialogue State 의 완벽한 set 을 디코딩하기 위해 slot 들 간에 inter-dependency 를 학습한다.
2. Real time 을 위한 Low latency 와 token level 뿐 아니라 slot level 의 Dependency 를 capture 한다.
3. Multi-domain Task-oriented MultiWOZ 2.1 에서 SOTA 를 기록했고, Inference latency 를 줄였다.
4. Ablation study 를 통해서 이 논문에서 제안하는 모델이 slot 간에 잠재적인 Signal 을 학습하고, 또 효과가 있다는 것을 보여준다. 또한 slot 의 sets 을 더 정확하게 dialogue domain 에서 생성한다는 것을 보여준다.

2. RELATED WORK

DST 와 Non-autoregressive decoding 에 대한 RELATED WORK 를 설명한다.

2.1 Dialogue State Tracking

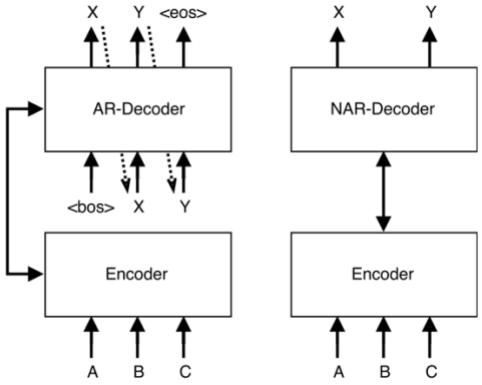
정통적으로 DST 는 NLU 와 결합되어 사용했었지만, 최근에는 NLU 와 DST 의 Combine 한다. credit assignment problem(CAP[3])을 해결하기 위해서 NLU 를 제거했다. 그리고 fixed vocabulary 에서 retrieval-based method 인데, 기본적으로 구축된 Slot ontology 로부터 모든 후보 pair(slot, value)를 고려하여 각 pair 가 가장 높은 확률을 계산한다. 최근 work 에서는 Open-Vocabulary 를 사용하는데, dialogue history 로부터 모든 후보 Pair 을 generate 한다. 이 논문에서는 Open-Vocabulary DST 이며, 다른 모델과 다른 점은 complete set 로써 dialogue state decoding 하기 위해서 slot 과 domain 의존성을 고려한다.

2.2 Non-AUTOREGRESSIVE DECODING

대부분 Non-Autoregressive Decoding 은 NMT 에서 translation 속도를 빠르게 하기 위해서 사용되어 왔다. 그리고 Decoding process 에서 longer sequence 에서 source sequence 를 projecting 함으로써 sequence labeling task 를 수행한다. 여기에 regularization term 을 추가하여 정확도 향상을 했다. 보통 NMT 에서 문제가 되는 것은 Large number of sequential latent variable 이다.(fertility sequence, projected target sequence). 이 latent variable 은 보통 non or semi autoregressive 를 위한 signals 를 supporting 하는데 사용된다. 그리고 이 논문에서는 slot value 의 concatenation 으로 sub-sequence 들과 같은 구조화된 sequence 로 dialogue state 를 reformulate 했다. 그리고 NMT 와 다르게 dialogue state 는 Dialogue State annotation 을 쉽게 추론할 수 있다. 따라서 NMT 에 비교해서 쉽게 DST 에 Non-Autoregressive approach 를 적용할 수 있다. 그리고 이 눈문이 DST generation based 에 Non-Autoregressive framework 를 적용하는 첫번째 논문일 것이다. 이 논문 Approach 가 slot 간에 joint state tracking 을 함으로써, 결과적으로 추론 시 높은 퍼포먼스와 low latency 를 가지는 모델임을 실험결과로 증명한다.

CAP[3]: network 에서 특정 문제를 예측할 때 얼마나 credit 을 주느냐 아니면 blame 주느냐를 생각했을 때 확률적으로 예측한 것을 신뢰하기 어렵다는 뜻이다. 위의 경우 예를 들어서 대화 Sequence 에서 "나는 철수 집에 방문했다"에서 철수 집이 어떤 Location 의 Slot 이란 것을 예측할 때 이전 대화 History 에 "철수 집은 은평구야"라는 정보를 같이 계산하면 철수의 집 Location 을 알 수 있지만, 현재 Sequence 에서는 NLU 로는 Location 을 알 수 없다. 따라서 Credit 을 Assign 하는 것이 부정확하다는 뜻이고, 이는 CAP 로 해석될 수 있다. 현재 문장에서 Slot 과 Value 를 찾는 NLU 를 사용하지 않고, DST 를 사용하는 것 같다는 생각이다.

추가적으로 AUTO-REGRESSIVE DECODING 설명



기존 Encoder Decoder 의 연결 구조를 병렬적으로 계산한다. 하지만 이렇게 하면 당연히 학습을 위해서 Multi-Modal 를 사용해야 하는데, 학습이 잘 되지 않는다. 따라서 Fertility 개념을 넣는데, 쉽게 설명하면 병렬적인 학습을 연결해주는 Fertility 가 Latent Variable z 을 대체하였다고 보면 된다. 또한 이를 통해 Multi-modal 학습이 잘되게 도와준다. 이 논문은 선행적으로 "NON-AUTOREGRESSIVE NEURAL MACHINE TRANSLATION" 논문을 봐야하는데 같은 저자들이 썼고, 이전에 NMT 에 Non-Autoregressive 모델을 제안한 Approach 를 DST 에 분야에 가져왔다.

3. Approach

NADST model 은 3 개의 중요한 Part 가 있다. 1) Encoder, 2) Fertility Decoder, 3) State Decoder 로 이루어 져있다.

3.1 APPOACHE BASIC

수식:

$$\text{Dialogue History } X = (x_1, x_2, \ldots, x_N)$$

$$\text{(domain, slot) } X_{ds} = ((d_1, s_1), (d_2, s_2), \ldots, (d_G, s_H))$$

$$G = total\ number\ domain, \qquad H = total\ number\ slot$$

Slot values 의 concatenation:

$$Y^{d_i, s_j}: Y = (Y^{d_1, s_1}, \ldots, Y^{d_I, s_J}) = ({y_1}^{d_1, s_1}, {y_2}^{d_1, s_1}, \ldots {y_1}^{d_I, s_J}, {y_2}^{d_I, s_J})$$

Output dialogue state 의 domain, slot 수

Delexicalized dialogue history:

이전 Dialogue turn 으로부터 현재 Dialogue turn state 를 예측한다.

$$x_{n,\text{del}} = \text{delex}(x_n) = \begin{cases} \text{domain}_{\text{idx}}\text{-slot}_{\text{idx}}, & \text{if } x_n \subset \hat{Y}_{t-1}. \\ x_n, & \text{otherwise.} \end{cases} \quad (1)$$

$$\text{domain}_{\text{idx}} = X_{\text{ds}\times\text{fert}}[\text{idx}][0], \quad \text{slot}_{\text{idx}} = X_{\text{ds}\times\text{fert}}[\text{idx}][1], \quad \text{idx} = \text{Index}(x_n, \hat{Y}_{t-1}) \quad (2)$$

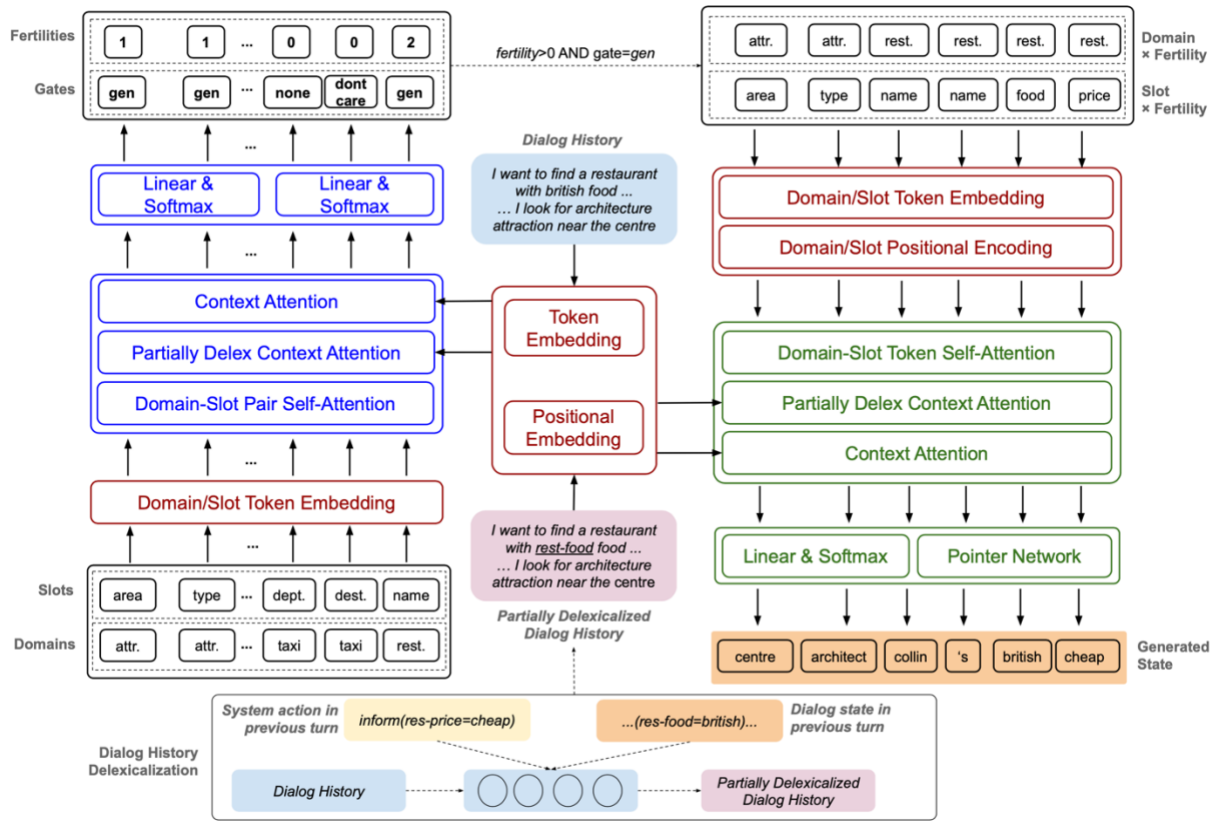
Ex) 예전 dialogue turn 에서 "저렴한"이란 단어가 "Hotel_pricerange" slot 으로 예측되었고, 현재 사용자가 "가격이 저렴한 호텔 찾아줘" 라고 했을 때, dialogue 를 Delexicalized 한다는 뜻은 "가격이 'Hotel_pricerange' 호텔 찾아줘"로 변형된다는 뜻이다.(참고. 1)

Figure.1 전체 NADST 구조도

(1) Red 색상 도형: encoder

(2) Blue 색상 도형: fertility decoder

(3) Green 색상 도형: state decoder



3.2 ENCODER

Embedding 된 Dialogue history X의 continuous representations:

$$Z = (z_1, z_2, z_3, \ldots\ldots z_N) \in \mathbb{R}^{N \times d}$$

Partially delexicalized dialogue history X_{del}의 continuous representation:

$$Z_{del} \in \mathbb{R}^{N \times d}$$

여기서 encode 된 Dialogue History Z는 Pointer network 를 사용한 dialogue state 생성에서 사용하기 위해서 메모리에 저장한다. 이는 Out of Vocabulary 문제를 해결하기 위한 방법으로 사용된다.

encode 된 각 (domain, slot) pair 를 continuous representation 나타내는 수식:

$$Z_{ds} \in \mathbb{R}^d$$

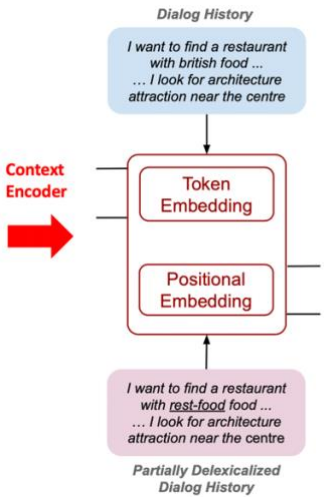
Z_{ds} 는 decoder 의 Input 으로 사용되는데 contextual signal 로 저장되어 2 개 decoding process 에서 slot prediction 과 fertility prediction 에 사용된다.

Context Encoder

Context Encoder 는 Token-level trainable embedding layer 와 layer normalization 가 포함되고. 또한 sine, cosine function 을 따르는 positional encoding 이 포함된다(Transformer 참고). 그리고 token-level vector 와 positional encoded vector 조합에 element-wise summation 이 사용된다. 그리고 embedding weight 가 공유되는데, 2 개의 dialogue history 를 embed 하기 위해서, 그리고 또 (2) fertility decoder 와 (3) state decoder input 으로 encode 하기 위해서 사용된다.

PE = positional Embedding

$$Z = Z_{\text{emb}} + \text{PE}(X) \in \mathbb{R}^{N \times d}$$
$$Z_{\text{del}} = Z_{\text{emb,del}} + \text{PE}(X_{\text{del}}) \in \mathbb{R}^{N \times d}$$



Domain and Slot Encoder

각 (Domain, Slot) pair 가 2 개 의 domain, slot embedding 백터를 사용해서 encode 된다. Domain 은 g , slot h 라 할 때, embedding 된 continuous representation 은 z_{d_g}, $z_{s_h} \in \mathbb{R}^d$ 라 한다.

최종 vector 들의 element-wise summation:

$$z_{d_g,s_h} = z_{d_g} + z_{s_h} \in \mathbb{R}^d$$

Embedding weights 를 2 개의 decoder 에서 공유하므로 이를 encoding 하는 식은 아래와 같다

$$\begin{aligned} Z_{\text{ds}} &= Z_{\text{emb,ds}} = z_{d_1,s_1} \oplus \ldots \oplus z_{d_G,s_H} \\ Z_{\text{ds}\times\text{fert}} &= Z_{\text{emb,ds}\times\text{fert}} + \text{PE}(X_{\text{ds}\times\text{fert}}) \\ Z_{\text{emb,ds}\times\text{fert}} &= (z_{d_1,s_1})^{Y_f^{d_1,s_1}} \oplus \ldots \oplus (z_{d_G,s_H})^{Y_f^{d_G,s_H}} \end{aligned}$$

여기서 $X_{ds\times fert}$ 는 (1) fertility Decoder 결과를 말함. Z_{ds} 는 입력 X_{ds}에 Embedding parameter 임. 또한 ⊕는 concatenation operation 을 뜻한다. 여기서 Transformer decoder 와 다른 점은 input sequence shift 가 없다는 점이다.(transformer decoder 와 유사하다는 뜻)

3.3 Fertility Decoder

Encoded 된 1) dialogue history Z, 2) delexicalized dialogue history Z_{del} , (domain, slot) pairs Z_{ds} 는 Attention Layer 를 통해서 Contextual Signal 이 학습이 된다. 그리고 이 눈문에서는 Multi-Head attention mechanism 을 사용한다.

Multi-Head Attention 수식:

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}V)$$

Attention 마지막에 ReLU Activation Function 을 사용한 Linear layer 를 사용하여 Decoding 을 수행한다. Fertility Decoder 에서는 3 개의 Attention 을 사용한다.

$$Z_{\text{ds}}^{\text{out}} = \text{Attention}(Z_{\text{ds}}, Z_{\text{ds}}, Z_{\text{ds}}) \in \mathbb{R}^{N \times d}$$
$$Z_{\text{ds}}^{\text{out}} = \text{Attention}(Z_{\text{ds}}^{\text{out}}, Z_{\text{del}}, Z_{\text{del}}) \in \mathbb{R}^{N \times d}$$
$$Z_{\text{ds}}^{\text{out}} = \text{Attention}(Z_{\text{ds}}^{\text{out}}, Z, Z) \in \mathbb{R}^{N \times d}$$

첫번째 Attention Layer 에서는 domain 과 slot 간에 잠재적인 dependency signal 를 attention 하고, 두번째 Delexicalized dialogue history Attention 에서는 real-value token 과 generalized domain-slot token 간에 mapping 되는 context signal 을 학습한다. 세번째는 dialogue history Attention 이며 두번째 Attention 과 같은 과정을 반복한다. 위와 같은 Attention 을 통해서 순차적으로 Dialogue history 에 중요한 slot, Domain 의 정보와 가장 관련이 높은 현재 dialogue token 의 Attention 을 학습한다.

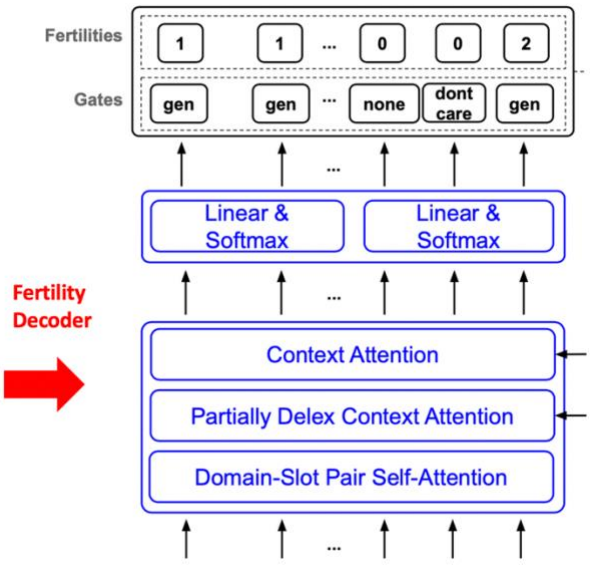
마지막 attention Layer 을 $Z_{ds}{}^{T_{fert}}$ 라 했을 때 독립적으로 2 개의 Linear Transformation 을 통해서 fertilities 와 gates 를 예측한다. ($W_{gate} \in \mathbb{R}^{d \times 3}$, $W_{fert} \in \mathbb{R}^{d \times 10}$)

$$P^{\text{gate}} = \text{softmax}(W_{\text{gate}} Z_{\text{ds}}^{T_{\text{fert}}})$$
$$P^{\text{fert}} = \text{softmax}(W_{\text{fert}} Z_{\text{ds}}^{T_{\text{fert}}})$$

2 개의 gate 와 fertility 에 대해서 standard cross entropy loss 를 이용해서 학습한다.

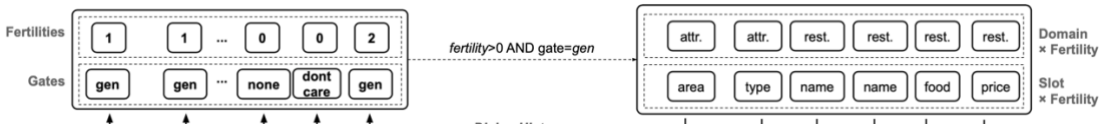
$$\mathcal{L}_{\text{gate}} = \sum_{d_g, s_h} -\log(P^{\text{gate}}(Y_g^{d_g, s_h}))$$
$$\mathcal{L}_{\text{fert}} = \sum_{d_g, s_h} -\log(P^{\text{fert}}(Y_f^{d_g, s_h}))$$

Fertility Decoder 구조도:



3.4 STATE DECODER

3.3 Fertility Decoder 에서 결과 구조도에서 Gates 에는 (Gen, None, Dontcare) 3 가지 label 로 예측된다. (domain, slot) pair 에서 Gen 으로 예측되고, fertility 가 0 보다 큰 Sequence 위치의 domain 과 slot 만 State Decoder 의 입력 domain, slot 으로 사용된다.



State Decoder 의 input 으로 Encoding 된 $Z_{ds\times fert}$ 이 사용되고, Attention 은 fertility decoder 와 같은 Attention 을 사용 하지만, domain,slot 보다 token level 에서 의존성이 capture 된다. 최종 Decoder 의 Attention 을 $Z_{ds\times fert}{}^{T_{state}}$ 라 할 때 state 예측은 아래와 같다.

$$P_{\text{vocab}}^{\text{state}} = \text{softmax}(W_{\text{state}} Z_{\text{ds}\times\text{fert}}^{T_{\text{state}}})$$

또한 OOV 문제를 해결하기 위해서 Point network 를 사용한다. 이는 저장해 놓은 Encoded Dialogue History 정보에서 OOV 한 state 를 찾겠다는 뜻과 같고, 수식으로는 state decoder output 과 Encoded

Dialogue History 의 dot product attention 을 수행하여 Dialogue history 에서 value 를 가르키는 네트워크를 학습한다.

$$P_{\text{ptr}}^{\text{state}} = \text{softmax}(Z_{\text{ds}\times\text{fert}}^{T_{\text{state}}} Z^T)$$

최종 state 를 예측하는 식은 P^{state}이다.

$$P^{\text{state}} = p_{\text{gen}}^{\text{state}} \times P_{\text{vocab}}^{\text{state}} + (1 - p_{\text{gen}}^{\text{state}}) \times P_{\text{ptr}}^{\text{state}}$$
$$p_{\text{gen}}^{\text{state}} = \text{sigmoid}(W_{\text{gen}} V_{\text{gen}})$$
$$V_{\text{gen}} = Z_{\text{ds}\times\text{fert}} \oplus Z_{\text{ds}\times\text{fert}}^{T_{\text{state}}} \oplus Z_{\text{exp}}$$

여기서 V_{gen}의 수식 중 Z_{exp}는 Dialogue History Z와 $Z_{ds\times fert}$ 의 Matching 된 Dimension 으로 확장된 Vector 이다.

마지막 Loss Function 개산은 아래와 같다.

$$\mathcal{L}_{\text{state}} = \sum_{d_g, s_h} \sum_{m=0}^{Y_f^{d_g, s_h}} -\log(P^{\text{state}}(y_m^{d_g, s_h}))$$

3.5 OPTIMIZATION

최종적으로 3 개의 Loss weighted sum 하여 모든 파라미터를 Jointly training 한다.

$$\mathcal{L} = \mathcal{L}_{\text{state}} + \alpha\mathcal{L}_{\text{gate}} + \beta\mathcal{L}_{\text{fert}}$$

where $\alpha \geq 0$ and $\beta \geq 0$ are hyper-parameters.

최종적으로 3 개의 Loss weighted sum 하여 모든 파라미터를 Jointly training 한다.

4 EXPERIMENTS

4.1 DATASET

Domain	attraction	hotel	restaurant	taxi	train	All
Slot	area name type	area bookday bookpeople bookstay internet name parking pricerange stars type	area bookday bookpeople booktime food name pricerange	arriveby departure destination leaveat	arriveby bookpeople day departure destination leaveat	-
train	3,381	3,103	2,717	3,813	1,654	8,438
val	416	484	401	438	207	1,000
test	394	494	395	437	195	1,000

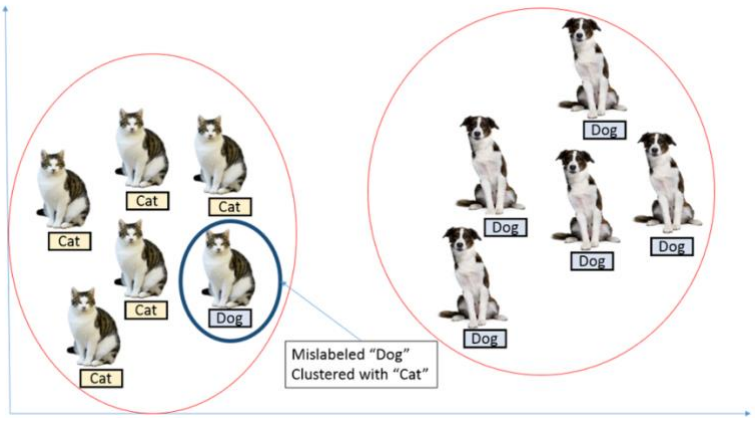
Table 7: Summary of MultiWOZ dataset 2.1

4.2 TRAINIG PROCEDURE

이 논문에서는 Label Smoothing 을 사용한다.

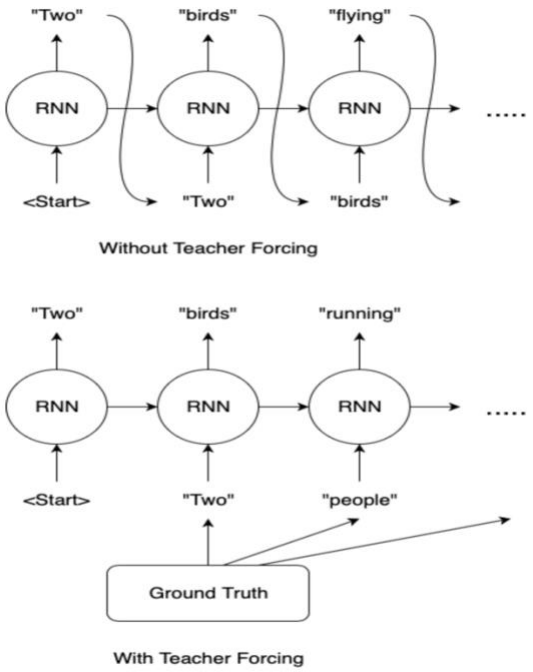
Label Smoothing:

일반적으로 하나의 라벨이 Mislabeled 되면 Loss 값이 크게 계산되고, 이는 모델의 일반화를 떨어트린다. 또한 이 라벨이 대량으로 있을 때 Mislabeled 은 모델 학습에 크게 영향을 미칠 수 있다. 따라서 label 을 smoothing 시켜서 이런 문제를 해결하는 것이 Label Smoothing 이다.



Label Smoothing 은 Dialogue State Y 를 예측하는데 사용되지만, Fertility Decoder 에서는 사용되지 않는다. 그리고 Training 동안 100% Teacher-forcing Learning 전략을 사용하는데, 이는 Multi-modal 학습이 잘 되지 않으니깐, 학습의 안전성을 위해서 추가하여 사용한 것 같다.

Teacher-Forcing Learning 전략:



이 전략을 state decoder 입력인 $X_{ds \times fert}$와 delexicalized history 에 적용된다. Inference 동안은 예측된 $\hat{Y}_{gate}$, $\hat{Y}_{fert}$ 로부터 $X_{ds \times fert}$가 구조화 된다. Parameter tuning 을 위해서 grid-search 를 사용했다. 그리고 이 모델은 github 에 공개되었다.

4.3 BASELINES

DST two baseline 이 존재한다. 1) Fixed Vocabulary , 2) open-vocabulary 중 1) Fixed Vocabulary 경우 inference 시 Unseen slot value 를 찾지 못하는 문제가 있다.

4.3.1 Fixed-Vocabulary

Fixed Vocabulary 기반 DST 모델

GLD, GCE, MDBT, FJST, HJST, SUMBT

4.3.2 Open-Vocabulary

Open-Vocabulary 기반 DST 모델

TSCP, DST Reader, HyST, TRADE(SOTA)

4.4 RESULTS

실험 결과로써 performance 측정 시 joint goal accuracy 와 slot accuracy 를 측정한다. Joint goal accuracy 는 정답 라벨과 모든 slot 의 예측된 value 가 모두 맞아야 정답이라고 인정하는 방식이다. Test 데이터 셋은 MultiWOZ 2.0 과 2.1 이있다. Table 2 결과와 같이 NADST 가 가장 좋은 성능을 내는 것을 보여주고 있다. 이는 Cross Domain 과 Cross-slot signal 을 학습함으로써 모델의 성능 향상을 시켰다고 볼 수 있다. 특히 Table 3 에서 restaurant 도메인에서 더 좋은 성능을 내고 있다.

Model	MultiWOZ2.1	MultiWOZ2.0
MDBT (Ramadan et al., 2018) †	-	15.57%
SpanPtr (Vinyals et al., 2015)	-	30.28%
GLAD (Zhong et al., 2018) †	-	35.57%
GCE (Nouri & Hosseini-Asl, 2018) †	-	36.27%
HJST (Eric et al., 2019) *	35.55%	38.40%
DST Reader (single) (Gao et al., 2019) *	36.40%	39.41%
DST Reader (ensemble) (Gao et al., 2019)	-	42.12%
TSCP (Lei et al., 2018)	37.12%	39.24%
FJST (Eric et al., 2019) *	38.00%	40.20%
HyST (ensemble) (Goel et al., 2019) *	38.10%	44.24%
SUMBT (Lee et al., 2019) †	-	46.65%
TRADE (Wu et al., 2019) *	45.60%	48.60%
Ours	**49.04%**	**50.52%**

Table 2: DST Joint Accuracy metric on MultiWOZ 2.1 and 2.0. †: results reported on MultiWOZ2.0 leaderboard. *: results reported by Eric et al. (2019). Best results are highlighted in bold.

Model	Joint Acc	Slot Acc
MDBT	17.98%	54.99%
SPanPtr	49.12%	87.89%
GLAD	53.23%	96.54%
GCE	60.93%	95.85%
TSCP	62.01%	97.32%
TRADE	65.35%	93.28%
Ours	**69.21%**	**98.84%**

Table 3: DST joint accuracy and slot accuracy on MultiWOZ2.0 *restaurant* domain. Baseline results (except TSCP) were from Wu et al. (2019).

	MultiWOZ2.1		MultiWOZ2.0	
Domain	**Joint Acc**	**Slot**	**Joint Acc**	**Slot**
Hotel	48.76%	97.70%	53.86%	97.75%
Train	62.36%	98.36%	58.58%	98.08%
Attraction	66.83%	98.89%	74.21%	99.19%
Restaurant	65.37%	98.78%	69.21%	98.84%
Taxi	33.80%	96.69%	34.94%	96.76%

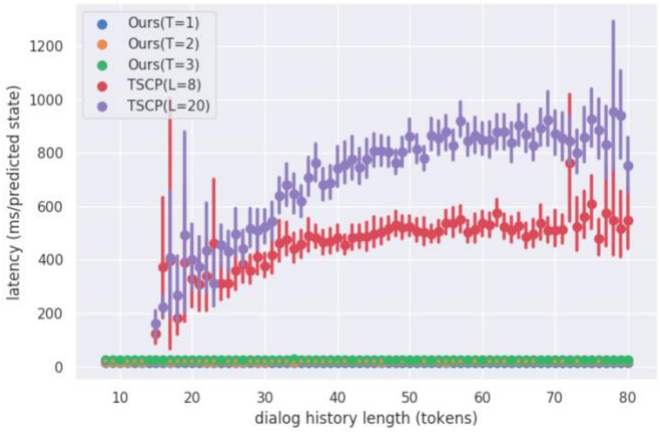
Table 8: Additional domain-specific results of our model in MultiWOZ2.0 and MultiWOZ2.1. The model performs best with the *restaurant* domain and worst with the *taxi* domain.

Latency 분석:

Latency 분석에서는 TRADE 모델과 TSCP 모델 2 개를 비교하였다. Test 는 Single GPU 에서 테스트를 하였고, $T = T_{fert} = T_{state} \rightarrow (1, 2, 3)$ 표시하여 각 fertility decoder 와 state decoder 를 추가했을 때 성능 차이가 있다. 당연히 T=3 일 때 정확도가 제일 높고, Latency 는 비교적 느리다. 하지만, 다른 모델에 비해서 수십 배에 달하는 Speed Up 이 있다는 것을 보여준다. 또한 TSCP 모델에 비교해서 Dialogue history 의 token 이 많아질 때 속도가 느린 것을 알 수 있는데 비교적 NADST 모델은 Token 수와 Latency 가 비례하지 않는 것을 볼 수 있다.

Model	**Joint Acc**	**Latency**	**Speed Up**
TRADE	45.60%	362.15	×2.12
TSCP (L=8)	32.15%	493.44	×1.56
TSCP (L=20)	37.12%	767.57	×1.00
Ours (T=1)	42.98%	**15.18**	**×50.56**
Ours (T=2)	45.78%	21.67	×35.42
Ours (T=3)	**49.04%**	27.31	×28.11

Table 4: Latency analysis on MultiWOZ2.1. Latency is reported in terms of wall-clock time in *ms* per prediction state.



Ablation Analysis:

모델 평가를 위해서 Joint Slot Accuracy, Slot Accuracy, Joint Fertility Accuracy, Joint Gate Accuracy, Oracle Joint Slot Accuracy, Oracle Slot Accuracy 를 측정한다. 여기서 Oracle 은 State Decoder 에 입력으로 들어가는 $X_{ds\times fert}$ and X_{del}를 predict 해서 구하지않고, 정답을 바로 입력으로 넣어줘서 state 를 예측한다. 이는 Teacher Forcing Learning 을 했기 때문에 훨씬 더 좋은 성능을 낸다고 해서 Oracle 이라고 논문에 붙인 듯 하다.

실험 결과를 보면 $X_{ds\times fert}$ 은 Positional Encoding(PE)를 제거 했을 때 Joint slot acc 성능이 떨어졌는데 그 이유는 PE 가 Non-autoregressive 의 Sequential 한 속성을 주입 시켜주기 때문이다. 두번째로 Slot gating 을 제거했을 때 성능이 조금 떨어졌는데, 이유는 Fertility 만으로 유효한 Slot 를 모두 찾기 어려웠기 때문이다. 세번째는 X_{del}를 제거했을 때 joint Fertility 의 정확도가 sharp 하게 떨어진 원인 때문이다. 마지막으로 Pointer Generation Network 을 제거 했을때 P_{vocab}^{state} 만 모델 성능에 영향을 주기 때문에 Unseen Slot Value 를 잘 찾지 못하는 문제가 있다. 따라서 성능이 떨어지는 것을 볼 수 있다.

X_{del}	Slot Gating	$PE(X_{ds\times fert})$	Pointer Gen.	Joint Gate Acc	Joint Fert. Acc	Joint Slot Acc	Slot Acc	Oracle Joint Slot Acc	Oracle Slot Acc
✓	✓	✓	✓	**66.65%**	63.18%	**49.04%**	**97.31%**	**73.44%**	**99.01%**
✓	✓		✓	59.23%	57.83%	19.56%	94.36%	72.12%	98.96%
✓		✓	✓	N/A	**64.23%**	48.74%	96.62%	73.01%	98.97%
	✓	✓	✓	48.23%	45.35%	39.45%	95.92%	66.27%	98.63%
✓(no sys. act)	✓	✓	✓	52.45%	56.81%	44.87%	96.95%	70.83%	98.74%
✓	✓	✓		63.19%	58.31%	43.46%	96.72%	64.37%	98.39%
	✓	✓		44.22%	42.01%	34.48%	95.89%	61.32%	98.24%
		✓		N/A	41.35%	33.52%	95.42%	60.99%	98.19%

Table 5: Ablation analysis on MultiWOZ 2.1 on 4 components: partially delexicalized dialogue history X_{del}, slot gating, positional encoding $PE(X_{ds\times fert})$, and pointer network.

Auto-regressive DST:

NADST 를 Fertility 를 제거하여 Auto-regressive 모델을 만들어 성능 차이가 있는지 실험하였다. 실험결과를 보면 별차이가 없는 것을 볼 수 있다. 이것이 Fertility NADST 에서 유용하게 쓰이고 있다는 것을 입증하는 것이고, 또한 slot gate 를 예측하는 것이 더 쉬운 Task 이다. 또한 전반적인 성능의 차이가 없는 것은 pair(domain, slot)의 High-level dependencies 를 학습하기 때문에 state 예측 정확도가 떨어지지 않는다.

MultiWOZ	Sys. Act	Joint Gate Acc	Joint Slot Acc	Slot Acc	Oracle Joint Slot Acc	Oracle Slot Acc
2.1	✓	65.89%	49.76%	97.40%	71.39%	98.92%
2.1		62.04%	46.57%	97.23%	66.72%	98.65%
2.0	✓	68.81%	50.08%	97.44%	79.04%	99.22%
2.0		65.27%	50.46%	97.43%	76.21%	99.08%

Table 6: Performance of auto-regressive model variants on MultiWOZ2.0 and 2.1. Fertility prediction is removed as fertility becomes redudant in auto-regressive models.

Visualization and Qualitative:

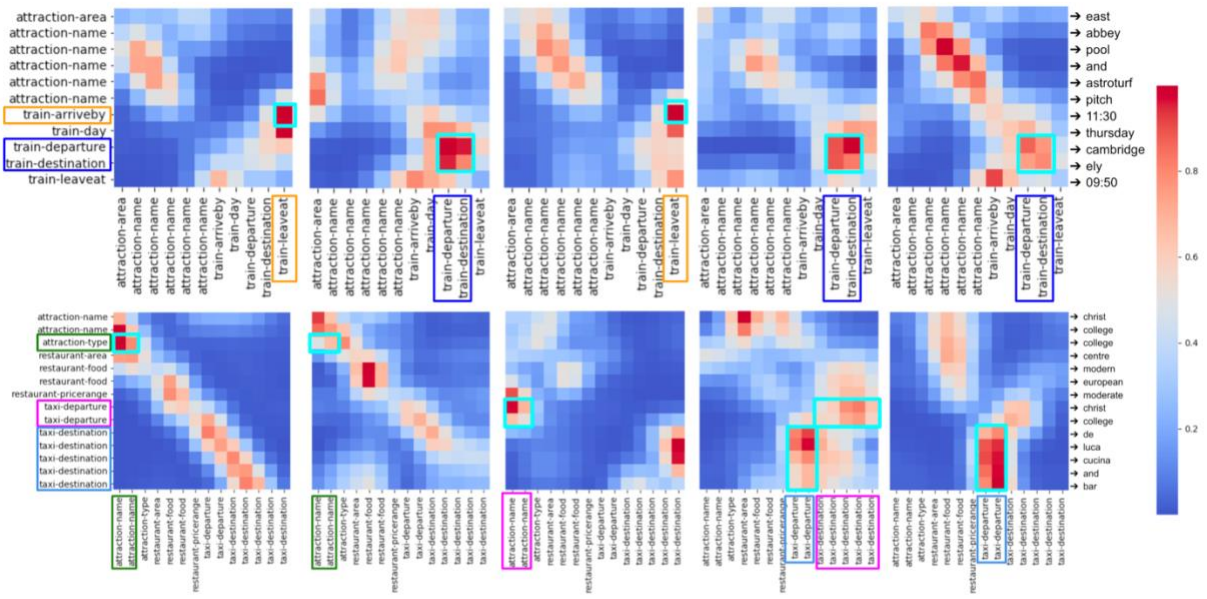


Figure 4: Heatmap visualization of self-attention scores of 5 heads between $Z_{ds \times fert}$ representations in the state decoder. The corresponding prediction output for each representation is presented on the right side. The examples are for the 6^{th} turn in dialogue ID MUL0536 (upper row) and PMUL3759 (lower row) in MultiWOZ2.1.

5. CONCLUSION

이 논문에서 제안한 NADST 는 Non-Autoregressive neural architecture 를 DST 에 적용했다. 또한 slot-level 과 token-level 의 의존성을 학습함으로써 individual slot accuracy 보다 joint slot accuracy 를 향상시켰다. 또한 parallel decoding 전략을 통해서 fast decoding 을 가능하게 했다. 또한 DST performance 측정에 사용되는 MultiWOZ corpus 실험에서 SOTA 를 달성하면서 정확도 측면과 inference latency 를 줄이는데 효과적으로 설계된 모델이라는 것을 증명했습니다.