ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS (ICLR 2020)

Bridging the Gap between Training and Inference for Neural Machine Translation (ACL 2019)

Multi-Domain Dialogue Acts and Response Co-Generation (ACL 2020)

ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS

Two problems of current pre-training method

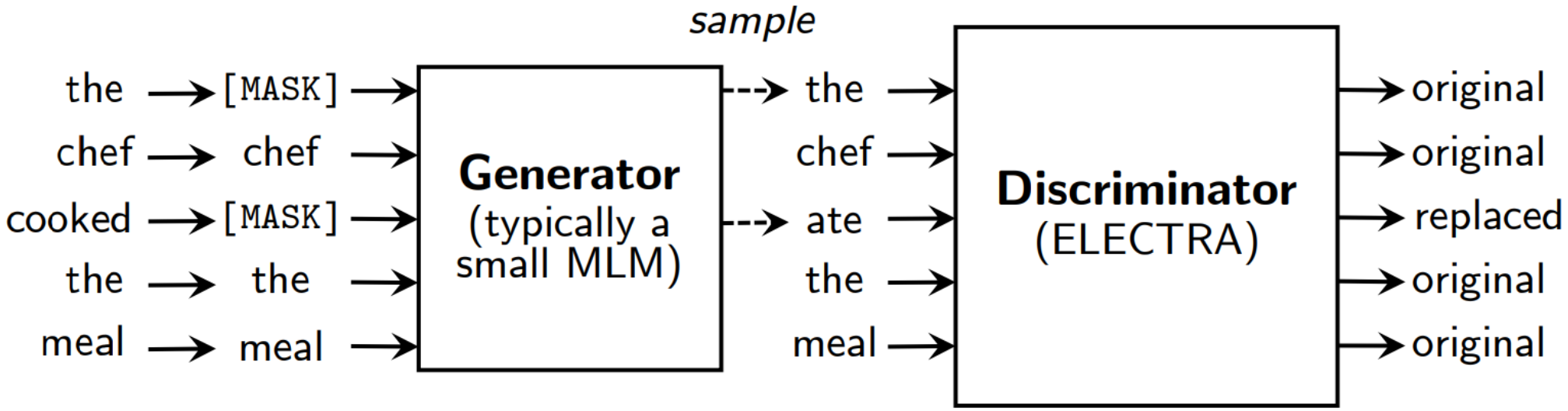
- Discrepancy of exposing the model to [MASK] tokens during pre-training but not fine-tuning.

- Only learn from a small subset of the unlabeled data (typically 15%).

- Method

Generator: MLM BERT

Discriminator: distinguish tokens in the data from tokens that have been replaced by generator samples



- Loss fuction

$$\mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i | \boldsymbol{x}^{\text{masked}})\right)$$

$$\mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D) = \mathbb{E}\left(\sum_{t=1}^{n} -\mathbb{1}(x_t^{\text{corrupt}} = x_t) \log D(\boldsymbol{x}^{\text{corrupt}}, t) - \mathbb{1}(x_t^{\text{corrupt}} \neq x_t) \log(1 - D(\boldsymbol{x}^{\text{corrupt}}, t))\right)$$

$$\min_{\theta_G, \theta_D} \sum_{\boldsymbol{x} \in \mathcal{X}} \mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) + \lambda \mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D)$$

After pre-training, throw out the generator and only fine-tune the discriminator (the ELECTRA model) on downstream tasks.

- Experiment Results

Model	Train / Infer FLOPs	Speedup	Params	Train Time + Hardware	GLUE
ELMo	3.3e18 / 2.6e10	19x / 1.2x	96M	14d on 3 GTX 1080 GPUs	71.2
GPT	4.0e19 / 3.0e10	1.6x / 0.97x	117M	25d on 8 P6000 GPUs	78.8
BERT-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	75.1
BERT-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	82.2
ELECTRA-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	79.9
50% trained	7.1e17 / 3.7e9	90x / 8x	14M	2d on 1 V100 GPU	79.0
25% trained	3.6e17 / 3.7e9	181x / 8x	14M	1d on 1 V100 GPU	77.7
12.5% trained	1.8e17 / 3.7e9	361x / 8x	14M	12h on 1 V100 GPU	76.0
6.25% trained	8.9e16 / 3.7e9	722x / 8x	14M	6h on 1 V100 GPU	74.1
ELECTRA-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	85.1

Table 1: Comparison of small models on the GLUE dev set. BERT-Small/Base are our implemen-

ELECTRA performs the best under the same computation.

- **ELECTRA 15%:** predict 15% of the tokens that were masked out of the input.
- **Replace MLM:** replacing masked-out tokens with tokens from a generator model.
- **All-Tokens MLM:** masked tokens are replaced with generator samples. Furthermore, the model predicts the identity of all tokens in the input.

Model	ELECTRA	All-Tokens MLM	Replace MLM	ELECTRA 15%	BERT
GLUE score	85.0	84.3	82.4	82.4	82.2

Table 5: Compute-efficiency experiments (see text for details).

Model	ELECTRA	All-Tokens MLM	Replace MLM	ELECTRA 15%	BERT
GLUE score	85.0	84.3	82.4	82.4	82.2

Table 5: Compute-efficiency experiments (see text for details).

Some observations can be found from the table:

1. Both replacing tokens task and training all tokens are important.

2. All-Tokens MLM VS. BERT Replace MLM VS. BERT

   Compared to replacing tokens, predicting all tokens contributes more.

3. Replace MLM VS. BERT

   Discrepancy of exposing the model to [MASK] tokens during pre-training but not fine-tuning can not be ignored.

Noise-Contrastive Estimation (NCE)

- A problem: how to reduce the computation of normalization factor Z in softmax layer (eg. language model)

$$p(w_i|w_0^{t-1}) = \frac{e^{-y(w_i|w_0^{t-1})}}{\sum_{j=1}^{N_v} e^{-y(w_j|w_0^{t-1})}}, Z = \sum_{j=1}^{N_v} e^{-y(w_j|w_0^{t-1})}$$

$$p(\boldsymbol{x}) = \frac{e^{G(\boldsymbol{x};\boldsymbol{\theta})}}{Z(\boldsymbol{\theta})}$$

NCE models the normalization factor as a parameter and converts the multiple classification to binary classification problem

$$p(1|\boldsymbol{x}) = \sigma\Big(G(\boldsymbol{x};\boldsymbol{\theta}) - \gamma\Big) = \frac{1}{1 + e^{-G(\boldsymbol{x};\boldsymbol{\theta})+\gamma}}$$

$$\arg\min_{\boldsymbol{\theta},\gamma} - \mathbb{E}_{\boldsymbol{x}\sim\tilde{p}(\boldsymbol{x})} \log p(1|\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim U(\boldsymbol{x})} \log p(0|\boldsymbol{x})$$

Here, $\tilde{p}(\boldsymbol{x})$ is the positive sample distribution and $U(\boldsymbol{x})$ is the negative sample distribution

- How to sample the negative samples?

Word2Vec: sampling by word frequency

Is the high frequency word leads to misclassification?

Not necessarily, high frequency **word** doesn't mean high frequency **n-gram**

ELECTRA: sampling by the model output probability (argmax), if the model predicts a wrong word with a high probability, it is the reason of failure

The sampling method depends on your task.

- Conclusion

1. Sufficiently exploiting the data is important.
2. Making the training and testing process consistent.
3. Try negative sampling if suitable.

Bridging the Gap between Training and Inference for Neural Machine Translation

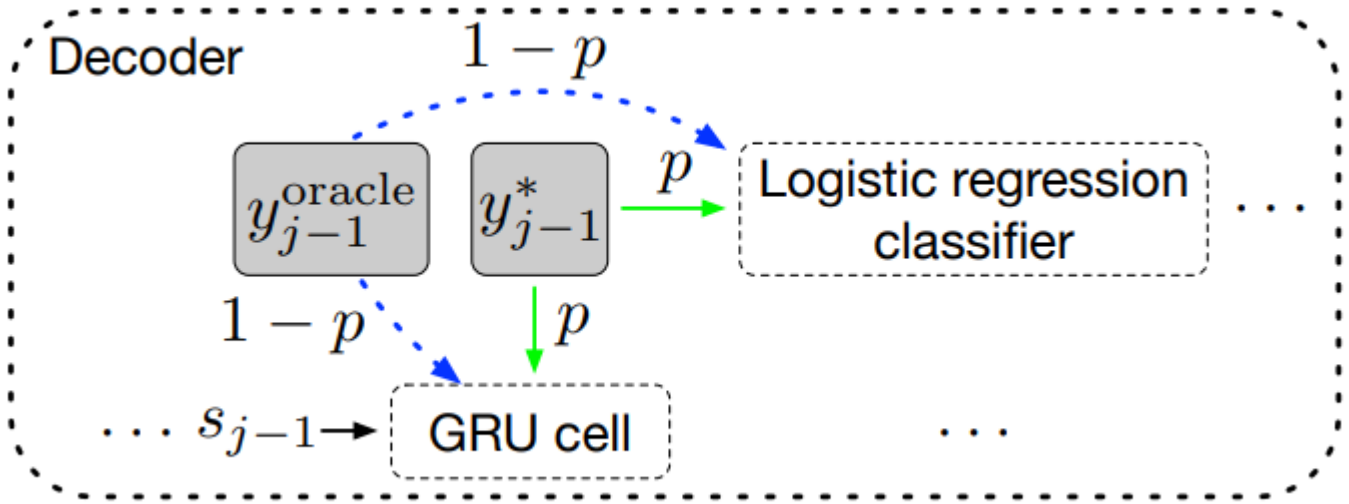
Existing problems in NMT

- discrepancy of the fed context in training (groundtruth) and inference (generated) leads to error accumulation (exposure bias)
- word-level training requires strict matching between the generated sequence and the ground truth sequence which leads to overcorrection over different but reasonable translations.

- Approach

1. sample from the groundtruth word with a probability of p or from the oracle word with a probability of 1−p

2. feed context either the ground truth word or the oracle word.



Oracle Word Selection

- Word-Level Oracle

Using Gumbel-Max technique to sample from the word distribution

- Sentence-Level Oracle

Using beam search to select k-best candidates and compute it’s BLEU score compared to groundtruth, and selecting the top first as oracle sentence

Force decoding trick: to make sure the oracle sentence has the same length with groundtruth

If the candidate translation gets a word distribution P_j at the j-th step where j is not the end and EOS is the top first word, then we select the top second word as the j-th word of this candidate translation

If the candidate translation gets a word distribution at the final step where EOS is not the top first word, then we select EOS as the end word of this candidate translation.

Experiments

Systems	Architecture	MT03	MT04	MT05	MT06	Average
Existing end-to-end NMT systems						
Tu et al. (2016)	Coverage	33.69	38.05	35.01	34.83	35.40
Shen et al. (2016)	MRT	37.41	39.87	37.45	36.80	37.88
Zhang et al. (2017)	Distortion	37.93	40.40	36.81	35.77	37.73
Our end-to-end NMT systems						
this work	RNNsearch	37.93	40.53	36.65	35.80	37.73
	+ SS-NMT	38.82	41.68	37.28	37.98	38.94
	+ MIXER	38.70	40.81	37.59	38.38	38.87
	+ OR-NMT	**40.40**‡†*	**42.63**‡†*	**38.87**‡†*	**38.44**‡	**40.09**
	Transformer	46.89	47.88	47.40	46.66	47.21
	+ word oracle	47.42	48.34	47.89	47.34	47.75
	+ sentence oracle	**48.31***	**49.40***	**48.72***	**48.45***	**48.72**

Systems	Average
RNNsearch	37.73
+ word oracle	38.94
+ noise	39.50
+ sentence oracle	39.56
+ noise	**40.09**

Table 2: Factor analysis on Zh→En translation, the results are average BLEU scores on MT03∼06 datasets.

Sentence oracle improves the performance most

Missing one important baseline

Paulus, R., Xiong, C., & Socher, R. (2017). A deep reinforced model for abstractive summarization.

$$L_{rl} = (r(\hat{y}) - r(y^s)) \sum_{t=1}^{n'} \log p(y_t^s | y_1^s, \ldots, y_{t-1}^s, x)$$

- One question for everyone

Is noise-contrastive estimation (NCE) suitable for NMT?

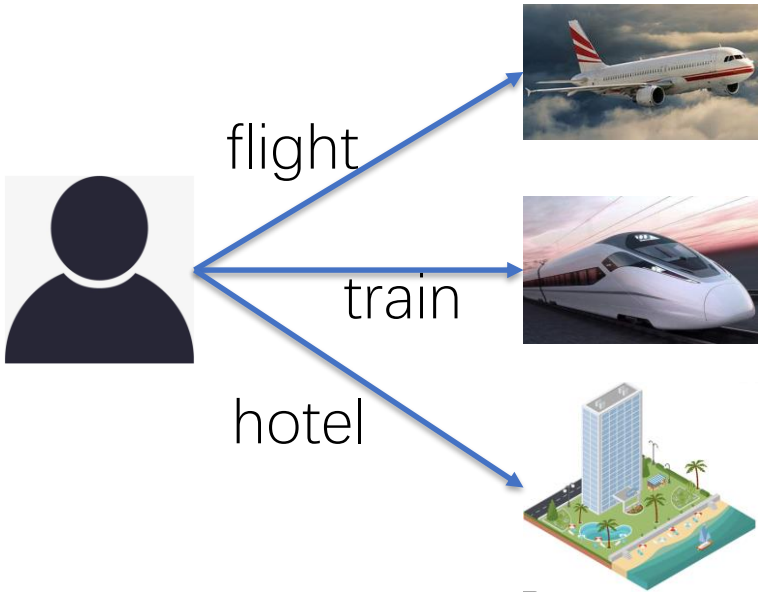
Multi-Domain Dialogue Acts and Response Co-Generation

Multi-domain Task-oriented Dialogue System

- **Task-oriented Dialogue System**

Facilitate customer services through natural language conversations eg., *hotel reservation*, *ticket booking*

Multi-domain dialogue contains multiple domains in single session

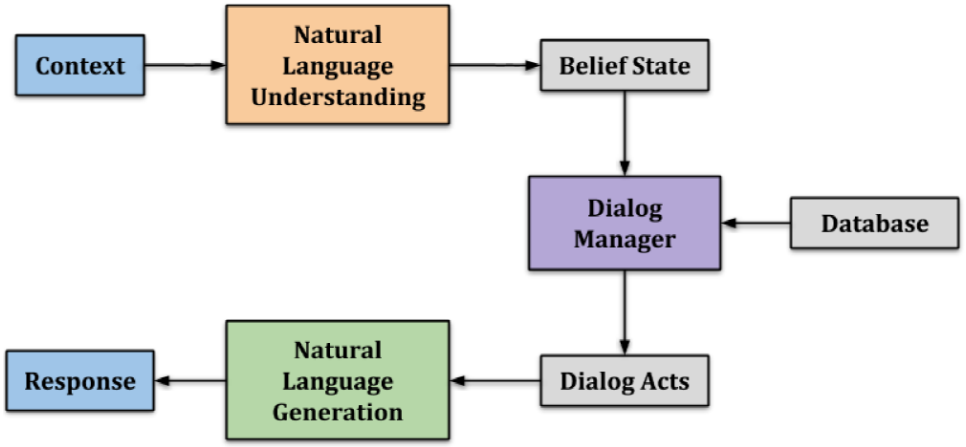


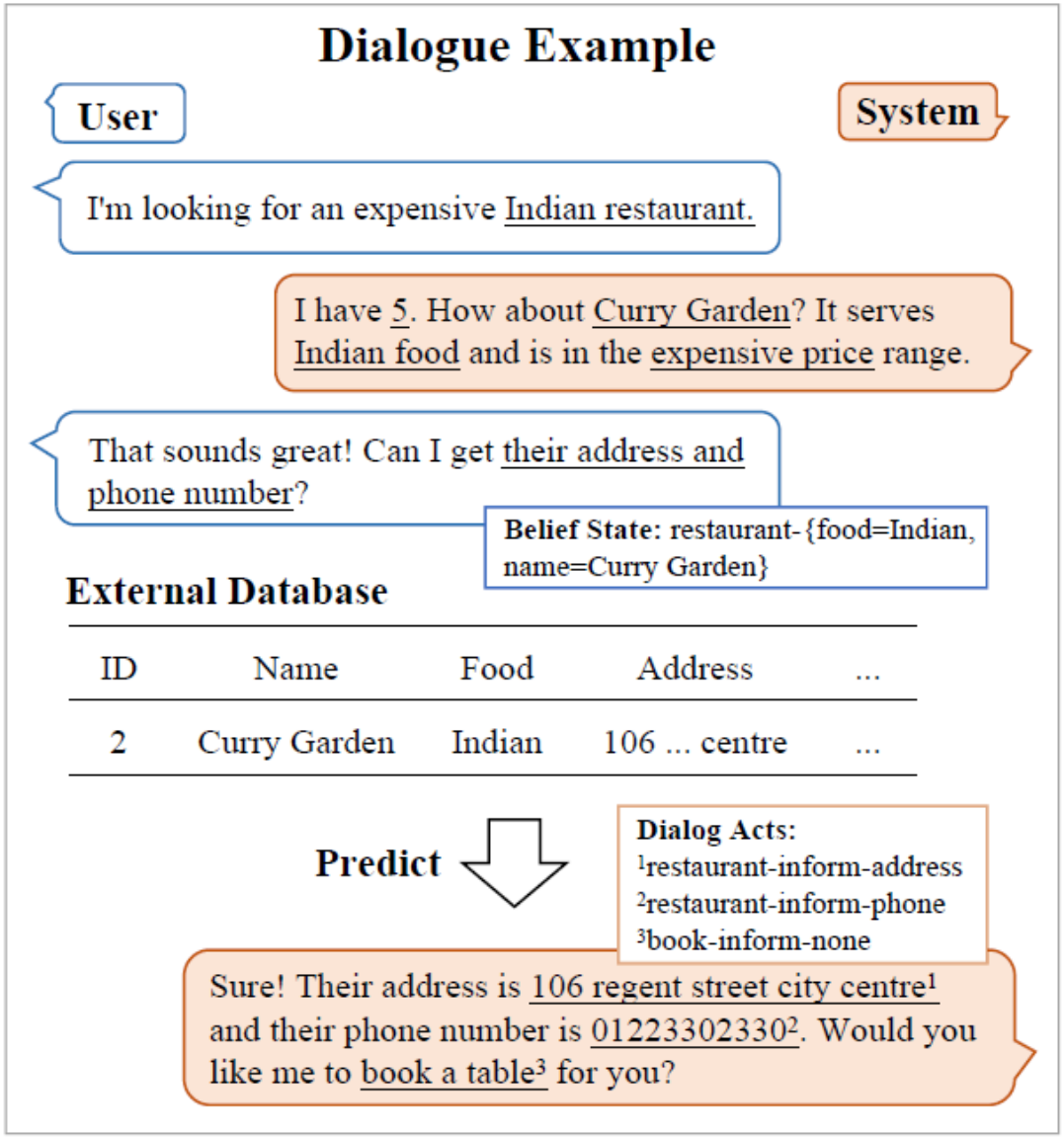
Multi-domain Task-oriented Dialogue System

- Architecture

Dialogue state tracking (DST)
Natural language generation (NLG)

DST -> predict user belief state
NLG -> dialogue act prediction and response generation





An example of multi-domain task-oriented dialogue system

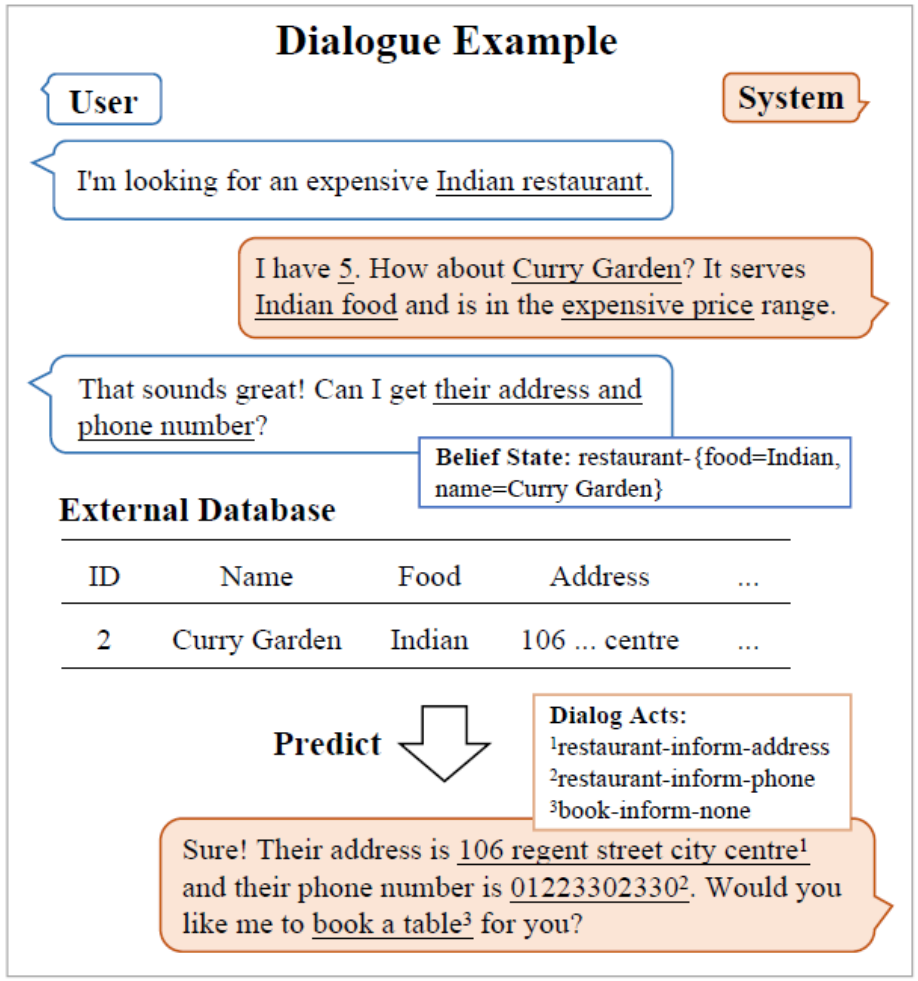
Hierarchical Dialogue Acts

➢ Dialogue Act

- Reflect what the system should do next
- Different response subsequences corresponds to different acts

➢ Hierarchical Structure

- Comprise of domains, actions, slots
- Multiple dialogue acts are involved in single turn



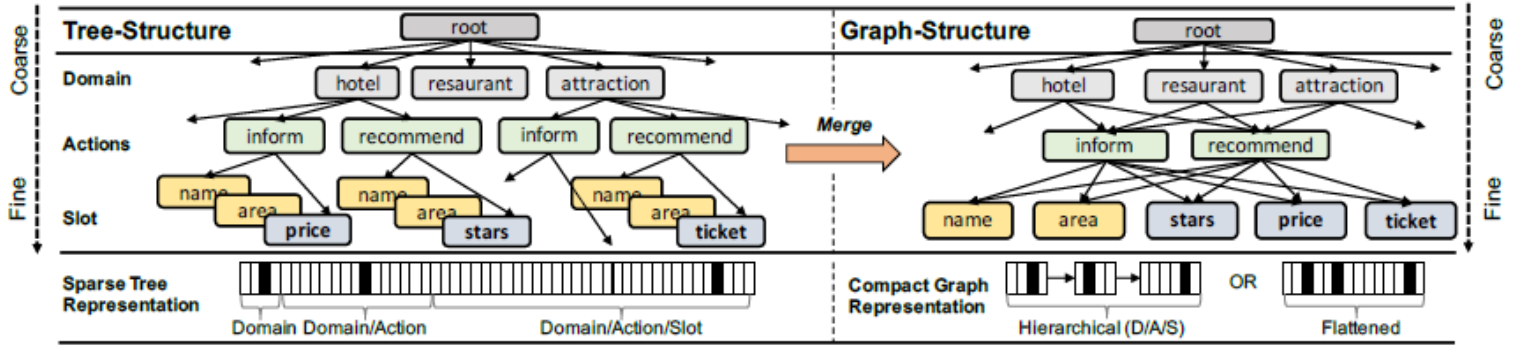
Dialogue Acts Representation

- One-hot vector

Each dimension is a triple (Wen et al., 2015)

Each dimension is an act item (Chen et al., 2019)

Inner and outer relationships between acts are ignored, response and acts have no connections!



Our contributions

- We model dialogue act prediction as a sequence generation problem to better incorporate their in-between semantic structures, and demonstrate that this approach can conduce to better act prediction and response generation.

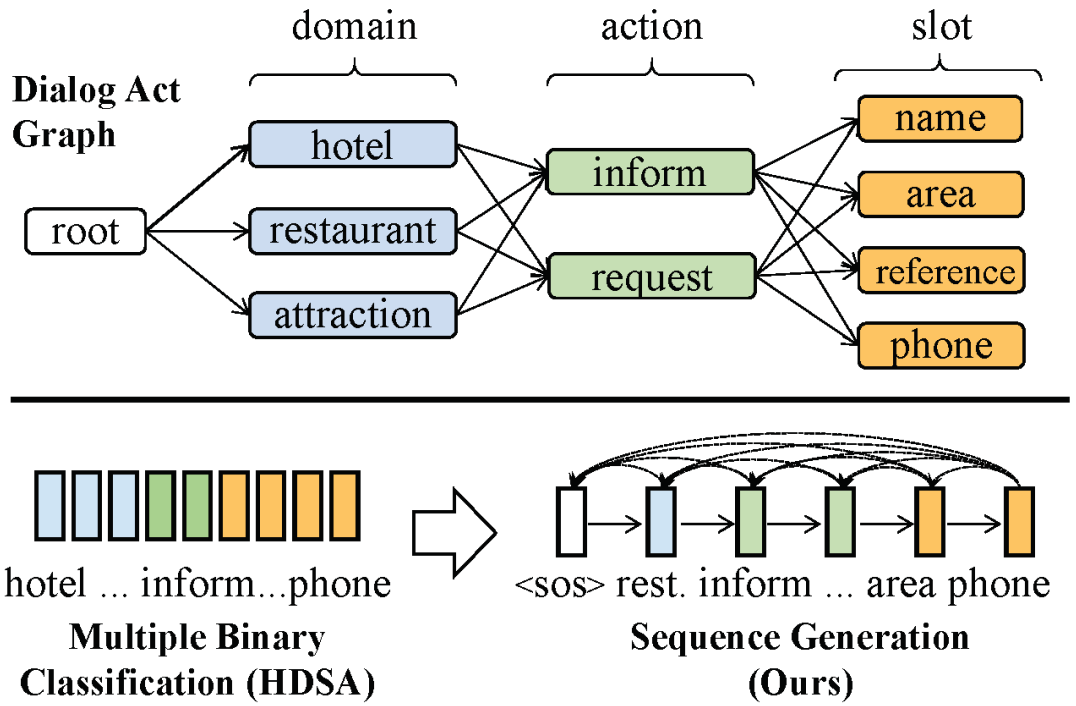
- We propose a neural co-generation model to generate act and response sequences concurrently, and introduce the uncertainty loss to learn adaptive weights with stable and superior performance.

- Experiments on the MultiWOZ dataset prove that our model outperforms the state-of-the-art methods in both automatic and human evaluations.

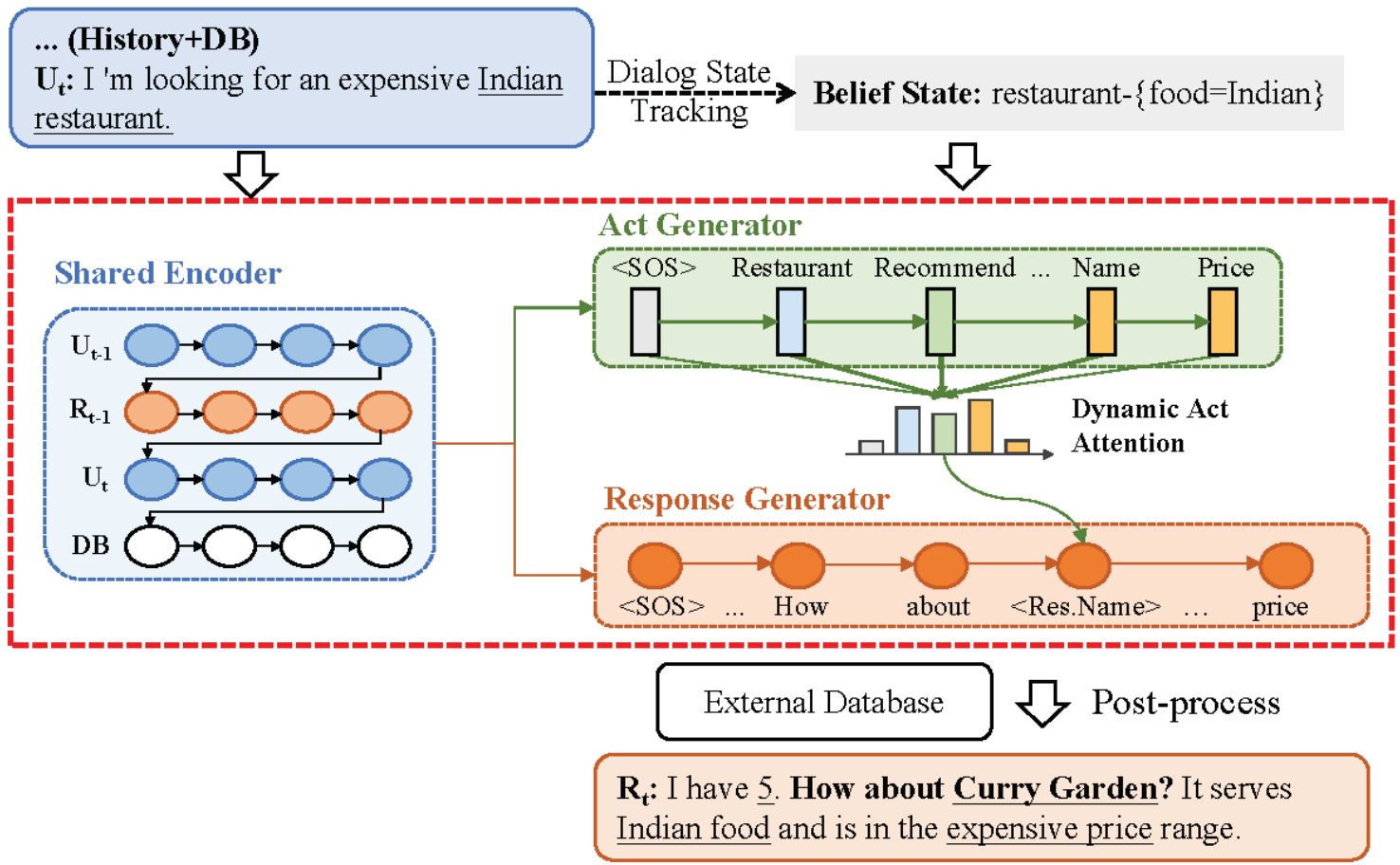
Dialogue Acts Generation

One-hot act -> Sequential act
Classification -> Generation

Act sequence establishes the relationships between acts



Acts and Response Co-Generation



Acts and Response Co-Generation

- **Shared Encoder**

Act generator and response generator share same encoder and input

- **Dynamic Act Attention**

The response generator can dynamically capture salient acts by attending to different generated acts

- **Joint Learning**

Joint learning improves each task

Joint Learning Optimization Method

Dialogue acts and responses vary seriously in sequence length and dictionary size

	Avg Sequence Length	Vocabulary Size
Response	17	3130
Dialogue Act	5	44

Traditional Loss:

$$\mathcal{L}(\theta) = \alpha \mathcal{L}_a(\theta) + (1 - \alpha)\mathcal{L}_r(\theta).$$

Two losses have different scales and the training is unstable!

Our Optimization Method

We adopt uncertainty loss to optimize the model

- Uncertainty loss (Kendall et al., 2018)

Use homoscedastic task uncertainty to adaptively learn task dependent weights

$$\mathcal{L}(\theta, \sigma_1, \sigma_2) = \frac{1}{2\sigma_1^2}\mathcal{L}_a(\theta) + \frac{1}{2\sigma_2^2}\mathcal{L}_r(\theta) + \log \sigma_1^2 \sigma_2^2$$

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics

Dataset

- MultiWOZ Dataset Statistics

Dialogs	Total Turns	Unique Tokens	Value
8538	115,424	24,071	4510
Dialog Acts	**Domain**	**Actions**	**Slots**
625	10	7	27

- Evaluation Metrics

Inform Rate: whether a system has provided an appropriate entity.

Request Success: whether a system has answered all requested attributes.

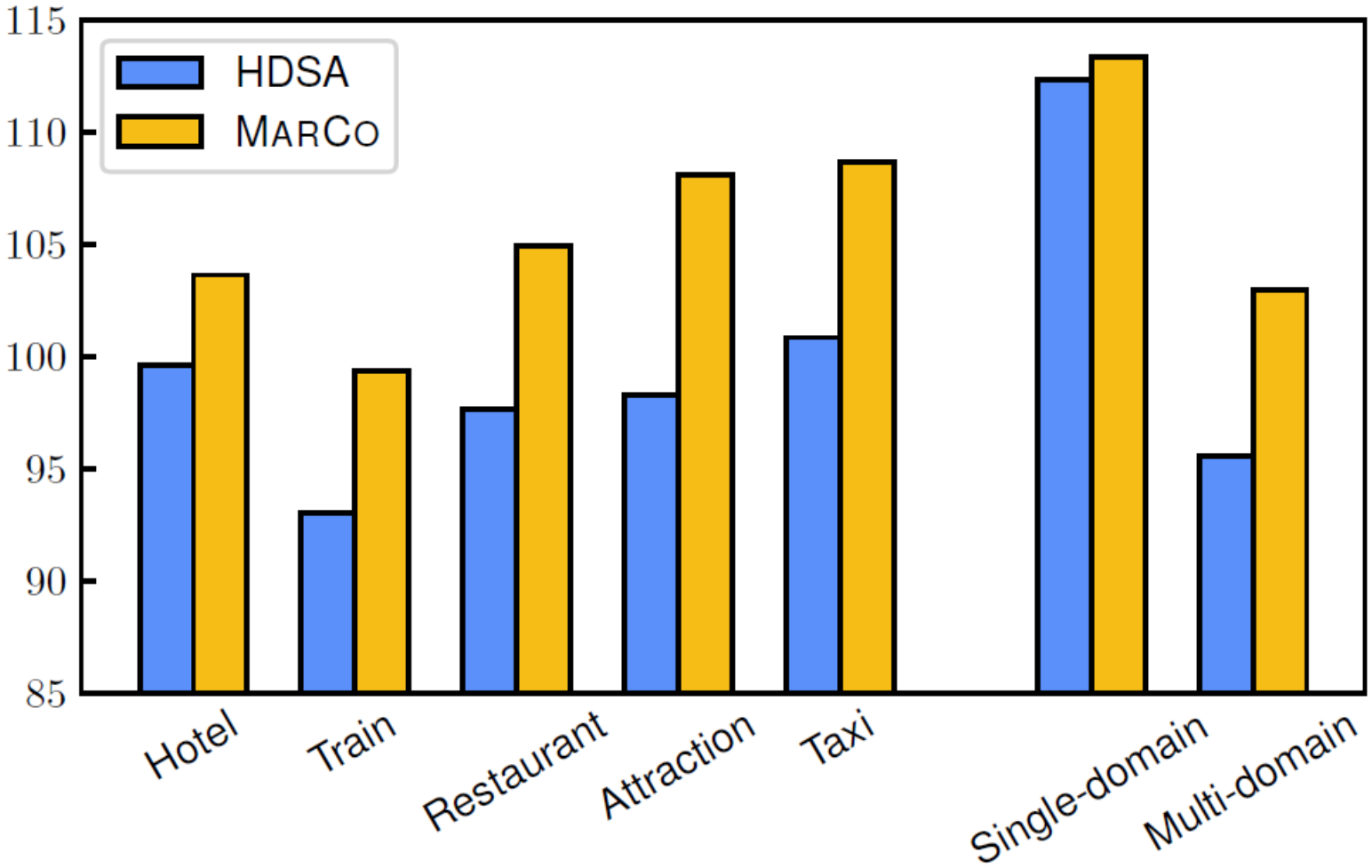
BLEU: overlap rate with ground-truth.

Combined score: (Inform Rate+Request Success)*0.5+BLEU

Overall Performance

Dialog Act	Model	Inform	Success	BLEU	Combined Score
Without Act	LSTM	71.29	60.96	18.80	84.93
	Transformer	71.10	59.90	19.10	84.60
	TokenMoE	75.30	59.70	16.81	84.31
	Structured Fusion	82.70	72.10	16.34	93.74
One-hot Act	SC-LSTM	74.50	62.50	20.50	89.00
	HDSA (MARCO)	76.50	62.30	**21.85**	91.25
	HDSA	82.90	68.90	**23.60**	99.50
Sequential Act	MARCO	**90.30**	**75.20**	19.45	**102.20**
	MARCO (BERT)	**92.30**	**78.60**	20.02	**105.47**

Performance Across Domains



Single-domain (32.63%): 8.93 turns
Multi-domain (67.37%): 15.39 turns

Further Analysis

Three questions:

- How is the performance of act generator comparing with existing classification methods?
- Can our joint model build semantic associations between acts and responses?
- How does the uncertainty loss contribute to our co-generation model?

Dialogue Act Prediction

Method	F1
BiLSTM	71.4
Word-CNN	71.5
Transformer	73.1
Transformer (GEN)	73.2
MARCO	**73.9**

Table 2: Results of different act generation methods, where BiLSTM, Word-CNN and Transformer are baselines from (Chen et al., 2019). MARCO is our act generator trained jointly with the response generator and Transformer (GEN) is that without joint training.

Our joint act generator achieves the best performance

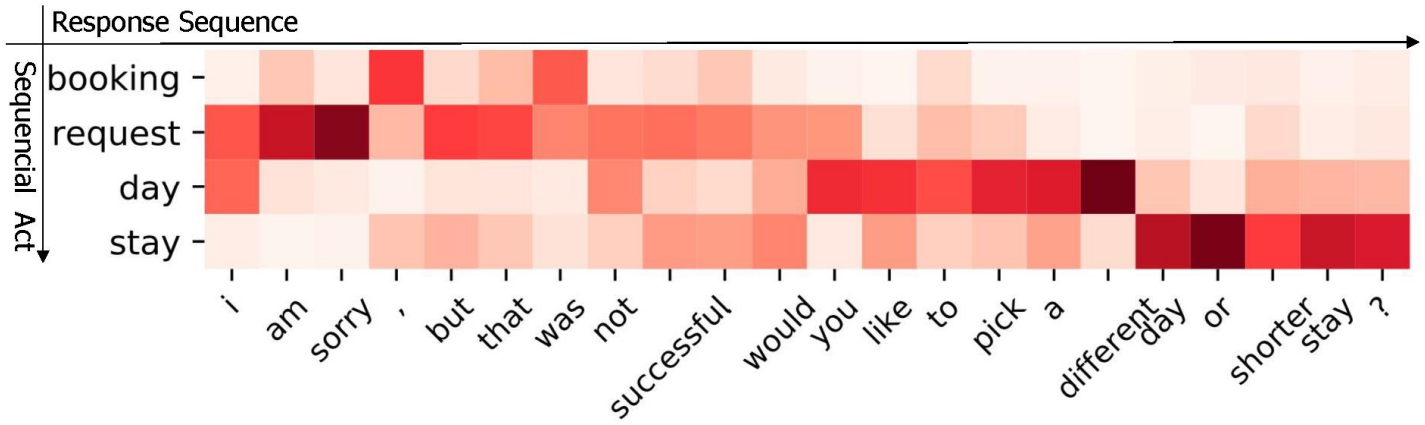
Joint vs. Pipeline

Model	Inform	Succ	BLEU	Combined
HDSA	92.9	68.9	23.60	99.50
Pipeline_1	84.3	54.4	16.00	85.35
Pipeline_2	86.6	66.0	18.31	94.61
Joint	**90.3**	**75.2**	**19.45**	**102.20**

Table 3: Results of response generation by joint and pipeline models, where Pipeline_1 and Pipeline_2 represent two pipeline approaches using or without using dynamic act attention. The performance of HDSA, which is the best pipeline model, is provided for comparison.

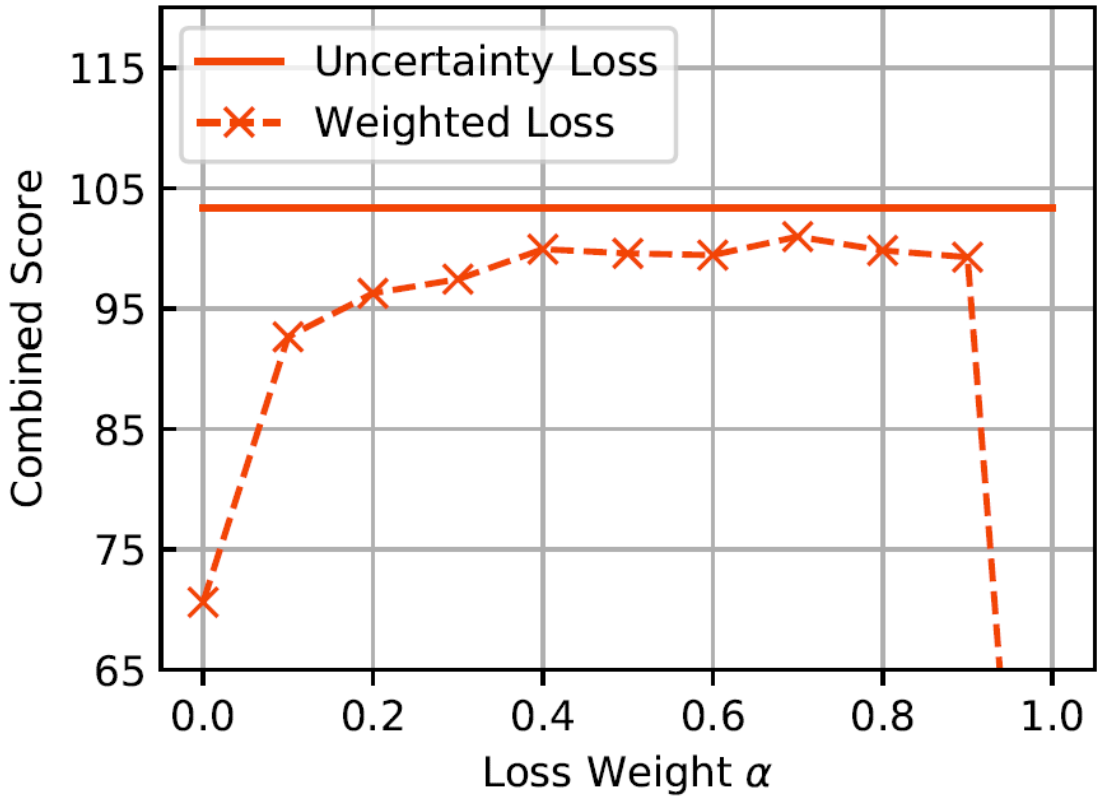
Dynamic Act Attention

An example



The response generator can attend to the local information such as “day” and “stay” as needed when generating a response asking about picking a different day or shorter stay.

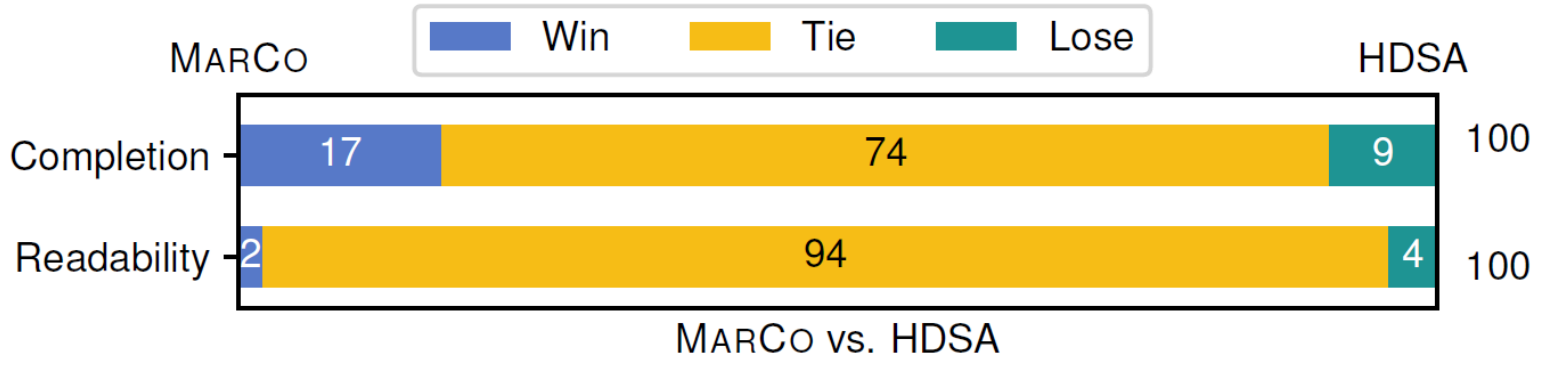
Uncertainty Loss



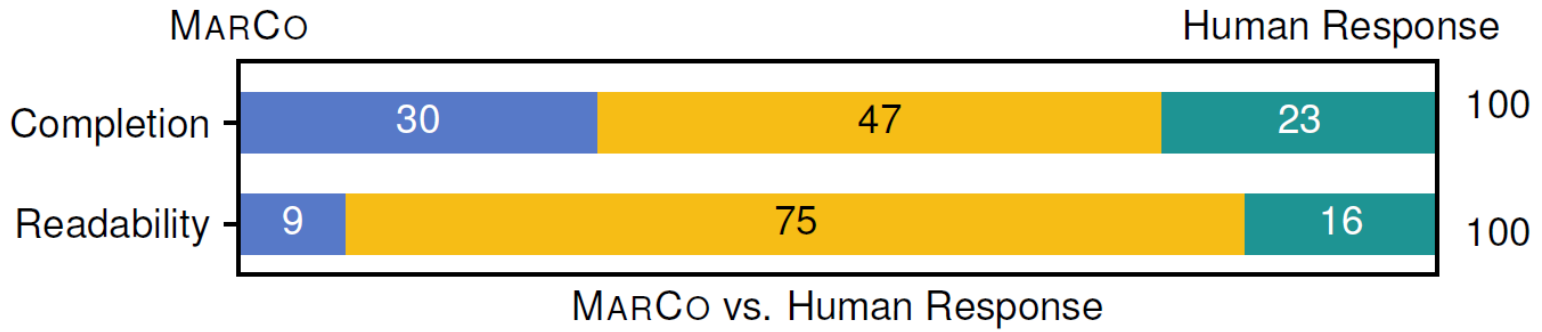
The uncertainty loss can learn adaptive weights with consistently superior performance

Human Evaluation

- Completion: if the response correctly answers a user query, including relevance and informativeness.
- Readability: reflects how fluent, natural and consistent the response is.



MARCO vs. HDSA



MARCO vs. Human Response

Case Study

Our model tends to provides more relevant information, and finish the goals in shorter turns

	Example 1
User:	I'm trying to plan a trip there but need a cheap place to stay.
	. . .
System:	Finches Bed and Breakfast is cheap.
User:	Okay. Sounds good. Can you tell me the type of hotel and provide me with the hotel's phone number?
	Results
Reference:	Finches is a guesthouse, and their number is 012***53. Can I help with anything else today?
HDSA:	The phone number for Finches is 012***53.
MARCO:	Finches is a guesthouse, and their number is 012***53.
	Example 2
User:	I am looking for a train that leaves on Thursday going to Cambridge.
	. . .
System:	TR1149 departs London Kings Cross at 17:17 and arrives in Cambridge at 18:08. Will that work?
User:	Yes it will. Can you book that for just me and provide the reference number?
	Results
Reference:	Sure thing! How many tickets do you need?
HDSA:	How many tickets would you like me to book for you?
MARCO:	I was able to book one ticket for you. Your reference number is R57G4DU4.

Thanks