Flexibly-Structured Task-Oriented Dialogue Modeling

Lei Shu, Piero Molino, Mahdi Namazifar, Hu Xu, Bing Liu, Huaixiu Zheng, Gokhan Tur

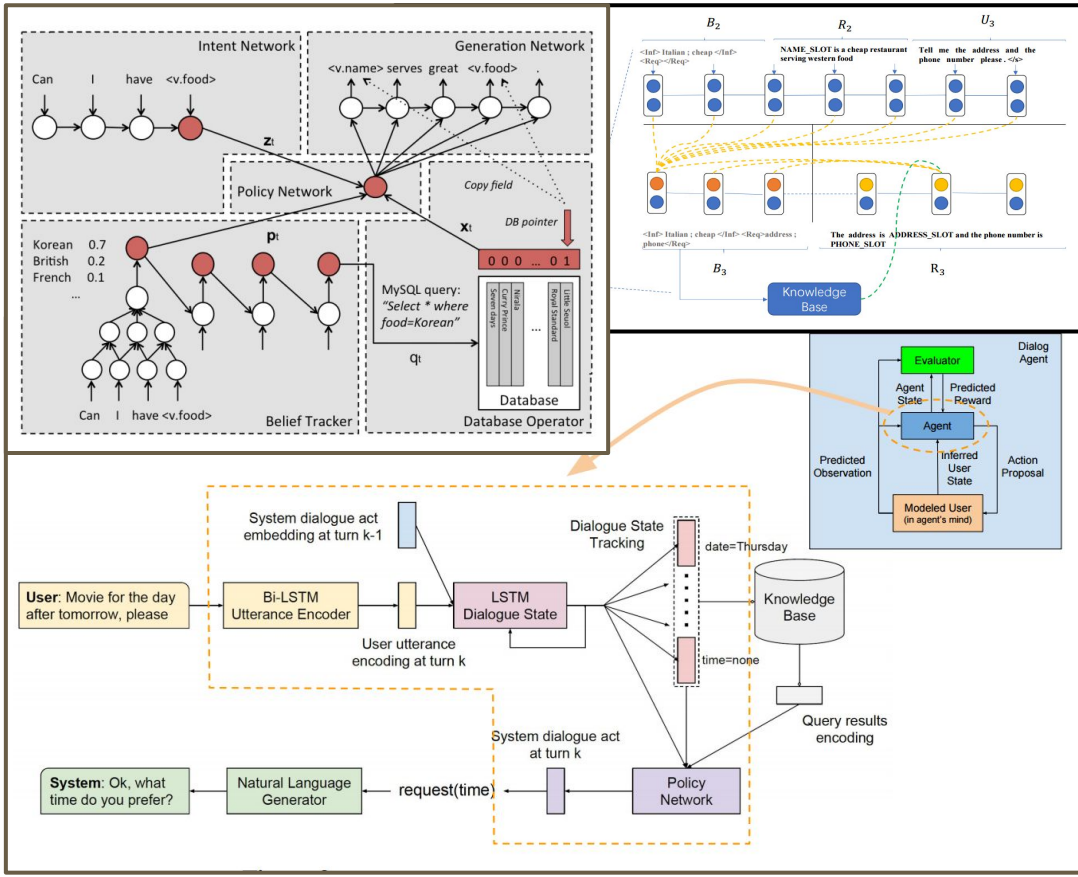
SIGDIAL 2019

Modularized End-to-End Dialogue Systems

Modules:
Natural language understanding,
dialogue state tracking,
knowledge base (KB) query,
dialogue policy engine,
response generation.

End-to-End fashion: modules are connected and trained together with text as input and text as output.

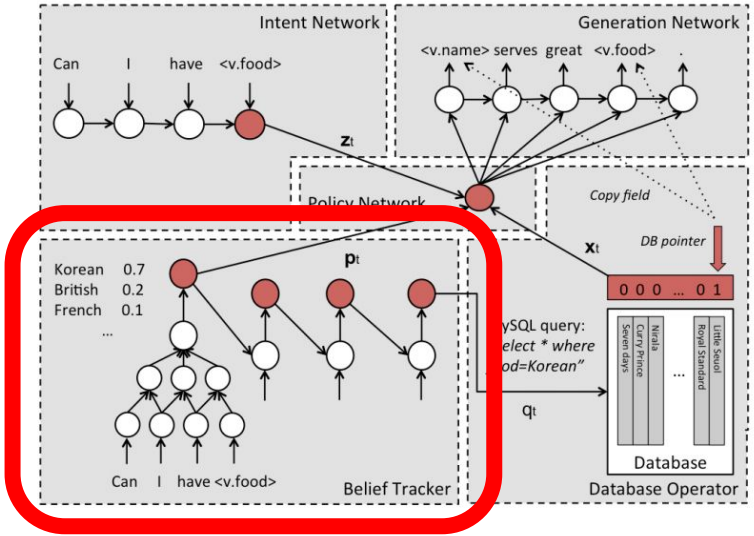
Advantage: reduce error propagation



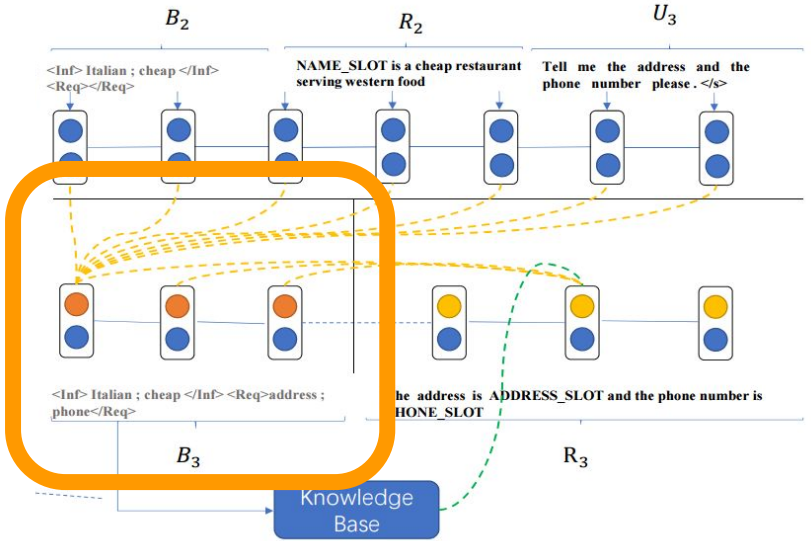
Wen et al 2017, Liu and Lane 2017, Lei et al 2018

Dialog State Tracking Module

It understands user's latest intention, memorizes dialog history, and updates dialog state at each turn.
The updated dialog state is for KB query and policy engine/response generation.
Two popular approaches: ***fully-structured*** and ***free-form***.



Wen et al 2017



Lei et al 2018

Fully-Structured Approach

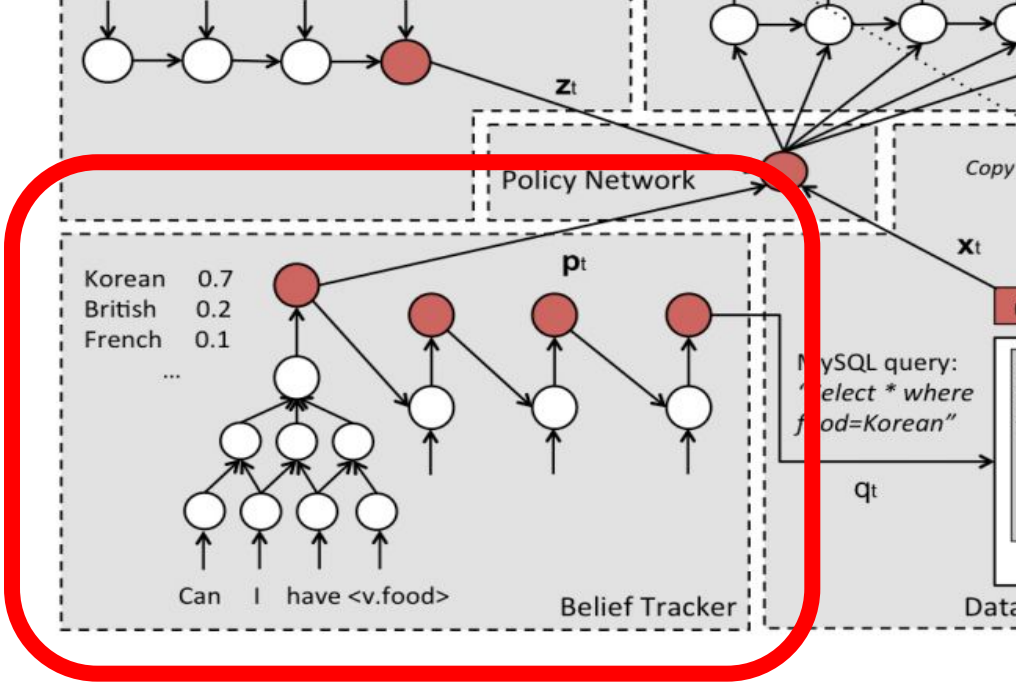
Use the full structure of the KB, both its schema and the values.

Assumption: the sets of informable slot values and requestable slots are fixed.

Network: multi-class classification.

Pros: value and slot are well aligned.

Cons: CANNOT adapt to dynamic KB and detect out-of-vocabulary values in the user's utterance.



Wen et al 2017

Free-Form Approach

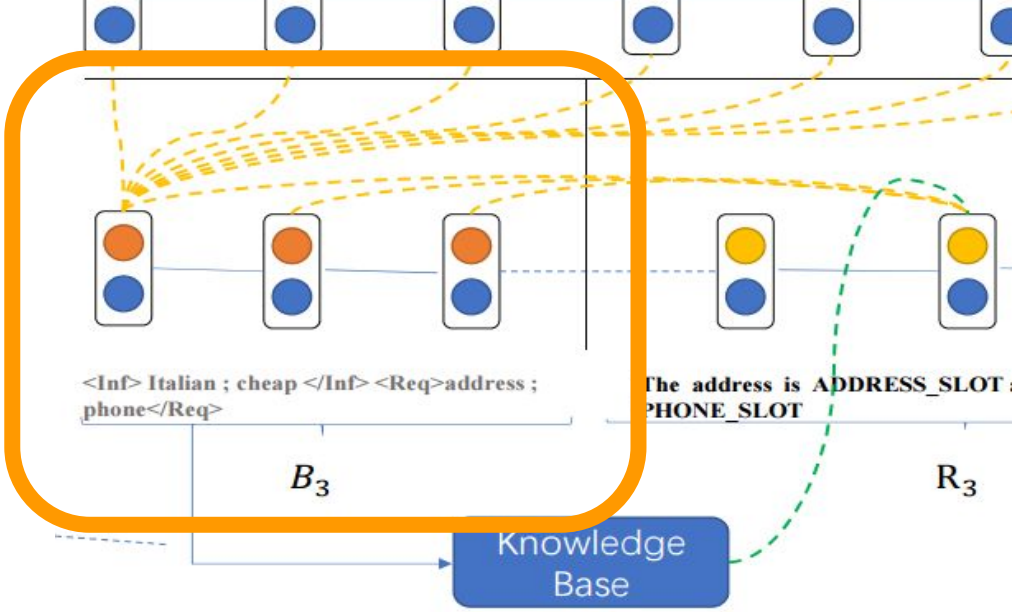
DO NOT integrate any information about the KB in the model architecture.

Network: sequence-to-sequence.

Pros:

(1) adaptable to new domains and changes in the content of the KB

(2) solve the out-of-vocabulary problem



Lei et al 2018

Cons:

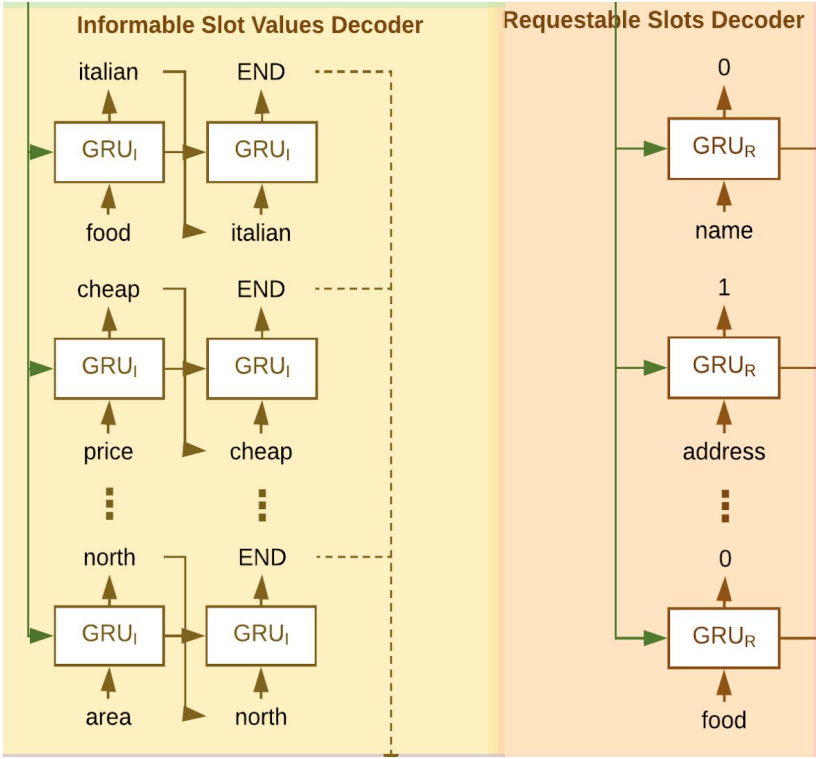
(1) value and slot are not aligned. E.g. in the travel booking system, “Chicago; Seattle”, can you tell which is the departure and which is the arrival?

(2) unwanted order of slots, e.g. “address; party”, “address; time; party”

(3) Invalid state like generate non-requestable-slot words

We propose: **Flexibly-Structured** DST Approach

Use only information in the schema of the knowledge base, but not information about the values.



Architecture:

(1) separate decoder for each informable slot (share parameter but different start token)
Informable Slot Value Decoder

(2) multi-label classifier for the requestable slots
Requestable Slot Decoder

Pros:

(1) slot and value are aligned

(2) solve the out-of-vocabulary problem

(3) adaptable to new domains and changes in the content of the KB

(4) No unwanted order of requestable slots and invalid state

How Nice Flexible-Structured DST is!

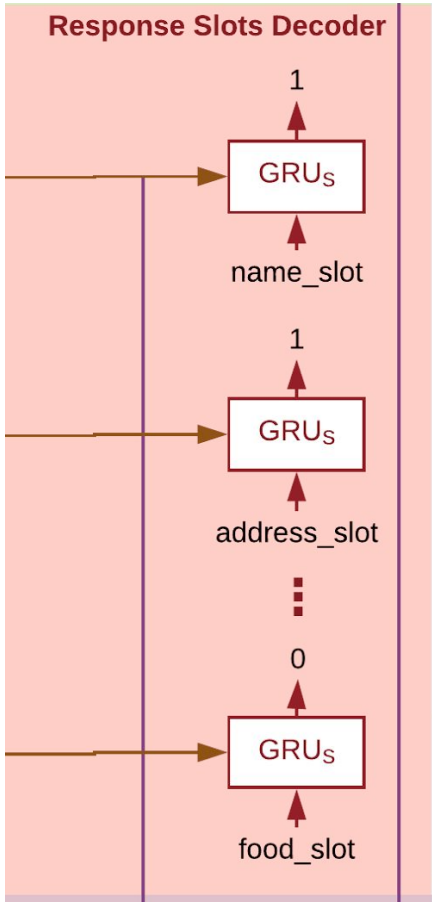
Explicitly assign values to slots like the fully structured approach, while also preserving the capability of dealing with OOV like the free-form approach.

It brings challenges in response generation:

(1) Is it possible to improve the response generation quality based on Flexible-Structured DST?

(2) How to incorporate the output from Flexible-Structured DST for response generation?

We propose: **Flexibly-Structured** Generation

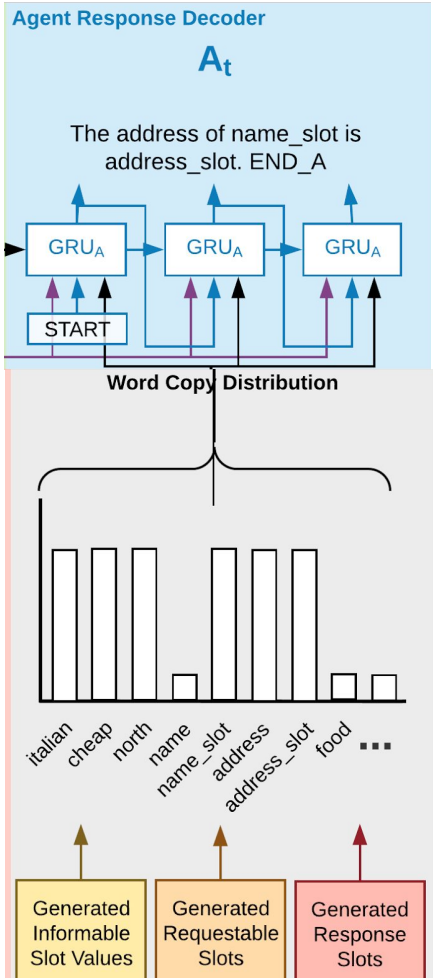


Response Slot Decoder

Response slots are the slot names that are expected to appear in a de-lexicalized response. Multi-label classifier is adopted for deciding what response slots to appear in the agent response.

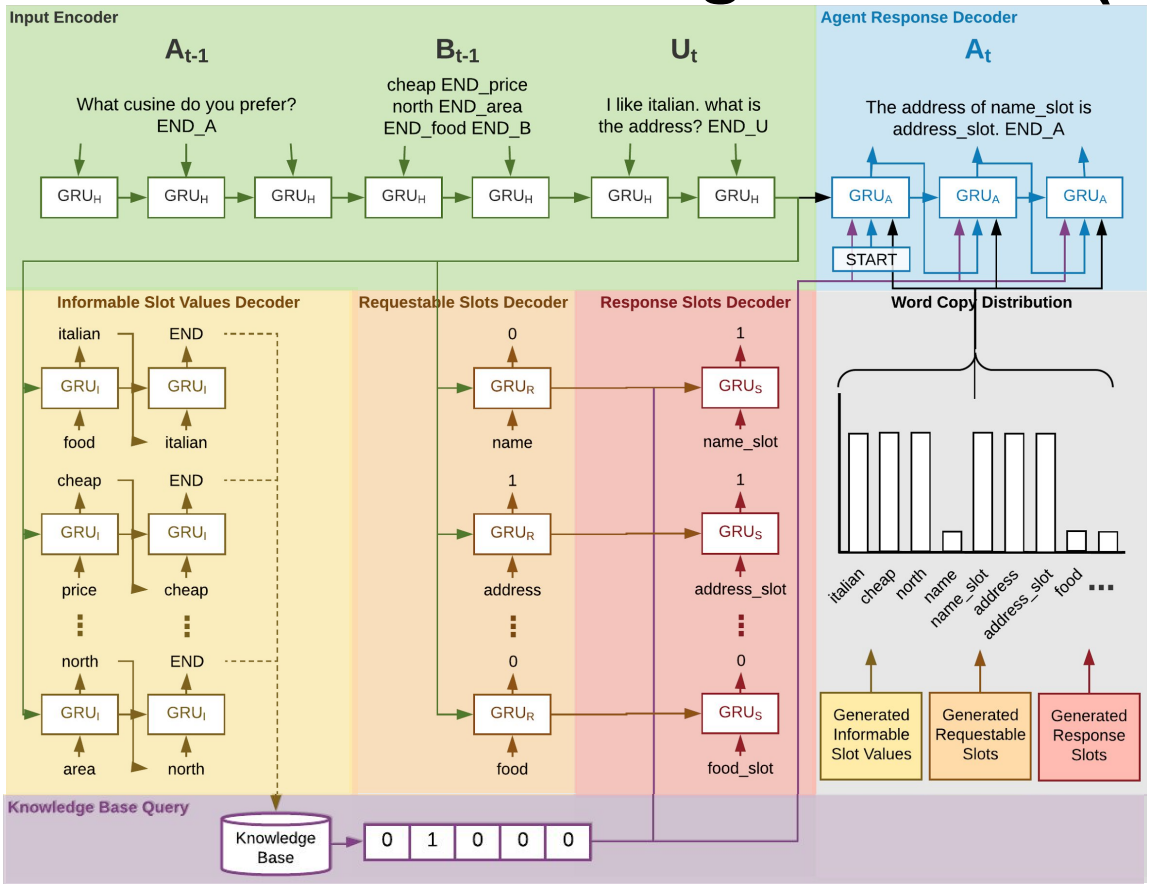
Word Copy Distribution

The chance of a word in generated informable slot values, requestable slots and response slots to appear in the agent response. Use together with copy-mechanism seq2seq (Gu et al 2016).



We call the whole end-to-end network as:

Flexible-Structured Dialogue Model (FSDM)



Overall of FSDM

five (5) components that work together in an end-to-end manner

(1) The encoder encodes the agent response, the belief state, and the current user utterance;

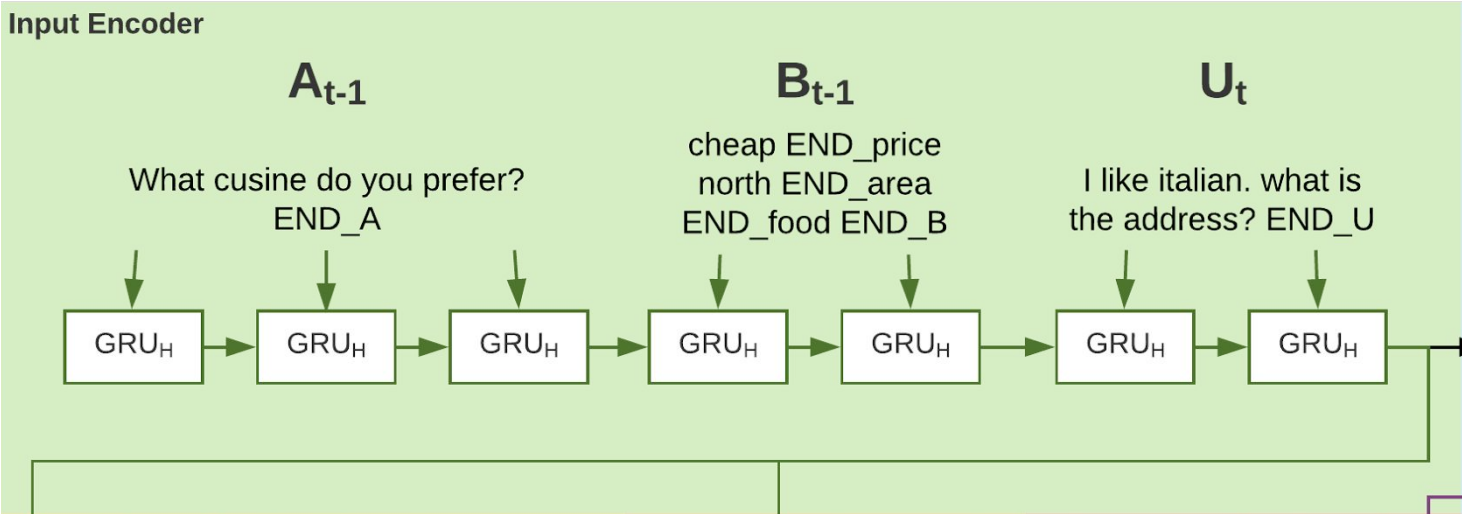
(2) The belief state tracker contains informable slot value decoder and requestable slot binary classifier; Both take the last hidden state of encoder as the initial state;

(3) Given generated informable slot values, the KB query component queries the KB and encodes the number of records returned in a one-hot vector ;

(4) The response slot binary classifier predicts what slots should appear in the agent response;

(5) The agent response decoder takes in the KB output, a word copy probability vector last hidden states of the input encoder as to generate a response.

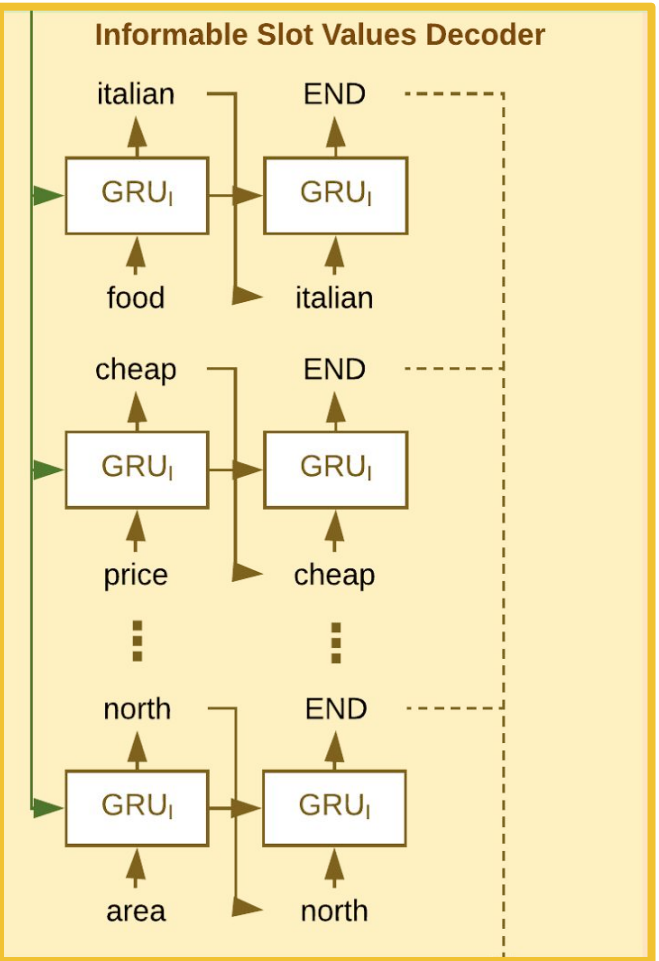
Encoder



Inputs: (1) the agent response A_{t-1} , (2) the belief state B_{t-1} from the (t-1)-th turn , (3) the current user utterance U_t .

Outputs: last hidden state of the encoder serves as the initial hidden state of the belief state tracker and the response decoder

Informable Slot Values Decoder

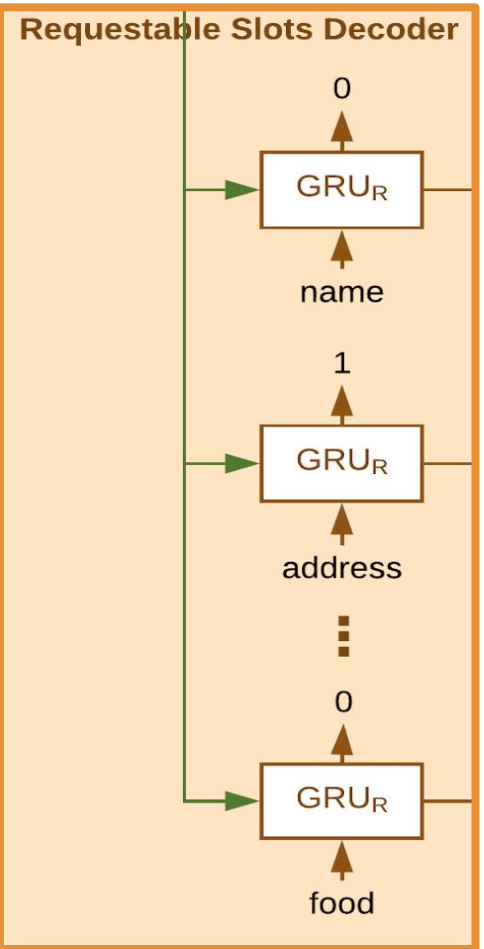


Inputs: (1) last hidden state of the encoder (2) unique start-of-sentence symbols for each slot, for example food slot's starting word is "food"

Outputs: For each slot, a sequence of words regarding this slot's value are generated. For example, the value generated for food slot is "italian END_food"

Intuition: The unique start-of-sentence symbols ensures slot and value alignment. The copy-mechanism seq2seq allows copying value directly from encoder input.

Requestable Slots Binary Classifier



Inputs: (1) last hidden state of the encoder (2) unique start-of-sentence symbols for each slot, for example food slot's starting word is "food".

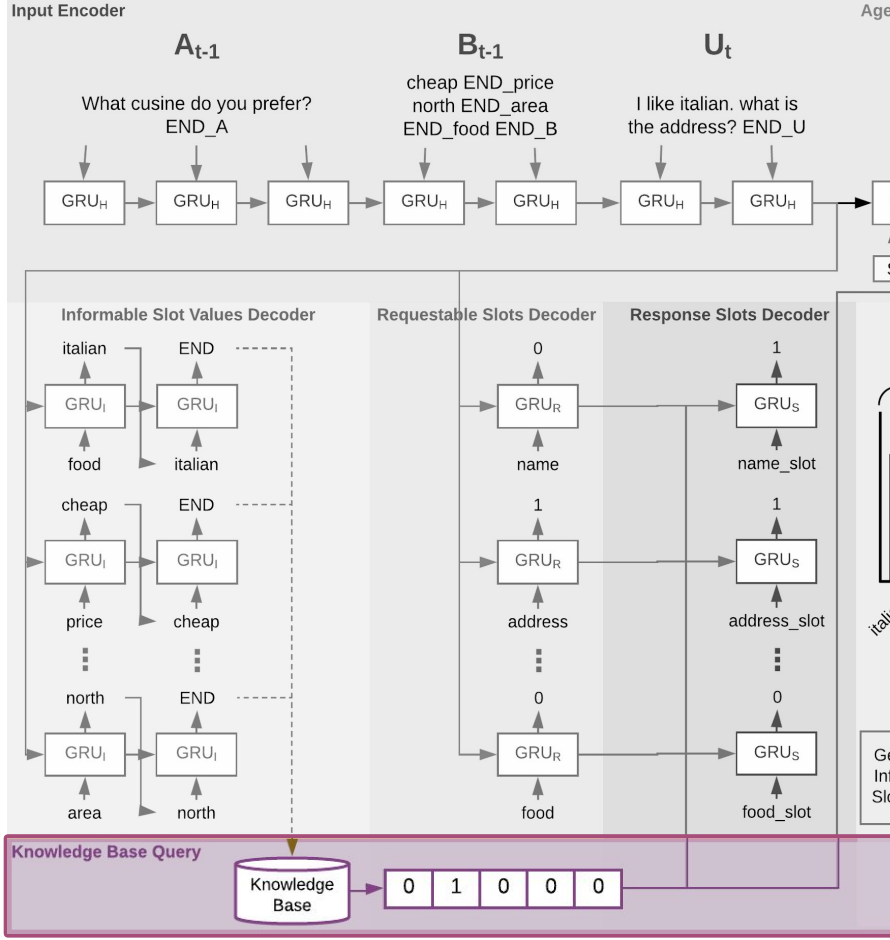
Outputs: For each slot, a binary prediction (1/0) is produced regarding whether this slot is requested by the user or not.

Note that the GRU here is only one-step. It may be replaced with any classification architecture. We choose GRU because we want to use the hidden state here as the initial state of response slot binary classifier.

Knowledge Base Query

Inputs: (1) generated informable slot values (2) Knowledge base

Outputs: a one-hot vector represents the number of records matched.

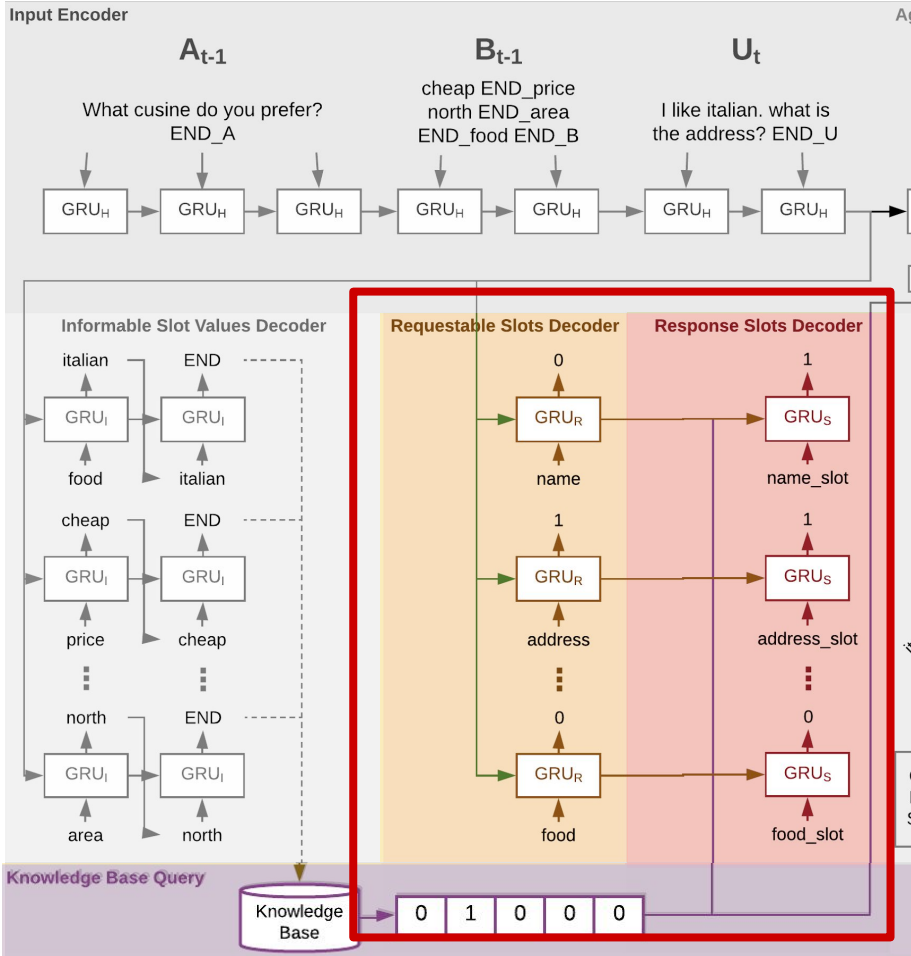


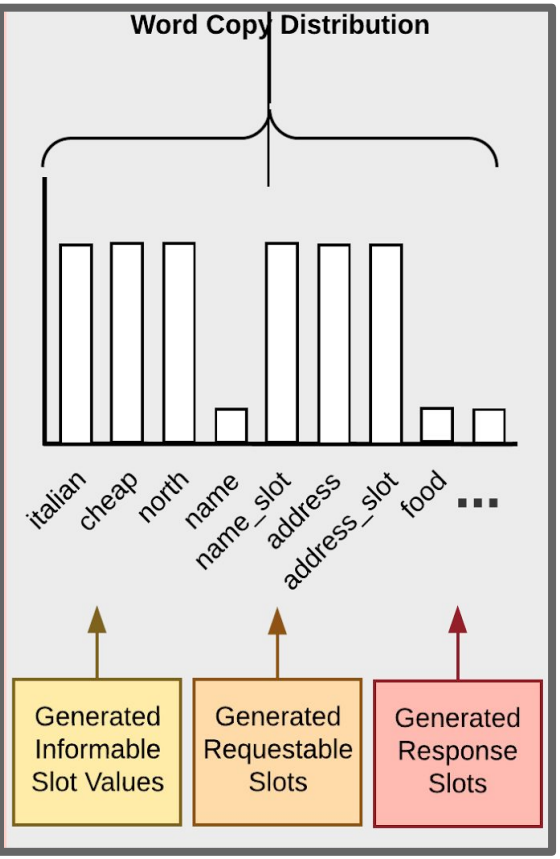
Response Slots Binary Classifier

Inputs: (1) KB queried result (2) hidden states of requestable slot binary classifier

Outputs: For each response slot, a binary prediction (1/0) is produced regarding whether this response slot appears in the agent response or not.

Motivation: incorporate all the relevant information about the retrieved entities into the response





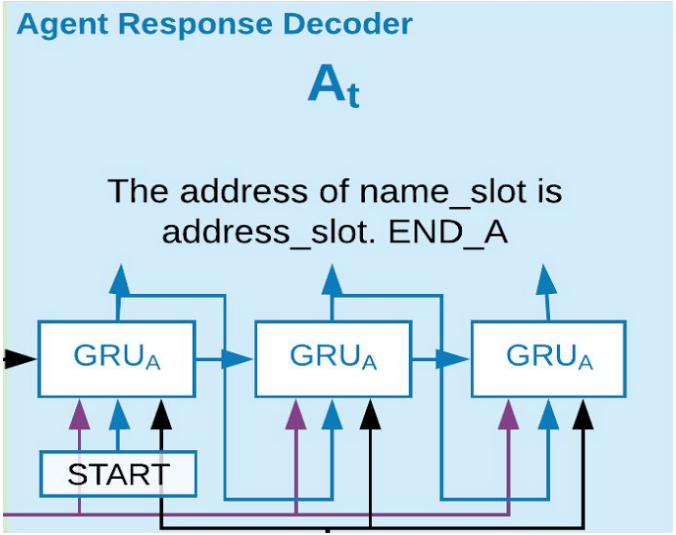
Motivation: The canonical copy mechanism only takes a sequence of word indexes as inputs but does not accept the multiple Bernoulli distributions we obtain from binary classifiers.

Inputs: prediction from (1) requestable slot binary classifier (2) response slot binary classifiers (3) informable slot value decoders

Outputs: if a word is a requestable slot or a response slot, the probability is equal to their binary classifier output; if a word appears in the generated informable slot values, its probability is equal to 1; for the other words in the vocabulary the probability is equal to 0.

$$\mathcal{P}^{\mathcal{C}}(w) = \begin{cases} y^{k^R}, & \text{if } w = k^R, \\ y^{k^S}, & \text{if } w = k^S, \\ 1, & \text{if } w \in I_t, \\ 0, & \text{otherwise,} \end{cases}$$

Decoder



Inputs: (1) last hidden state of encoder (2) Knowledge base queried result (3) Word copy distribution

Outputs: a de-lexicalized agent response

Final Loss

$$\mathcal{L} = \alpha^I \mathcal{L}^I + \alpha^R \mathcal{L}^R + \alpha^S \mathcal{L}^S + \alpha^A \mathcal{L}^A$$

Informable slot values

Requestable slots

Response slots

Agent response

Experiment Setting

Dataset:
Cambridge Restaurant dataset (CamRest) (Wen et al 2016)
Stanford in-car assistant dataset (KVRET) (eric et al 2017)

Evaluation Metrics:
For belief state tracking, **precision**, **recall**, and **F_1 score** of informable slot values and requestable slots.

For task completion evaluation, the **Entity Match Rate (EMR)** and **Success F_1 score (SuccF_1)** are reported.

BLEU is applied to the generated agent responses for evaluating language quality.

Experiment Setting-Baselines

NDM (Wen et al 2016) proposes a modular end-to-end trainable network. It applies de-lexicalization on user utterances and responses.

LIDM (Wen et al 2017) improves over NDM by employing a discrete latent variable to learn underlying dialogue acts. This allows the system to be refined by reinforcement learning.

KVRN (Eric et al 2017) adopts a copy-augmented Seq2Seq model for agent response generation and uses an attention mechanism on the KB. It does not perform belief state tracking.

TSCP/RL (Lei et al 2018) is a two-stage CopyNet which consists of one encoder and two copy-mechanism-augmented decoders for belief state and response generation. **TSCP** includes further parameter tuning with reinforcement learning to increase the appearance of response slots in the generated response.

Turn-Level Belief State Tracking Result

Dataset	CamRest						KVRET					
Method	Inf P	Inf R	Inf F_1	Req P	Req R	Req F_1	Inf P	Inf R	Inf F_1	Req P	Req R	Req F_1
TSCP/RL†	0.970	0.971	0.971	0.983	0.935	0.959	**0.936**	0.874	0.904	0.725	0.485	0.581
TSCP†	0.970	0.971	0.971	0.983	0.938	0.960	0.934	0.890	0.912	0.701	0.435	0.526
FSDM/Res	0.979	0.984	0.978	0.994	0.947	0.967	0.918	0.930	0.925	0.812	0.993	0.893
FSDM	**0.983***	**0.986***	**0.984***	**0.997***	**0.952**	**0.974***	0.92	**0.935***	**0.927***	**0.819***	**1.000***	**0.900***

Dialogue-Level Task Completion Result

Dataset	CamRest			KVRET		
Method	BLEU	EMR	SuccF_1	BLEU	EMR	SuccF_1
NDM	0.212	0.904	0.832	0.186	0.724	0.741
LIDM	0.246	0.912	0.840	0.173	0.721	0.762
KVRN	0.134	-	-	0.184	0.459	0.540
TSCP	0.253	0.927	0.854	**0.219**	0.845	0.811
TSCP/RL †	0.237	0.915	0.826	0.195	0.809	0.814
TSCP†	0.237	0.913	0.841	0.189	0.833	0.81
FSDM/St	0.245	-	0.847	0.204	-	0.809
FSDM/Res	0.251	0.924	0.855	0.209	0.834	0.815
FSDM	**0.258***	**0.935***	**0.862***	0.215	**0.848***	**0.821***

Example of generated belief state and response

(calendar scheduling domain)

user msg	what is the date and time of my next meeting and who will be attending it ?
	belief state
GOLD	informable slot (event=meeting), requestable slot (date, time, party)
TSCP	'meeting' '⟨EOS_Z1⟩' 'date' ';' 'party'
FSDM	event=meeting date=True time=True party = True
	agent response
GOLD	your next meeting is with party_SLOT on the date_SLOT at time_SLOT.
TSCP	your next meeting is at time_SLOT on date_SLOT at time_SLOT .
FSDM	you have a meeting on date_SLOT at time_SLOT with party_SLOT

Conclusions

Propose a novel end-to-end architecture flexibly-structured dialogue model for task-oriented dialogue.

It uses the structure in the schema of the KB to make architectural choices that introduce inductive bias and address the limitations of fully structured and free-form methods.

The experiment suggests that this architecture is competitive with SOTS models

Code (coming Sep 2019)**:**

https://github.com/uber-research/FSDM

Special Thanks to

Dr. Alexandros Papangelis
Janice Lan
Uber AI Labs
SIGDIAL Reviewers

Reference

Bing Liu and Ian Lane. 2018. End-to-end learning of task-oriented dialogs. In Proceedings of the NAACL-HLT.

Bing Liu, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck. 2018. Dialogue learning with human teaching and feedback in end-to-end train-able task-oriented dialogue systems. In NAACL.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In ACL.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In ACL (1). The Association for Computer Linguistics.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkˇsi´c, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017b. A network-based end-to-end trainable task-oriented dialogue system. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, volume 1, pages 438–449. ACL.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve J. Young. 2017a. Latent intention dialogue models. In ICML, volume 70 of Proceedings of Machine Learning Research, pages 3732–3741. PMLR.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In SIGDIAL Conference, pages 37–49. Association for Computational Linguistics.