Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems

Chien-Sheng Wu , Andrea Madotto , Ehsan Hosseini-Asl , Caiming Xiong , Richard Socher and Pascale Fung

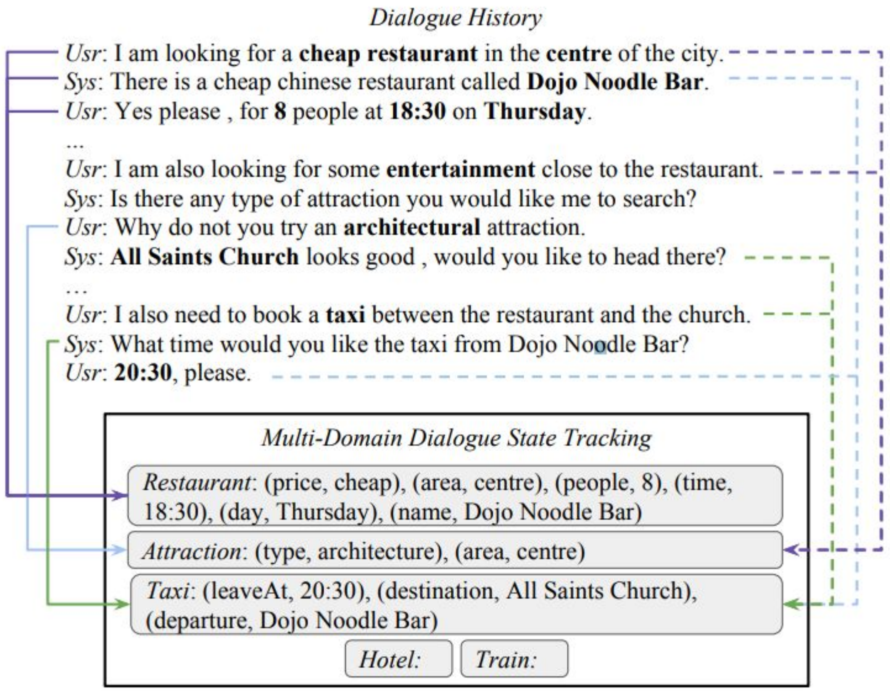
CS 886
Joshua Lemmon

Task-Oriented Dialogue Systems

- Systems like Siri or Alexa
- Must interpret what the user is saying/requesting
- Requires knowledge of multiple domains
- Sometimes must use inferences from dialogue history to interpret user meaning

Motivation

- Dialogue state tracking (DST) is an important component in task-oriented dialogue systems
- Some systems have an over-dependence on domain ontology or lack knowledge sharing across domains
- Prior systems struggled with inferring unknown dialogue slot values and have difficulty adapting to new domains
- Want to have model that can track multiple dialogue states over several domains across a dialogue history


Dialogue History

Usr: I am looking for a **cheap restaurant** in the **centre** of the city.
Sys: There is a cheap chinese restaurant called **Dojo Noodle Bar**.
Usr: Yes please , for **8** people at **18:30** on **Thursday**.
...
Usr: I am also looking for some **entertainment** close to the restaurant.
Sys: Is there any type of attraction you would like me to search?
Usr: Why do not you try an **architectural** attraction.
Sys: **All Saints Church** looks good , would you like to head there?
...
Usr: I also need to book a **taxi** between the restaurant and the church.
Sys: What time would you like the taxi from Dojo Noodle Bar?
Usr: **20:30**, please.

Multi-Domain Dialogue State Tracking

Restaurant: (price, cheap), (area, centre), (people, 8), (time, 18:30), (day, Thursday), (name, Dojo Noodle Bar)
Attraction: (type, architecture), (area, centre)
Taxi: (leaveAt, 20:30), (destination, All Saints Church), (departure, Dojo Noodle Bar)
Hotel:
Train:

Example dialogue history from the dataset. Note the system must refer to previous lines in the dialogue to gain more context to a new dialogue line.

Some Definitions

$X = \{(U_1, R_1), \ldots, (U_T, R_T)\}$ Set of user utterances and system response pairs

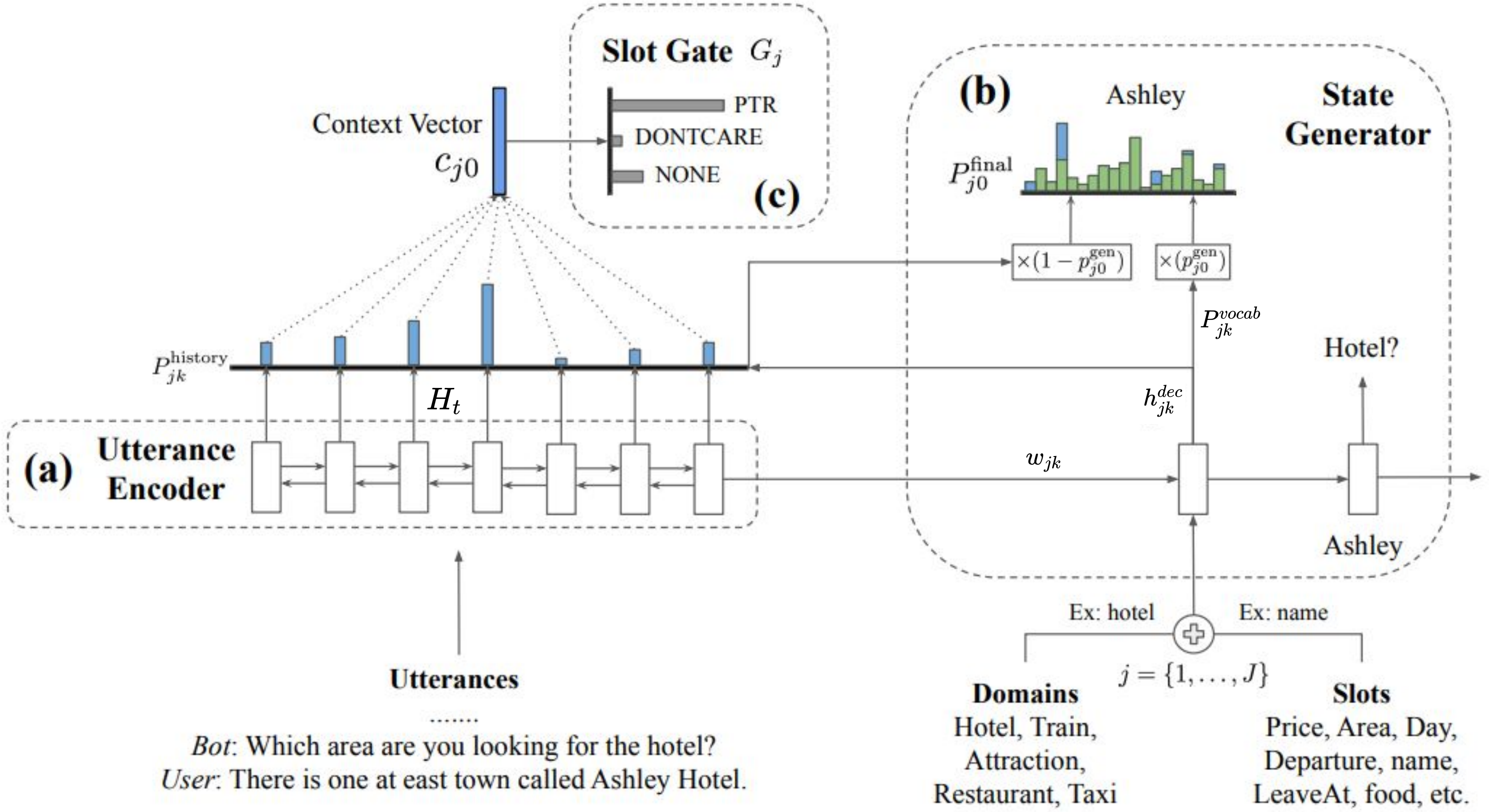
$B = \{B_1, \ldots, B_T\}$ - Dialogue states for each turn

B_t is a tuple(domain: D_n, slot: S_m, value: Y_j^{value}) - The domain of the state and the slot of the state. Given J possible domain/state pairs, Yj is the true word sequence

Domain: An overall area of user interest, e.g. restaurant
Slot: Important attributes of a domain, e.g. for restaurant some slots are price and food
Value: A possible word that is the value for a domain-slot pair, e.g. (restaurant, price, cheap) or (restaurant, food, italian)

TRADE

- **TRA**nsferable **D**ialogue stat**E** generator
- Generates dialogue states from utterances using a copy mechanism which facilitates knowledge transfer when predicting (*domain*, *slot*, *value*) triplets that were unknown during training
- Three major parts:
  - Utterance Encoder
  - Slot Gate
  - State Generator
- Generates slot values instead of predicting probability distribution of every predefined ontology term



Slot Gate G_j
PTR
DONTCARE
NONE
(c)
Context Vector
c_{j0}
(b)
Ashley
State Generator
P_{j0}^{final}
$\times(1-p_{j0}^{\text{gen}})$
$\times(p_{j0}^{\text{gen}})$
P_{jk}^{vocab}
Hotel?
h_{jk}^{dec}
w_{jk}
Ashley
P_{jk}^{history}
H_t
(a) Utterance Encoder
Utterances
....
Bot: Which area are you looking for the hotel?
User: There is one at east town called Ashley Hotel.
Ex: hotel
Ex: name
$j = \{1, \ldots, J\}$
Domains
Hotel, Train, Attraction, Restaurant, Taxi
Slots
Price, Area, Day, Departure, name, LeaveAt, food, etc.

Utterance Encoder

- Encodes dialogue utterances into fixed length vectors
- TRADE uses bi-directional gated recurrent units
- Input is the dialogue ***history*** $X_t = [U_{t-l}, R_{t-l}, \ldots, U_t, R_t] \in \mathcal{R}^{|X_t| \times d_{emb}}$
  - U is a user utterance, R is a system response
  - ***l*** is the number of selected dialogue turns before the final turn t
  - $d_{emb}$is the size of the embedding
  - Concatenation of all words in the dialogue history
- The encoded dialogue history is represented as $H_t = [h_1^{enc}, \ldots, h_{|X_t|}^{enc}] \in \mathcal{R}^{|X_t| \times d_{hdd}}$
  - d_{hdd} is the size of the hidden encoder state

Section (b) of slide 7

State Generator

- A copy mechanism is required to generate slot values using text from the input
- TRADE uses soft-gated pointer-generator copying
    - Combines a distribution over the vocabulary and a distribution over the dialogue history into a single distribution
- State generator uses a GRU to decode and predict each of the possible domain/slot pairs
    - First input to the decoder is the summed embedding of domain and slot
- At decoding step k for j-th pair, the generator takes an embedding w_{jk} and outputs a hidden state h_{jk}^{dec}
- The generator maps this into the vocabulary space P_{jk}^{vocab} using a trainable embedding $E \in \mathcal{R}^{|V| \times d_{hdd}}$
- At the same time, the hidden state is also used to compute the history attention $P_{jk}^{history}$ over H_t

$$
\begin{aligned}
P_{jk}^{\text{vocab}} &= \text{Softmax}(E \cdot (h_{jk}^{\text{dec}})^\top) \in \mathbb{R}^{|V|}, \\
P_{jk}^{\text{history}} &= \text{Softmax}(H_t \cdot (h_{jk}^{\text{dec}})^\top) \in \mathbb{R}^{|X_t|}
\end{aligned}
$$

State Generator

- Final weighted distribution is $P_{jk}^{final} = p_{jk}^{gen} \times P_{jk}^{vocab} + (1 - p_{jk}^{gen}) \times P_{jk}^{history} \in \mathcal{R}^{|V|}$
- p_{jk}^{gen} is a trainable scalar value that tries to find the optimal combination ratio
- p_{jk}^{gen} is computed by: $Sigmoid(W_1 \cdot [h_{jk}^{dec}; w_{jk}; c_{jk}]) \in \mathcal{R}^1$
- Where $c_{jk} = P_{jk}^{history} \cdot H_t \in \mathcal{R}^{d_{hdd}}$ and W_1 is a trainable weight matrix
- Since P_{jk}^{final} takes the history attention into account, the model can generate words that were not predefined in the vocabulary

Slot Gate

- The slot gate is a three way classifier
- Maps a context vector from the hidden states H_t to a probability distribution over *ptr*, *none* and *dontcare* classes
  - *ptr* means that the context points to a slot that matches the current *(domain, slot)* pair, and the generated words from the state generator as the slot value
  - *none* means that there is no valid context
  - *dontcare* means that the context is for a slot that doesn't match the current pair, so it is ignored
- The slot gate is defined as $G_j = Softmax(W_g \cdot (c_{j0})^\top) \in \mathcal{R}^3$
- c_{j0} is the context vector for the first decoder hidden state
  - Paper does not mention why only the first decoder hidden state is used

Training Optimization

- Both the slot gate and the state generator are optimized
- The slot gate uses a cross-entropy loss function with the predicted slot gate G_j and the true label y_j^{gate}

$$L_g = \sum_{j=1}^{J} -log(G_j \cdot (y_j^{gate})^\top)$$

- The state generator uses a different cross entropy loss with P_{jk}^{final} and the true slot word Y_j^{label}

$$L_v = \sum_{j=1}^{J} \sum_{k=1}^{|Y_j|} -log(P_{jk}^{final} \cdot (y_{jk}^{value})^\top)$$

- Both losses are optimized as a weighted sum using two hyper parameters α and β

$$L = \alpha L_g + \beta L_v$$

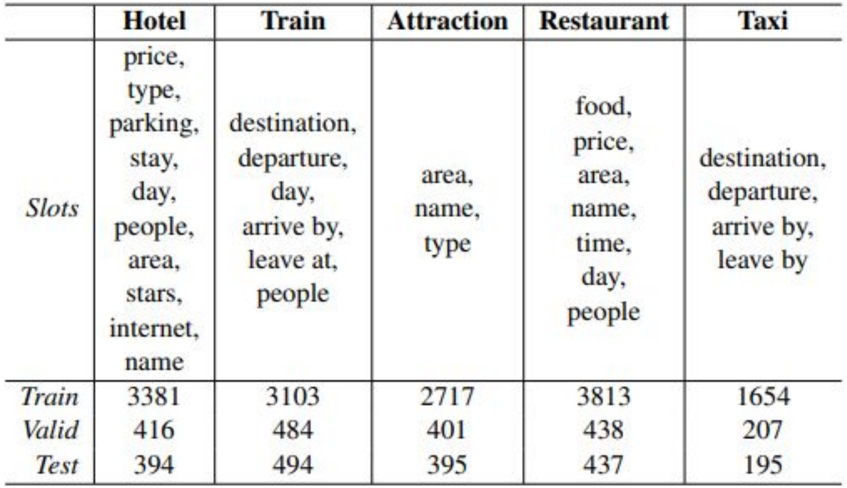
Unseen Domain DST

- Strength of TRADE is that it can generalize fairly well to an unseen domain
- Can utilize both zero-shot transferring and few-shot domain expanding
    - Zero-shot transferring: Assume that the unknown domain has no training data
    - Few-shot domain expanding: Assume only 1% of training data for the unseen domain is available
- Unseen domain DST is useful because creating a large-scale dataset for a domain is time-consuming and difficult
- Many unknown domains are encountered in real world scenarios

Unseen Domain DST

- TRADE can perform zero-shot transferring if slots from previous domains appear in the new domain
    - E.G. if model can track the ***departure*** slot for the ***train*** domain, there should be overlap with the unseen ***taxi*** domain and its ***departure*** slot
- For few-shot domains, TRADE utilizes elastic weight consolidation (EWC) and gradient episodic memory (GEM) for training
- EWC uses a special loss $L_{ewc}(\Theta) = L(\Theta) + \sum_i \frac{\lambda}{2} F_i (\Theta_i - \Theta_{S,i})^2$ where F is the Fisher information matrix, λ is a hyperparameter, Θ_S is the model parameters for the source (known) domain(s) and Θ the parameters for the target (unseen) domain
- GEM keeps a small number of source domain samples, and applies a constraint on the loss while learning the unseen domain
    - This constraint prevents the loss from increasing when using the stored samples

Experimental Dataset



	Hotel	Train	Attraction	Restaurant	Taxi
Slots	price, type, parking, stay, day, people, area, stars, internet, name	destination, departure, day, arrive by, leave at, people	area, name, type	food, price, area, name, time, day, people	destination, departure, arrive by, leave by
Train	3381	3103	2717	3813	1654
Valid	416	484	401	438	207
Test	394	494	395	437	195

TRADE is trained on a dataset called MultiWOZ, the largest existing human-human conversation corpus.

Training and Evaluation Details

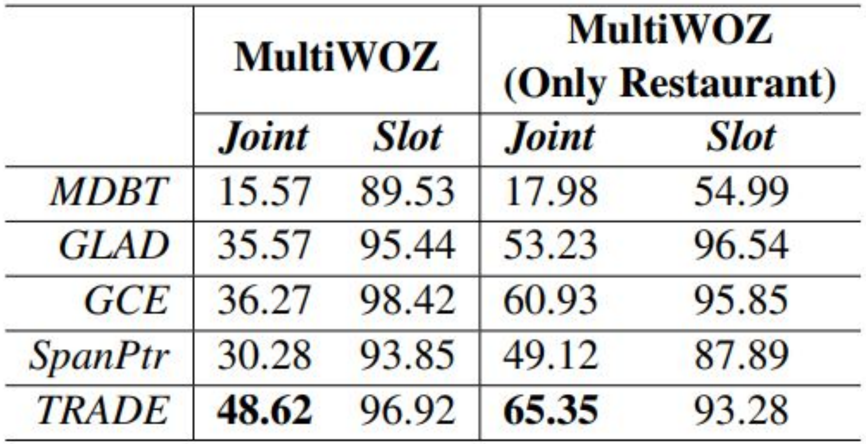
- Trained end-to-end using the Adam optimizer, a batch size of 32, an annealing learning rate in [0.001, 0.0001] and a dropout ratio of 0.2
- The α and β parameters were both set to 1
- The initial word embeddings are the concatenation of the Glove embedding and the character embedding, with a dimension of 400
- To simulate an out-of-vocabulary setting, a word dropout is used in the utterance encoder to randomly mask small amounts of input tokens
- Two evaluation metrics are used
    - Joint Goal Accuracy: compares predicted dialogue states to the ground truth B_t and only considered correct if all the output values match exactly
    - Slot accuracy: compares predicted (domain, slot, value) triplet to the ground truth label

Comparison to Similar Models

TRADE is compared to the following models:

- MDBT
- GLAD
- GCE (current S.O.T.A)
- SpanPtr

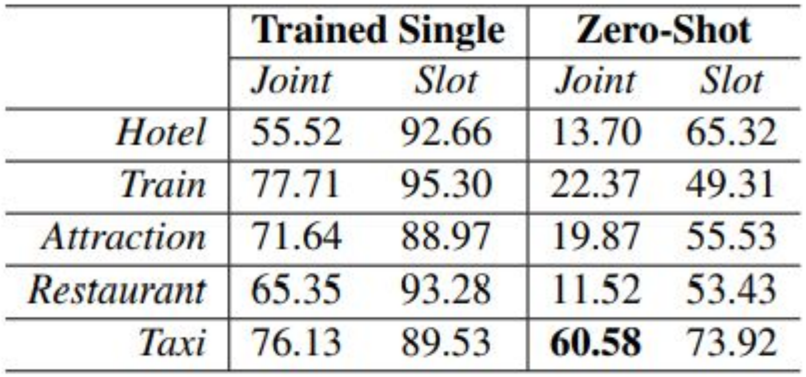
The authors have modified the above models to be usable with the MultiWOZ dataset.
While TRADE does not always achieve highest slot accuracy, it does have the best joint accuracy. This means that when training on multiple domains at the same time, TRADE can predict the exact dialogue state more often than the other models.



	MultiWOZ		**MultiWOZ (Only Restaurant)**	
	Joint	*Slot*	*Joint*	*Slot*
MDBT	15.57	89.53	17.98	54.99
GLAD	35.57	95.44	53.23	96.54
GCE	36.27	98.42	60.93	95.85
SpanPtr	30.28	93.85	49.12	87.89
TRADE	**48.62**	96.92	**65.35**	93.28

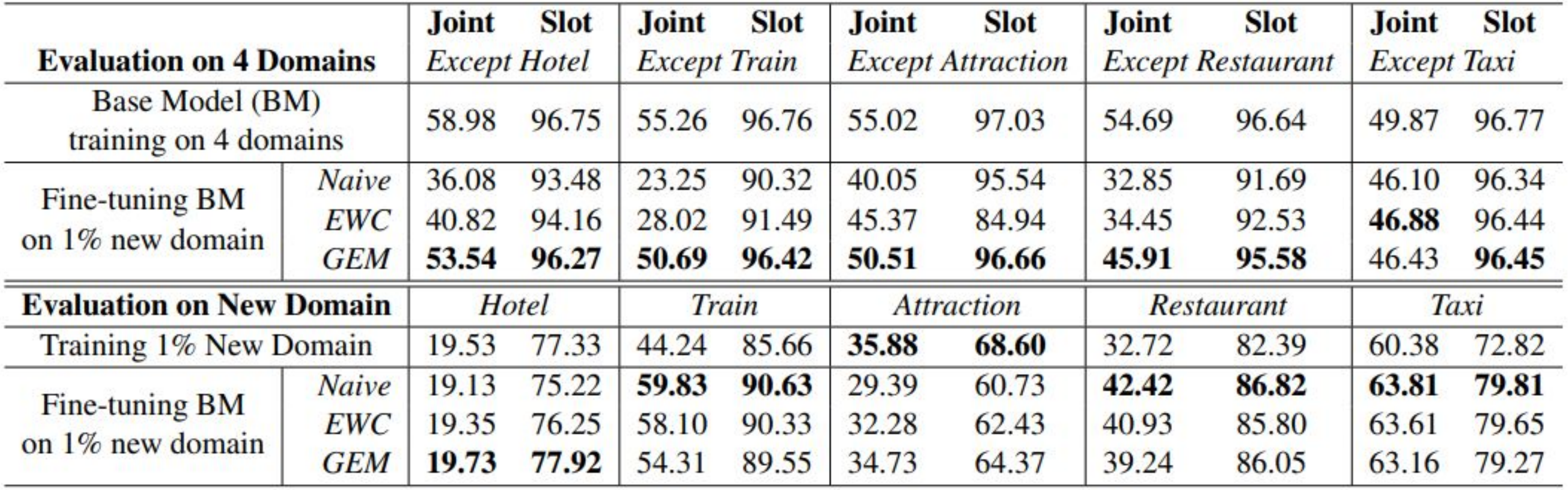
Zero Shot results

- Results on the zero-shot experiment done by excluding one domain during training.
- The ***Taxi*** domain achieves relatively high results, and is quite close to the singly trained taxi domain.
- ***Taxi*** is high because all of it's slots share similar values to the ***Train*** domain.
- The authors mention that while the other domain performances are not as promising, the slot accuracy is still impressive despite having no in-domain samples.
- Using more similar training domains could help increase the zero-shot accuracy.



	Trained Single		**Zero-Shot**	
	Joint	*Slot*	*Joint*	*Slot*
Hotel	55.52	92.66	13.70	65.32
Train	77.71	95.30	22.37	49.31
Attraction	71.64	88.97	19.87	55.53
Restaurant	65.35	93.28	11.52	53.43
Taxi	76.13	89.53	**60.58**	73.92

Domain Expanding Results



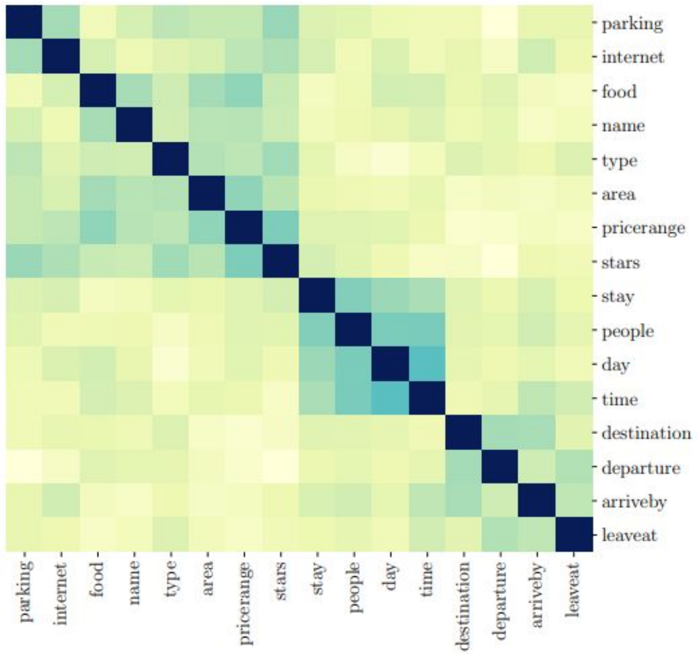
Evaluation on 4 Domains		Joint *Except Hotel*	Slot	Joint *Except Train*	Slot	Joint *Except Attraction*	Slot	Joint *Except Restaurant*	Slot	Joint *Except Taxi*	Slot
Base Model (BM) training on 4 domains		58.98	96.75	55.26	96.76	55.02	97.03	54.69	96.64	49.87	96.77
Fine-tuning BM on 1% new domain	*Naive*	36.08	93.48	23.25	90.32	40.05	95.54	32.85	91.69	46.10	96.34
	EWC	40.82	94.16	28.02	91.49	45.37	84.94	34.45	92.53	**46.88**	96.44
	GEM	**53.54**	**96.27**	**50.69**	**96.42**	**50.51**	**96.66**	**45.91**	**95.58**	46.43	**96.45**
Evaluation on New Domain		*Hotel*		*Train*		*Attraction*		*Restaurant*		*Taxi*	
Training 1% New Domain		19.53	77.33	44.24	85.66	**35.88**	**68.60**	32.72	82.39	60.38	72.82
Fine-tuning BM on 1% new domain	*Naive*	19.13	75.22	**59.83**	**90.63**	29.39	60.73	**42.42**	**86.82**	**63.81**	**79.81**
	EWC	19.35	76.25	58.10	90.33	32.28	62.43	40.93	85.80	63.61	79.65
	GEM	**19.73**	**77.92**	54.31	89.55	34.73	64.37	39.24	86.05	63.16	79.27

Domain Expanding Results

- The Base Model (BM) indicates results evaluated on the four domains using their in-domain training data
    - BM is then fine tuned using the 1% unknown domain
    - Naive means that the BM was just retrained with the new data with no special technique
- In general GEM outperforms the naive model and EWC in terms of overcoming catastrophic forgetting
    - Shows very little accuracy drop after learning the new domain
- Pre-training followed by fine-tuning outperforms training from scratch on a single domain
- Much smaller accuracy drop after fine-tuning with GEM
- Fine-tuning increases evaluation of the new domain in general
    - For the new domain evaluation EWC and GEM are generally outperformed by the naive retraining approach
- Overall fine-tuning is better than the BM, and GEM is useful for minimizing accuracy loss of the original domains

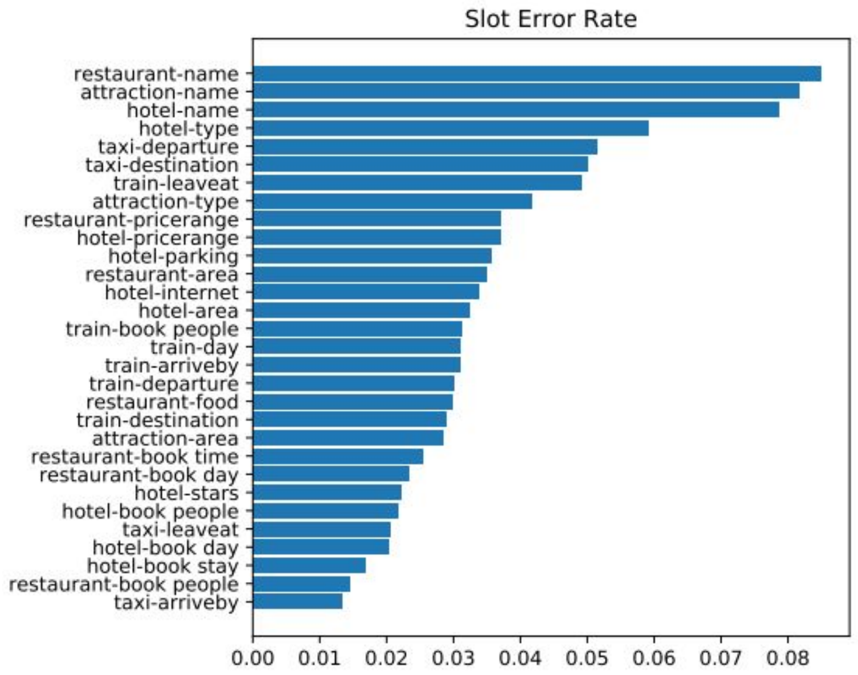
Slot Embedding Similarity

- Calculated the cosine similarity between the embeddings of all the possible slots in MultiWOZ
- More blue = more similar
- Shows slots with similar or correlated values learn similar embeddings
- ***destination*** and ***departure*** share similar values
  - Names of cities
- ***price range*** and ***stars*** share correlated values
  - 5 star hotels will likely be more expensive



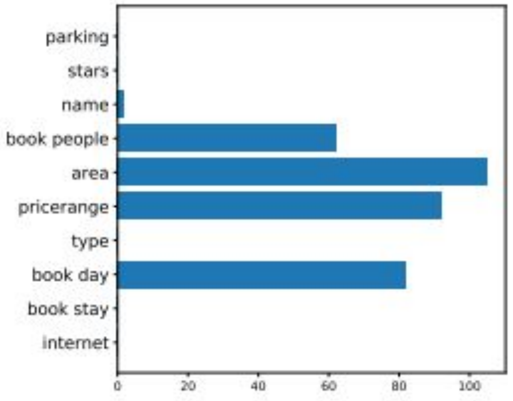
Slot Error Rates

- Analysis of the highest error-rate slots
- x-axis is proportion of mis-predicted values in a given slot
- Typically name slots have highest error rate since they have so many possible values
- Number-related slots typically have the lowest error rate
- Some error rates higher than they should because the labels of the domain-slot pair is missing from the data
    - Missing label results in an incorrect prediction no matter what

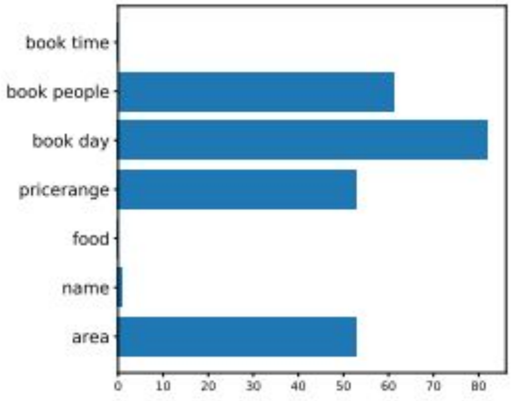


Knowledge Transferring Error Analysis

- Model was trained zero-shot on hotel and restaurant domains only
  - Trained on ***Restaurant*** and tested on ***Hotel*** for (a) and vice versa for (b)
- x-axis is # of correctly predicted slot values
- Shows very high accuracy results between the similar slots
- Slots that don't exist between domains are very difficult to track



(a) Hotel



(b) Restaurant

Conclusion

- TRADE model can learn to track states without predefined domain ontology
- TRADE shares all parameters across multiple domains
- Achieves state-of-the-art joint goal and slot goal accuracy on MultiWOZ dataset
- Can perform limited zero-shot DST for unseen domains
- Can perform few-shot DST for unseen domains without forgetting learned domains