Embedding Memory into Advanced Dialogue Management

MASTER THESIS

Spring Semester 2019

Author: Chia-An YU

Machine Learning and Optimization Laboratory

School of Computer and Communication Sciences

École Polytechnique Fédérale de Lausanne

Swisscom Digital Lab

Professor: Martin JAGGI

Supervisor: Claudiu MUSAT

August 15, 2019

Abstract

Frame tracking is a way to embed memory into a dialogue system. Inspired by how human resolves frame references, we propose a new model with an attention mechanism. We also devise a new method to generate synthetic frame tracking data. Empirical analyses show that our method improves accuracy by 6.7 percentage points and shortens training time.

Contents

List of Figures

List of Tables

Embedding Memory into Advanced Dialogue Management

Chia-An YU

August 15, 2019

1 Introduction

Dialogue based assistants, also known as chatbots, are becoming increasingly popular. As improvements in machine learning help us better understand the nuances of natural language, dialogue systems can accomplish more and more complex tasks.

Existing implementations of dialogue systems are most often goal-oriented and help the user perform a specific task like booking a train ticket or reserving a table. A goal-oriented dialogue system is like an interface between user and database. When a user knows exactly what he wants, the workflow is straightforward: the system extracts information from user's requests, translate it into database query, and return results to the user. However, this does not work for users who want to explore different options and even compare them, which could happen in many situations such as booking hotels or flights. The reason is that the system keeps only the latest information and loses previous ones when a user changes his mind.

The frame tracking task is proposed to solve this problem [1]. In frame tracking, the system maintains a list of frames during a dialogue. A frame is the context for the current turn, or in other words, a summary of a discussion in a dialogue. For example, in a dialogue about booking a traveling package, a frame may contain the price, the number of people, check-in and check-out date, etc., as shown in Table 1.1. The main idea of frame tracking is to make use of this information to improve the system. For each turn in a dialogue, the system finds out the frames related to the current utterance and create frame references. As the conversation goes on, the system may create new frames or change the referred frames. If we consider the list

TABLE 1.1: An example of a frame.

Slot	Value
Budget	21.3
Origin city	Kochi
Destination city	Denver
# adults	2
# children	6
Duration	7
Start date	August 27
End date	September 1
Category	4.0
Price	19028.93

of frames as the memory of the system, the two operations are like storing into and retrieving from memory.

In this paper, we focus on the problem of predicting frame reference. The input of the problem consists of two parts: one is a list of frames $F = \{f_1, \ldots, f_n\}$ representing the dialogue history, the other is a natural language understanding (NLU) label of the current utterance. The output is the predicted frame reference, which should be a frame f_i in the list F indicating that this is the frame most related to the label.

Previous research [2] on frame tracking uses recurrent neural networks (RNN) to encode frames into vectors. However, this ignores the fact that each slot-value pair has different importance when the reference NLU label is different. For example, when the label is a location, the destination city and origin city slots should be more important than the number of people when we try to find the most relevant frame. Therefore, we propose a new model that uses an attention mechanism to account for this.

Another challenge of frame tracking is the limited amount of available data. The only dataset that has frame tracking labels is FRAMES [1], which only has 1369 dialogues. This is relatively small comparing to other benchmark dialogue datasets. So we come up with a method to create synthetic frame tracking data using large dialogue datasets that don't have frame tracking labels.

1.1 Related work

Recently, researches start to develop memory-enhanced dialogue systems. In addition to creating extra frame tracking component to the dialogue system, some methods focus on incorporating memory into existing components such as slot filling and state tracking. Chen et al. [3] used RNN to encode contextual sentences, which are further encoded using an attention mechanism to enhance the language understanding tagging sequence. Several pieces of research work on improving state tracking: Lee et al. [4] work at the semantic level, i.e. representing each turn as the NLU results, while Sharma et al. [5] use the textual content directly. Perez et al. [6] approach this problem differently. They formulate state tracking as a question-answering problem and use MemN2N [7] to model the memory.

2 Frame tracking

2.1 FRAMES dataset

FRAMES dataset is created specifically for the frame tracking task. Its dialogues are longer and more complicated comparing to the ones in other datasets, and there are special labels for frame tracking. The dialogues were collected in a wizard-of-oz fashion: two people playing the role of user and system, with the user having a goal hidden from the system, and with the system having a database unknown to the user [8]–[10]. The users' tasks are designed so that users have to go back-and-forth between options to achieve their goal and thus creates complicated dialogue structures. For each utterance, there are annotations of NLU labels, i.e. action, slot, and value, and an optional frame reference label if this action-slot-value tuple is related to a previous frame. The frame is then represented as a list of slot-value pairs.

The frame references appearing in FRAMES can be categorized into three types: explicit, anaphora, and implicit. Explicit reference is when the NLU label appears in the referred frame, i.e. the user mentions a slot value explicitly. Reference using anaphora is similar except that the user uses anaphora instead of a slot value. The problem of resolving this kind of reference is closely related to the coreference resolution in dialogues. Implicit reference happens when the utterance itself contains no clue and the reference depends heavily on the context of the dialogue. In such cases, the referred frame is usually the one mentioned in the previous turn.

Here are some examples of each type of frame reference. The slot values having a reference are in bold.

- Explicit: a detailed example is in Table 2.1.
  - How many days would I be in **Kobe**?
  - Ok, I would like to purchase the trip with the **4 star** hotel.
- Anaphora: a detailed example in Table 2.3.
  - No, **that**'s too far for me. I need a flight that leaves from Birmingham.
  - Ok, then I would like to purchase **this package**. What activities are included in this package?

TABLE 2.1: An example of explicit frame reference. The frames are in Table 2.2.

Turn	Author	Utterance	Remark
1	User	Hello, I am looking to book a trip for 2 adults and 6 children for $21,300 or less. We are departing from Kochi for Denver.	Create frame 1
2	System	I have several options available within your budget. How long would you like to travel for? And do you have dates in mind?	Update frame 1
3	User	I do not have any dates in mind. I would like to spend as much time in Denver as my budget will allow.	Update frame 1
4	System	I can book 7 days at a 4.0 star hotel for 19028.93USD. I can also book 7 days at a 3.0 star hotel for 12824.84USD.	Create frame 2 and 3
5	User	Do these packages have different departure dates? When would I be leaving for each of them?	
6	System	The 3.0 star trip leaves Kochi August 26 and returns August 31. The 4.0 star leaves August 27 from Kochi and returns September 1.	Update frame 2 and 3
7	User	Ok, I would like to purchase the trip with the **4-star** hotel.	Refer to frame 2

- Implicit: without context, these might look the same as explicit references. The difference is that the slots and values here do not appear in any existing frames thus can not be explicit references. A detailed example is in Table 2.5.
  - Is **breakfast** included?
  - Yes perfect. How is the **hotel**?
  - Reasonable. Any free **wifi** for the kids?

2.2 Synthetic dataset

The amount of data in FRAMES is limited. To have more training data for frame tracking, we generate synthetic dialogues that have a similar structure as the ones in FRAMES. More precisely, we create actions of switching frame reference by interleaving multiple simple dialogues.

The operation of interleaving a set of dialogues is defined as follows. For each user turn, the utterance is chosen from any dialogue in the set, and utterance of the following system turn is the response to the corresponding user utterance. The chosen

TABLE 2.2: Frames created in the example dialogue of explicit frame reference (Table 2.3).

(A) Frame 1

Slot	Value
Intent	book
Budget	21.3
Origin city	Kochi
Destination city	Denver
# adults	2
# children	6
Count	several options
Start date	-1
End date	-1
Max duration	$MAX

(B) Frame 2

Slot	Value
Intent	book
Budget	21.3
Origin city	Kochi
Destination city	Denver
# adults	2
# children	6
Max duration	$MAX
Duration	7
Start date	August 27
End date	September 1
Category	4.0
Price	19028.93

(C) Frame 3

Slot	Value
Intent	book
Budget	21.3
Origin city	Kochi
Destination city	Denver
# adults	2
# children	6
Max duration	$MAX
Duration	7
Start date	August 26
End date	August 31
Category	3.0
Price	12824.84

TABLE 2.3: An example of anaphora frame reference. The frames are in Table 2.4. The reference in turn 11 and “the hotel" part in turn 13 is an anaphora, and the “Fortaleza" in turn 13 is explicit because it directly mentions a slot value.

Turn	Author	Utterance	Remark
1	User	Hey im looking to check out fortaleza. I’m leaving from Sapporo on August 27	Create frame 1
2	System	Hi there! I can offer you a six-day vacation package starting August 27th for only 3933.10USD. Does this fit within your budget?	Create frame 2
3	User	When is the return date?	
4	System	You would be returning on the 31st of August.	Update frame 2
5	User	I would like to travel until September 7, or closer to that date. ...	
⋮	⋮	⋮	⋮
9	User	Ok. One last destination I am wondering about. Curitiba - what’ve you got there?	Create frame 6
10	System	How about August 30th - September 6th at the Hotel Leisure? 4234.65USD if you book now.	Create frame 7
11	User	I like the sounds of that. What is **the hotel** like? Any details?	Refer to frame 7
12	System	The Hotel Leisure is a 3-star resort with a 6.91/10 guest rating, free breakfast, free wifi and free parking. Would you like me to book this destination?	Update frame 7
13	User	What is **the hotel in Fortaleza** like?	Refer to frame 2

TABLE 2.4: Frames created in the example dialogue of anaphora frame reference (Table 2.3).

(A) Frame 1

Slot	Value
Intent	book
Origin city	Sapporo
Destination city	Fortaleza
Start date	August 27

(B) Frame 2

Slot	Value
Intent	book
Origin city	Sapporo
Destination city	Fortaleza
Start date	August 27
End date	August 31
Duration	6
Price	3933.1

(C) Frame 6

Slot	Value
Intent	book
Origin city	Sapporo
Destination city	Curitiba
Start date	August 27

(D) Frame 7

Slot	Value
Intent	book
Origin city	Sapporo
Destination city	Curitiba
Start date	August 30
End date	September 6
Name	Hotel Leisure
Category	3.0
Price	4234.65
Guest rating	6.91
Breakfast	true
Parking	true
Wifi	true

TABLE 2.5: An example of implicit frame reference. The frames are in Table 2.6.

Turn	Author	Utterance	Remark
1	User	Hello I am looking to bring my 6 kids and I to Portland and leave from Minneapolis. We can travel whenever and have no budgetary constraints.	Create frame 1
2	System	Unfortunately we don't have any available trips to Portland as of right now. Is there another city you can go to?	Update frame 1
3	User	Ahh... Let's check Santos then.	Create frame 2
4	System	Okay, our best deal in Santos is a 7 day trip at a 3.5 star hotel near a museum and a palace.	Create frame 3
5	User	Reasonable. Any **free wifi** for the kids?	Refer to frame 3

TABLE 2.6: Frames created in the example dialogue of implicit frame reference (Table 2.5).

(A) Frame 1

Slot	Value
Intent	book
Origin city	Minneapolis
Destination city	Portland
# children	6
Start date	-1
Budget	-1
No result	true

(B) Frame 2

Slot	Value
Intent	book
Origin city	Minneapolis
Destination city	Santos
# children	6
Start date	-1
Budget	-1

(C) Frame 3

Slot	Value
Intent	book
Origin city	Minneapolis
Destination city	Santos
# children	6
Start date	-1
Budget	-1
Category	3.5
Duration	7
Museum	true
Palace	true

TABLE 2.7: Synthetic datasets.

	# dialogues	Domain(s)
FRAMES	1369	Hotel + flight
Synthetic 1	5488	Single domain
Synthetic 2	3820	Hotel + restaurant
Synthetic 3	3820	Hotel + transportation (taxi and train)

dialogue can be arbitrary in each user turn, but the turns should be taken in the same order as in the original dialogue. The result is as if multiple users were talking to the system at the same time. This is an effective way to create artificial frame switching actions because there is a context switch whenever we choose a dialogue that is different from the source of the previous turn. However, such switches can be ambiguous if dialogues are chosen arbitrarily in each turn. Some turns do not have enough information themselves to identify a frame without context. If those turns are placed after another turn from a different dialogue, it might be impossible to find out the referred frame or even notice a switch exists. To avoid this problem, we define the notion of identifying turn. An identifying turn is a turn that has explicit mention of slot values in its referred frame. When interleaving dialogues, we switch dialogues only at identifying turns.

The frame label of the interleaved dialogue can be easily obtained from the source dialogues: if a new frame is created in a turn of source dialogue, the same frame is created in the corresponding turn in the interleaved dialogue. Similarly, for each slot-value in the utterance of interleaved dialogue, the frame reference is copied directly from its source dialogue.

The dataset we use to create synthetic data is MultiWOZ 2.0 [11]. It is a large scale goal-oriented dataset created for multi-domain dialogue modeling. It consists of about 10,000 dialogues and covers a wide range of domains, including restaurant, train, hotel, etc. It has both single domain dialogue and multi-domain dialogue. Despite having rich content, it only has dialogue state label for every two turns (user turn and system turn) and no annotation of frames so we have to convert the labels into frame labels. We then take the single domain dialogues along with their frame labels and interleave them to generate synthetic datasets. We generate one synthetic dialogue by interleaving two dialogues. The two dialogues are in the same domain for the first synthetic dataset, and the domains are different for the other two synthetic datasets. More information on the three synthetic datasets we generated are listed in Table 2.7.

3 Methodology

3.1 Model

Our proposed model consists mainly of two parts: one is an NLU label encoder, and the other is a frame encoder. The two encoders produce a vector for the corresponding part of the input. We use a dot product to compute the similarity score between an NLU label and a frame. The output is a probability distribution over all candidate frames computed by taking softmax of all the similarity scores of the frames. Softmax is a kind of normalization that transforms a real vector into a discrete probability distribution. It is defined as

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}, \text{for } i = 1 \dots n\,. \tag{3.1}$$

The high-level overview of the model architecture is in Figure 3.1.

3.1.1 Embedding of input

Both NLU label encoder and frame encoder use text embedding and token embedding to convert structured input into vectors. The embeddings are shared between the two encoders. There are three embeddings: text embedding, act embedding, slot embedding. The act embedding and slot embedding are very similar. We treat each act and slot as a unique token and assign them a trainable embedding vector. The text embedding is used to embed slot values.

We use two different text embedding models. The first one, shown in Figure 3.2, is based on a pre-trained BERT [12] feature. We use a uncased 12-layered BERT model[1]. The embedding of a slot value is the average of the last layer. We use a linear projection to adjust the dimension of the embedding so that it is compatible with the other part of the model.

The other text embedding is based on letter trigram and RNN (Figure 3.3. We first tokenize the text using nltk's TweetTokenizer[2] and split each token into letter trigrams. For example, the list of letter trigram of "hotel" is "#ho", "hot", "ote", "tel", and "el#". We assign each trigram to a trainable embedding vector. The embedding

[1] https://github.com/huggingface/pytorch-transformers
[2] https://www.nltk.org/

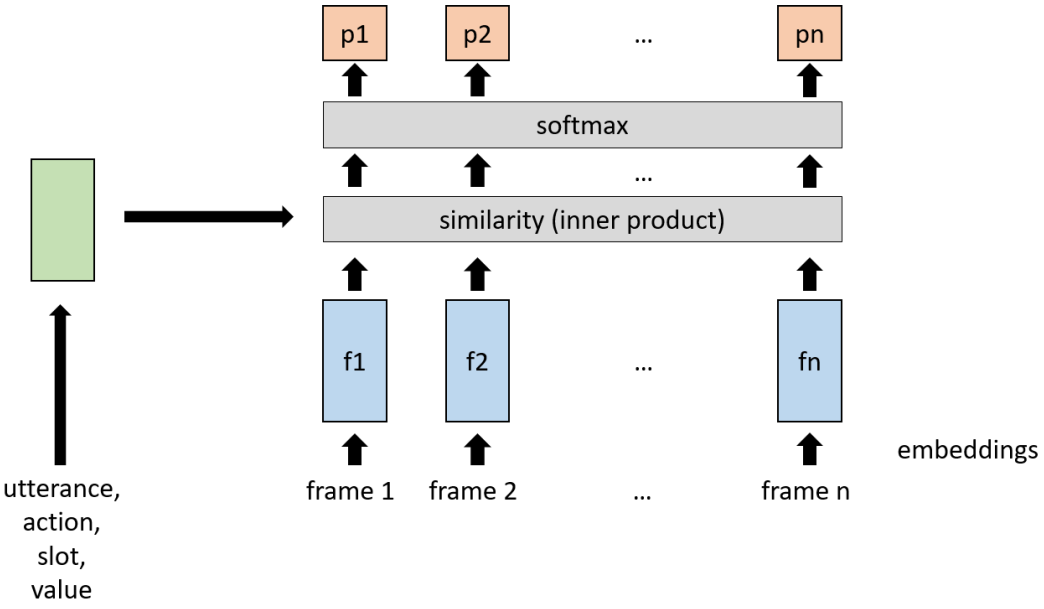

FIGURE 3.1: Model architecture. The green block on the left is NLU label encoder (Figure 3.4). The blue blocks at the bottom are frame encoders (Figure 3.5).

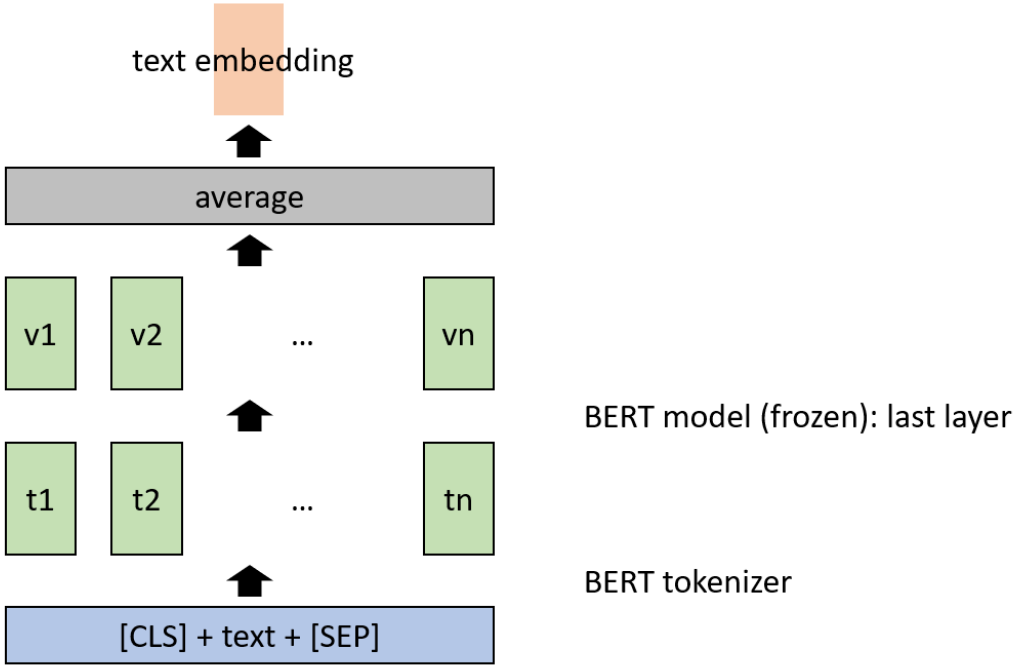

FIGURE 3.2: Architecture of text embedding using pre-trained BERT.

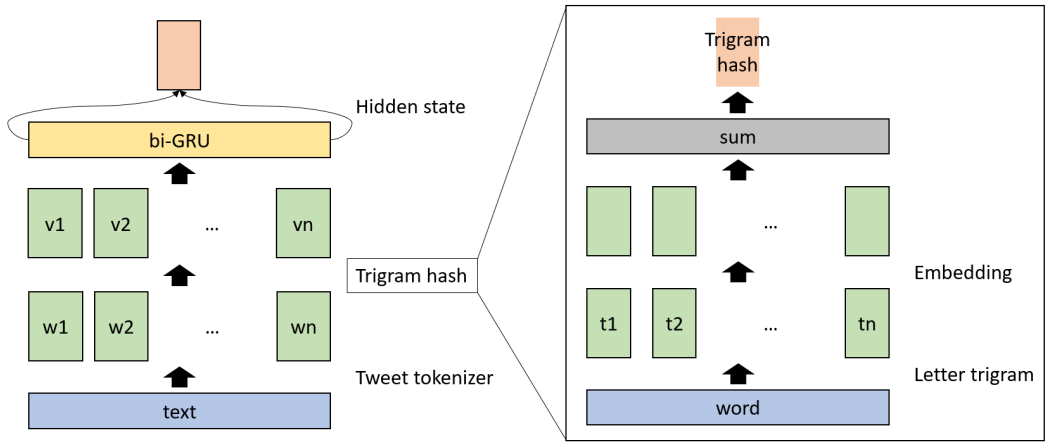

FIGURE 3.3: Architecture of text embedding using letter trigram and GRU.

of a token is then the sum of embedding of its trigrams. We then use a bidirectional GRU to encode all tokens into one vector. The final embedding is the concatenation of forward and backward hidden state of the GRU.

3.1.2 NLU label encoder

This encoder takes as input an act embedding, a slot embedding, and a value embedding. We treat them as a sequence of length three and pass it through a bidirectional GRU. We then concatenate the forward and backward hidden state and project it to have a proper dimension. The architecture is in Figure 3.4.

3.1.3 Frame encoder

A frame is a list of slot-value pairs. To encode this, we concatenate the slot and value embedding vectors for each slot-value pair and pass the list of concatenated vectors through a bidirectional GRU.

If there is no attention mechanism, we concatenate the forward and backward hidden state of the GRU and use a linear projection to compute the output vector. If an attention mechanism is used, we concatenate the outputs of both direction and project each vector to adjust the dimension. The final output is then the linear combination of the resulting vectors, with coefficients being attention scores normalized by softmax. The architecture with attention mechanism is shown in Figure 3.5.

The detailed formula of each attention mechanism is in Table 3.1. In addition to an encoded frame, most of the attention mechanisms we use also take an encoded NLU label as input so that the attention score is adapted for different situations. The only

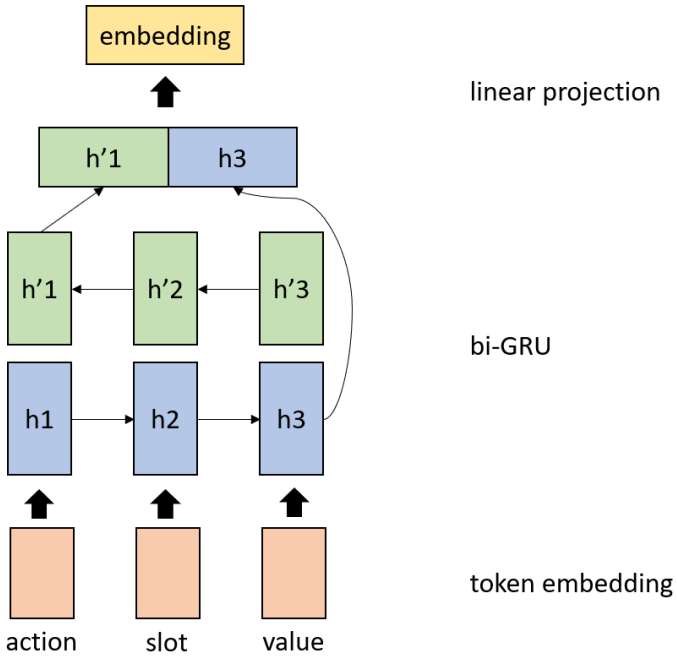

FIGURE 3.4: Architecture of NLU label encoder.

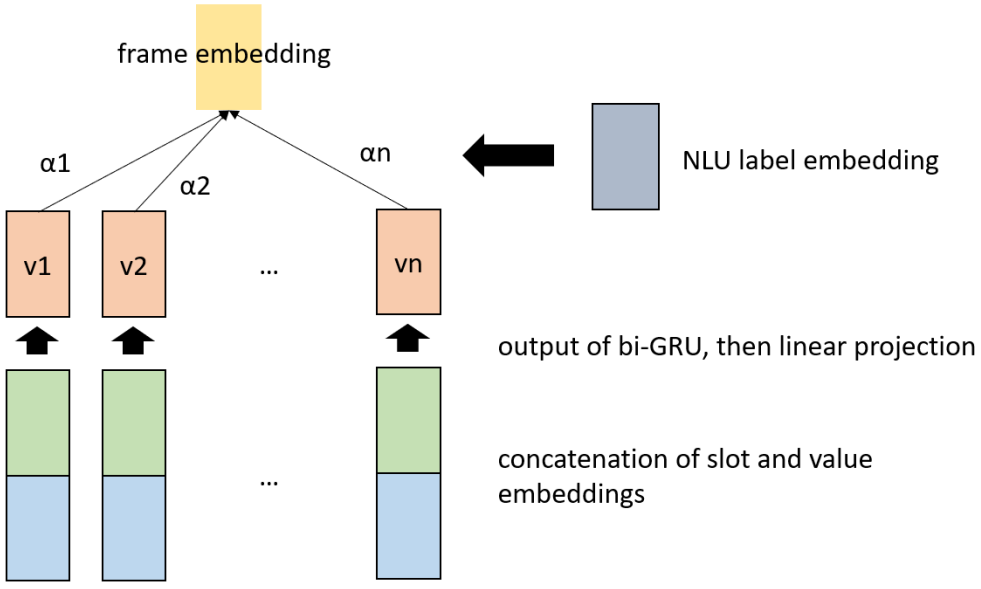

FIGURE 3.5: Architecture of frame encoder.

TABLE 3.1: Formulas of attention mechanisms. x is the embedding of a NLU label, y is an the embedding of a slot-value pair in a frame, d_y is the dimension of y, and d_k is the dimension of $W_k y$. w, W, W_k, W_q, and b are trainable parameters.

Type	Formula for α (attention score)
Content-based	$\tanh(wy + b)$
Dot-product	$x^T y$
Cosine	$x^T y \ / \ \|x\|\|y\|$
General	$x^T W y$
Query-key	$(W_q x)^T W_k y$
Scaled dot-product	$x^T y / \sqrt{d_y}$
Scaled query-key	$(W_q x)^T W_k y / \sqrt{d_k}$

exception is the content-based attention, which computes the score using only the content of slot-values of the given frame.

4 Experiments

4.1 Settings

We train the proposed model using the setting described below. FRAMES dataset comes with a predefined 10-fold split. We use the first eight folds for training, the ninth fold for validation, and the tenth fold for testing. We choose Adam [13] as the optimization algorithm with 10^{-4} learning rate and 10^{-4} weight decay.

We set the batch size to one for the training because the input has nested variable-length lists, which are hard to gather into one batch. We train the model for 10 epochs because it is long enough to have training processes converge. We take the model with the highest validation score and evaluate it on the testing set.

The performance of the model is measured by accuracy. We consider only user turns for evaluation even though system turns are also used for training. The main reason is that there is no need to predict frame reference for system turn in a real-world situation. The same metric is used in [2]. For each experiment, we repeat it with five different random seeds and report the average and the variance of the results.

TABLE 4.1: Accuracy scores of baselines and the proposed model. We use dot product for the attention mechanism. The result from [2] uses 10-fold cross-validation. "Shuffle" means that we shuffle the training samples for each epoch, otherwise we use the same order as they appear in a dialogue.

Methods	No attention	Attention
Random	58.1 ± 0.22	-
Maluuba [2]	76.4 ± 4.49	-
BERT w/o shuffle	77.5 ± 0.52	81.0 ± 0.69
GRU w/o shuffle	79.3 ± 0.28	81.9 ± 1.05
BERT w/ shuffle	81.0 ± 0.73	82.3 ± 1.70
GRU w/ shuffle	79.5 ± 0.65	$\mathbf{82.8 \pm 0.52}$

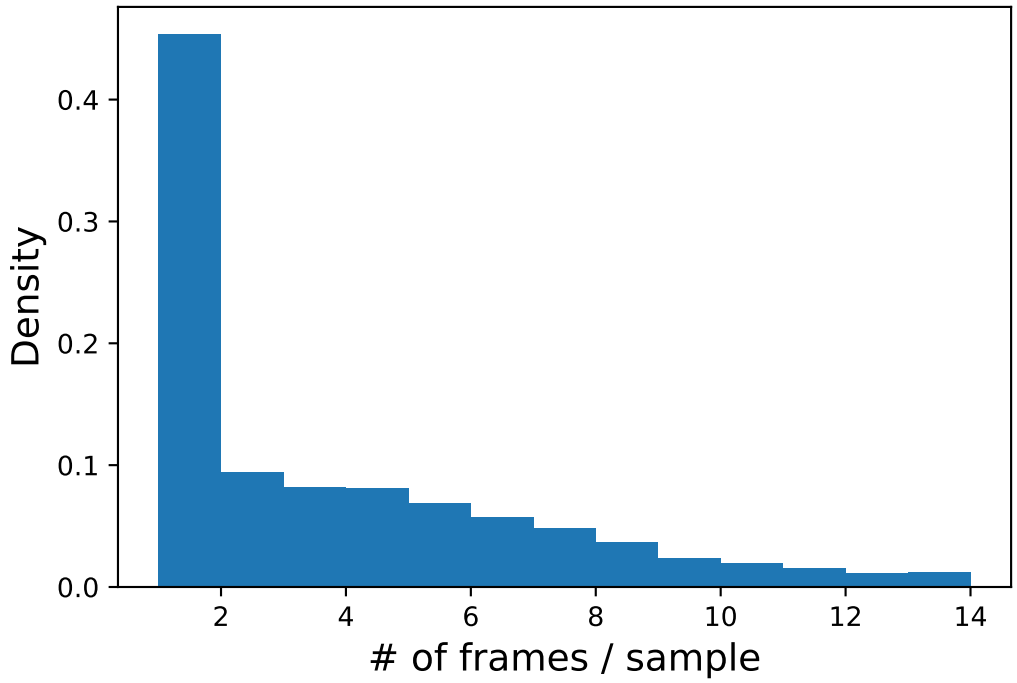

FIGURE 4.1: Distribution of number of frames.

4.2 Comparison with baselines

We compare our results with two baselines. The first one is a random baseline. It predicts a frame reference by choosing one from the list of frames uniformly at random. The other baseline is from the authors who proposed this frame tracking task [2]. Their model predicts different types of frame reference at the same time, including slot-based prediction, which is the definition used in this paper. The results are in Table 4.1.

The accuracy of the random baseline might seem unreasonably high, given that there could be several frames to choose from. However, this is not true at the beginning of a dialogue, where there are very few, or even one, frame created. As shown in Figure 4.1, about 43% of the samples has only one frame, meaning that predictions of those samples are always correct. This explains why a random model can have an accuracy of over 50%. We keep those samples for evaluation so that we can compare our results with other baselines.

For the proposed model, we can see that it consistently outperforms the two baselines regardless of the configuration. Also, adding the attention mechanism always improves the accuracy by at least one percentage point. This demonstrates the efficacy of attention mechanisms in the frame tracking task. Another important point is the effect of shuffling training samples. At first, we train the model without shuffling, i.e. using the same order as they appear in dialogues. However, even with the

TABLE 4.2: Accuracy scores of different attention mechanisms.

Attention mechanisms	GRU	BERT
No attention	79.5 ± 0.65	81.0 ± 0.73
Content-based	79.6 ± 1.09	81.9 ± 0.77
Cosine	80.7 ± 0.33	81.3 ± 0.93
Dot product	$\mathbf{82.8 \pm 0.52}$	82.3 ± 1.70
Scaled dot product	81.9 ± 1.02	$\mathbf{83.0 \pm 0.38}$
General	82.4 ± 0.80	82.4 ± 0.11
Query-key	82.6 ± 0.81	82.7 ± 0.56
Scaled query-key	81.3 ± 0.55	82.5 ± 1.70

TABLE 4.3: Results of hyperparameter search.

	GRU	BERT
Accuracy	82.9 ± 1.10	83.1 ± 0.48
Attention	Query-key	Scaled query-key
Input dim.	128	256
Hidden dim.	256	128
Embed dim.	64	64
Attention dim.	64	32

intrinsic order relation between samples, the training still benefits a lot from shuffling, leading to the best result in Table 4.1.

4.3 Attention mechanisms

We further investigate different attention mechanisms. The definition of each mechanism is in Table 3.1 and the results are shown in Table 4.2.

First, we notice that all attention mechanisms outperform our no attention baseline model, strengthen our claim about the advantage of attention in frame tracking. For the comparisons between attention mechanisms, we can see that the accuracy scores of content-based attention are rather low and are even worse than most attention mechanisms. This suggests the importance of having attention scores depending on NLU label embeddings. On the other hand, there is no clear winner among these attention mechanisms: scaled dot product obtains the best result in the table, but its accuracy is only moderate with LSTM as text embedding. Nonetheless, the results of dot product, general and query-key are good for both text embeddings.

TABLE 4.4: Accuracy scores with transfer learning. Attention is dot product. The best results in each column are marked bold.

Pre-training set	GRU w/o attention	GRU w/ attention	BERT w/o attention	BERT w/ attention
Syn. 1	77.7 ± 0.64	81.0 ± 0.26	79.8 ± 1.37	82.0 ± 0.97
Syn. 2	76.9 ± 1.21	80.4 ± 0.64	79.8 ± 0.75	80.5 ± 0.91
Syn. 3	76.9 ± 0.42	80.0 ± 0.43	79.7 ± 1.14	$\mathbf{82.4 \pm 0.42}$
Syn. 1 + 2	78.2 ± 1.24	$\mathbf{81.4 \pm 0.63}$	80.2 ± 0.53	82.0 ± 1.80
Syn. 1 + 3	77.7 ± 0.73	$\mathbf{81.4 \pm 0.53}$	79.8 ± 0.41	82.3 ± 0.93
Syn. 1 + 2 + 3	$\mathbf{78.9 \pm 0.48}$	81.1 ± 0.25	$\mathbf{80.5 \pm 0.52}$	82.0 ± 0.60

4.4 Hyperparameter tuning

In this section, we do a hyperparameter search to see how far we can push the accuracy. We use a Python library called hyperopt [14]. It searches the hyperparameter space using Tree-structured Parzen Estimator (TPE), which can find good hyperparameters with fewer trials comparing to random or grid search.

The hyperparameters we tune include attention mechanisms and dimensions in the model. For the dimensions, we put them into three groups: input, hidden, and embedding. The input dimension is the output dimension of text embedding, slot embedding, and act embedding. The hidden dimension is the dimension of the two GRUs in the NLU label encoder and the frame encoder. The embedding dimension is the output dimension of the two encoders. The options for these dimensions are 64, 128, 256, 512. For attention mechanisms, we consider no attention and all the seven mechanisms in Table 3.1. Additionally, there are two query-key attention with different attention dimensions, i.e. the output dimension of linear transforms W_q and W_k: one is 32 and the other is 64. So there are in total $4^3 \times 9 = 576$ points in this hyperparameter space. The dimension we use in previous results is 128 for input, hidden and embedding dimension, and 64 for the query-key attention dimension.

We do a hyperparameter search for both GRU and BERT text embeddings. The results are in Table 4.3. For both text embeddings, the accuracy scores are improved by 0.1 percentage point. Besides, the results show that the query-key and scaled query-key attention mechanisms are the best among all attention mechanisms. This is consistent with the observation in Section 4.3.

4.5 Transfer learning

In this section, we study how transfer learning can help in frame tracking. We use the large synthetic datasets described in Section 2.2 as pre-training datasets and fine-tune the pre-trained model on FRAMES dataset. We use 80% of the pre-training dataset for training and the rest for validation. The model is trained for 20 epochs

and the one having the best validation score is used as the starting point of fine-tuning. The fine-tuning process is the same as in Section 4.1.

We experiment with different combinations of synthetic datasets. The description of the datasets is in Table 2.7. The result is in Table 4.4. We see again that using attention improves accuracy. As for the comparison between pre-training sets, it is hard to conclude which one is better in terms of accuracy. The best result after fine-tuning is slightly worse than that of training from scratch (cf. Table 4.1). However, if we look at the learning curves in Figure 4.2, we see that the pre-trained models always start with higher accuracy, showing the advantage of pre-training at the beginning of fine-tuning. Furthermore, they also converge faster. After a few epochs, models training from scratch overtake pre-trained models.

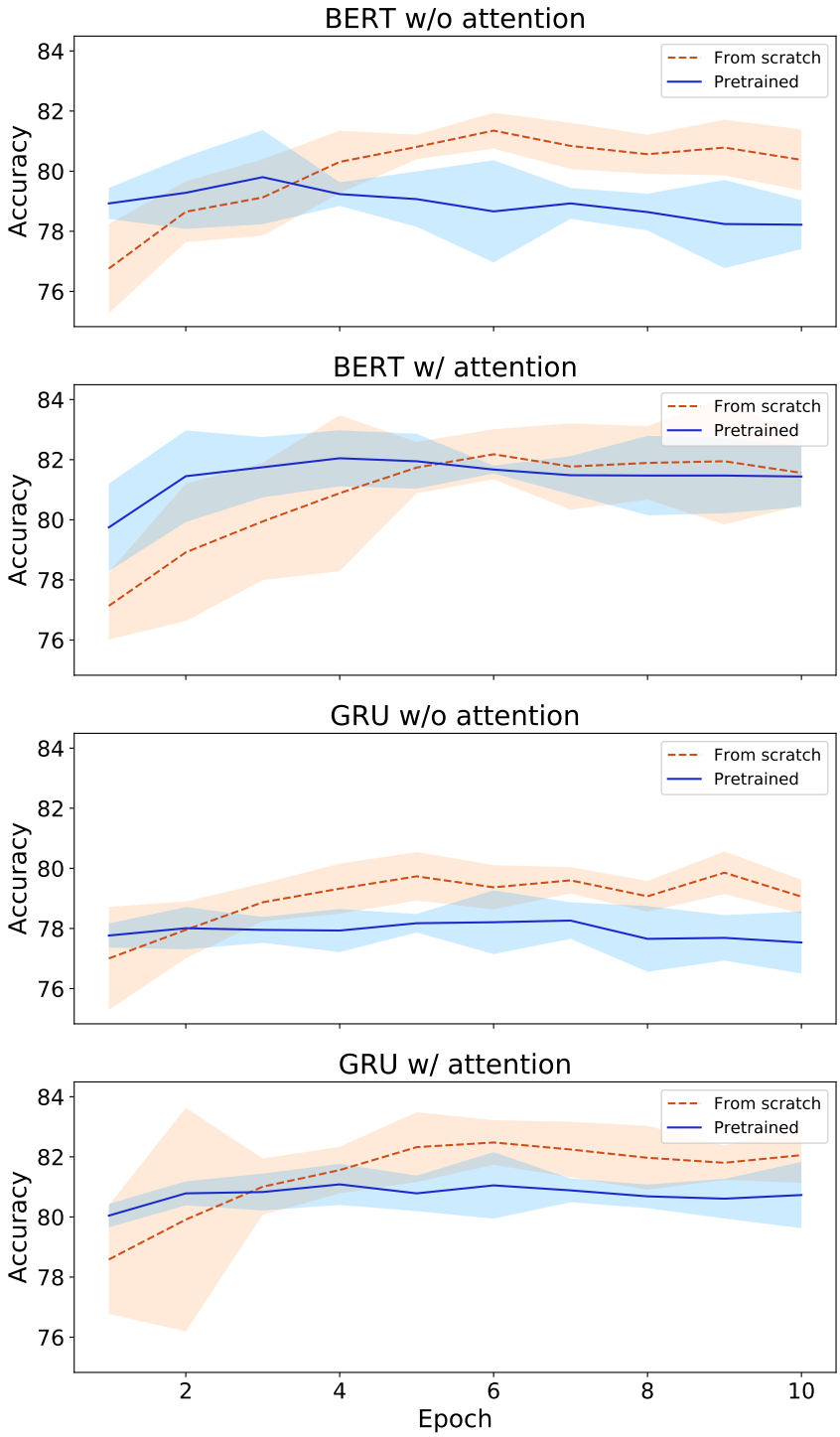

FIGURE 4.2: Learning curves of transfer learning. The pre-traning set is synthetic 1.

5 Discussion

5.1 Improve after deployment

We propose a framework in this section to discuss the possibility of improving a frame tracking model after its deployment. A deployed dialogue system has the chance of processing through a large number of real dialogues. It would be a huge advantage if we can make use of those data to improve the system. A similar idea is discussed in [15], where the authors proposed a self-feeding chit-chat dialogue system. In a chit-chat dialogue system, the roles of the user and the system are symmetric. This is not true for a goal-oriented dialogue system, so we adapt their method to fit into our setting.

The main idea is to obtain user feedback as frame tracking labels of the corresponding dialogues. There are many ways to achieve this. Here we design a framework that collects active user-feedback while still being user friendly and taking care of user experience. Due to the time constraint of this project, we don't implement and experiment with this framework.

5.1.1 Proposed framework

Our system asks for feedback only when the estimated user satisfaction is low, in other words, when it thinks it made a mistake (in the previous turn). Comparing to asking for feedback at every turn, this should reduce the extra effort imposed on a normal user of the system. When asking for feedback, the system provides a guess at the same time to be helpful and offer a better user experience. The user feedback is then transformed into training data and labels.

Ideally, in the context of frame tracking, the user feedback would tell us the correct frame reference. This is however infeasible because the concept of frame reference is too complicated to incorporate into a normal dialogue. Instead, correcting or confirming a slot value is more reasonable in terms of the amount of work imposed on the user. So when asking for feedback, the system would pick a slot in the previous turn and ask for correction. This operation is better to be done using natural language rather than an abrupt multiple choice question or a list of options so that the data collection part and the original dialogue are seamlessly combined.

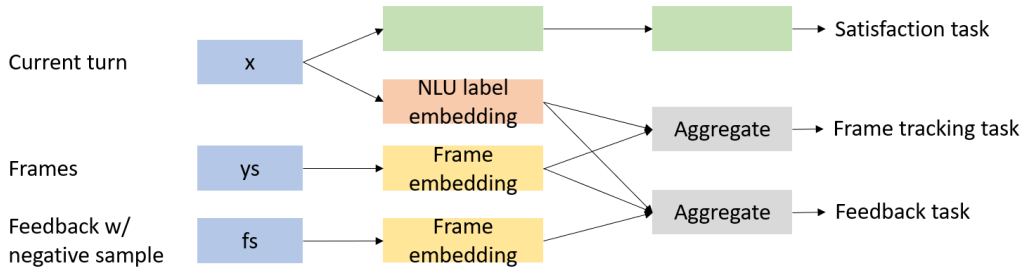

FIGURE 5.1: Example model for frame tracking after deployment.

This system consists of three tasks. The first one is the frame tracking task. The second one is the satisfaction task, which is used to decide when to ask for feedback. The input of the satisfaction task is the user utterance and the output is a binary class indicating whether the system should ask for feedback or not. The last and most important one is the feedback task. The label we have for this task is a slot value in the referred frame, so this is the target we want to predict rather than the frame reference itself as in frame tracking task. However, we still want to make it as similar to frame tracking as possible so that the system can improve on frame tracking when training on feedback data. To account for this, we define the feedback task as follows. The input consists of two parts: the same input as of frame tracking, i.e. NLU labels of the current utterance and a list of frames, and the other part is a candidate slot value. The goal of this task is to predict whether the candidate slot value is a correct one, in other words, whether this slot value appears in the referred frame.

Inspired by [15], we propose to combine these three tasks using multitask learning, hoping that training on new feedback data gathered after deployment would improve the performance of frame tracking. Figure 5.1 is an example of a multitask learning model.

Here we give a procedure of how to create the whole system from scratch.

1. Train on Frame tracking task
2. Collect data for Satisfaction task
3. Train on Frame tracking + Satisfaction
4. Deploy and collect data for Feedback task
5. Train on Frame tracking + Satisfaction + Feedback
6. Back to step 4.

6 Conclusion

In this paper, we study how to model memory in a dialogue system using frame tracking. We propose a new model using attention mechanisms to mimic the way humans approach this problem. We evaluate several types of attention mechanisms using the FRAMES dataset. The results give strong evidence of the efficacy of attention, and also show that query-key and scaled query-key are the best mechanisms among the ones we have tried. In addition to the new model, we propose a method to generate synthetic frame tracking data from ordinary dialogues. Experimens show that the training converges faster with model pre-trained on synthetic datasets. As a final touch, we propose a framework for improving a frame tracking model after its deployment.

Bibliography

[1] L. E. Asri, H. Schulz, S. Sharma, J. Zumer, J. Harris, E. Fine, R. Mehrotra, and K. Suleman, "Frames: A corpus for adding memory to goal-oriented dialogue systems", *arXiv preprint arXiv:1704.00057*, 2017.

[2] H. Schulz, J. Zumer, L. E. Asri, and S. Sharma, "A frame tracking model for memory-enhanced dialogue systems", *arXiv preprint arXiv:1706.01690*, 2017.

[3] Y.-N. Chen, D. Hakkani-Tür, G. Tür, J. Gao, and L. Deng, "End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding.", in *Interspeech*, 2016, pp. 3245–3249.

[4] S. Lee and A. Stent, "Task lineages: Dialog state tracking for flexible interaction", in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2016, pp. 11–21.

[5] S. Sharma, P. K. Choubey, and R. Huang, "Improving dialogue state tracking by discerning the relevant context", *arXiv preprint arXiv:1904.02800*, 2019.

[6] J. Perez and F. Liu, "Dialog state tracking, a machine reading approach using memory network", *arXiv preprint arXiv:1606.04052*, 2016.

[7] J. Weston, S. Chopra, and A. Bordes, "Memory networks", *arXiv preprint arXiv:1410.3916*, 2014.

[8] J. F. Kelley, "An iterative design methodology for user-friendly natural language office information applications", *ACM Transactions on Information Systems (TOIS)*, vol. 2, no. 1, pp. 26–41, 1984.

[9] V. Rieser, I. Kruijff-Korbayová, and O. Lemon, "A corpus collection and annotation framework for learning multimodal clarification strategies", in *6th SIGdial Workshop on DISCOURSE and DIALOGUE*, 2005.

[10] T.-H. Wen, D. Vandyke, N. Mrksic, M. Gasic, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, and S. Young, "A network-based end-to-end trainable task-oriented dialogue system", *arXiv preprint arXiv:1604.04562*, 2016.

[11] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić, "MultiWOZ - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling", *arXiv preprint arXiv:1810.00278*, 2018.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[14] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms", in *Proceedings of the 12th Python in science conference*, Citeseer, 2013, pp. 13–20.

[15] B. Hancock, A. Bordes, P.-E. Mazare, and J. Weston, "Learning from dialogue after deployment: Feed yourself, chatbot!", *arXiv preprint arXiv:1901.05415*, 2019.