Multi-turn Response Selection using Dialogue Dependency Relations

Qi Jia[1] **Yizhu Liu**[2] **Siyu Ren**[3] **Kenny Q. Zhu**[4*]
Shanghai Jiao Tong University
Shanghai, China
{[1]Jia_qi,[2]liuyizhu,[3]roy0702}@sjtu.edu.cn [4]kzhu@cs.sjtu.edu.cn

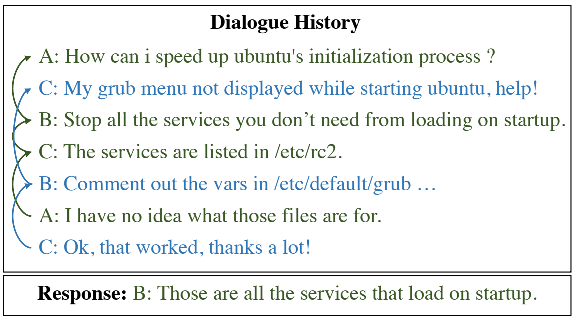
Haifeng Tang
China Merchants Bank Credit Card Center
Shanghai, China
thfeng@cmbchina.com

Abstract

Multi-turn response selection is a task designed for developing dialogue agents. The performance on this task has a remarkable improvement with pre-trained language models. However, these models simply concatenate the turns in dialogue history as the input and largely ignore the dependencies between the turns. In this paper, we propose a dialogue extraction algorithm to transform a dialogue history into threads based on their dependency relations. Each thread can be regarded as a self-contained sub-dialogue. We also propose Thread-Encoder model to encode threads and candidates into compact representations by pre-trained Transformers and finally get the matching score through an attention layer. The experiments show that dependency relations are helpful for dialogue context understanding, and our model outperforms the state-of-the-art baselines on both DSTC7 and DSTC8*, with competitive results on UbuntuV2.

1 Introduction

Dialogue system is an important interface between machine and human. An intelligent dialogue agent is not only required to give the appropriate response based on the current utterance from the user, but also consider the dialogue history. Dialogue context modeling has been a key point for developing such dialogue systems, including researches on state tracking (Eric et al., 2019; Ren et al., 2019), topic segmentation (Nan et al., 2019; Kim, 2019), multi-turn response selection (Tao et al., 2019; Gu et al., 2019), next utterance generation (Zhang et al., 2019; Chen et al., 2019), etc. In this paper, we target on the multi-turn response selection task, which is first proposed by Lowe et al. (2015) and is also a track in both DSTC7 (Gunasekara et al., 2019) and DSTC8 (Kim et al., 2019).



Dialogue History

A: How can i speed up ubuntu's initialization process ?
C: My grub menu not displayed while starting ubuntu, help!
B: Stop all the services you don't need from loading on startup.
C: The services are listed in /etc/rc2.
B: Comment out the vars in /etc/default/grub …
A: I have no idea what those files are for.
C: Ok, that worked, thanks a lot!

Response: B: Those are all the services that load on startup.

Figure 1: An example of the tangled dialogue history. A, B and C are three participants. Texts in different colors represent different dialogue threads.

Given a dialogue history made up of more than one utterance, the selection task is to choose the most possible next utterance from a set of candidate responses. Previous work on this task can be roughly divided into two categories: sequential models and hierarchical models. The former ones, including (Lowe et al., 2015; Yan et al., 2016; Chen and Wang, 2019), concatenate the history utterances into a long sequence, try to capture the similarities between this sequence and the response and give a matching score. The latter ones, including (Tao et al., 2019; Wang et al., 2019; Gu et al., 2019), extract similarities between each history utterance and the response first. Then, the matching information is aggregated from each pair (mostly in a chronological way) to get a final score. There is little difference between the performance of these two kinds of architectures until the emergence of large pre-trained language models.

Work such as (Whang et al., 2019; Vig and Ramea, 2019) has shown the extraordinary performance of the pre-trained language models on dialogues. These pre-trained models are easily transferred to the response selection task by concatenating all of the utterances as the input. All of

[*] The corresponding author.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 1911–1920,
November 16–20, 2020. ©2020 Association for Computational Linguistics

the words in dialogue history can directly interact with each other via transformers like Bi-encoder, even the words both in the dialogue history and the candidate response if time permits, such as Cross-encoder (Humeau et al., 2019). However, since such models can be regarded as the ultimate architecture of the sequential-based models, the dialogue dependency information between the utterances is largely ignored due to the concatenation operation (Wu et al., 2017). An example is shown in Figure 1. The dependency relations can definitely help us to understand the two tangled dialogue threads. Besides, we always need to truncate the earlier dialogue history to limit the size of the model and make the computation efficient. However, it isn't always that the nearest utterances are more important. As we can see in Figure 1, several dialogue threads may be tangled especially in multi-party chat rooms, it's hard to tell which dialogue thread will be moving on.

In this paper, we propose to incorporate dialogue dependency information into the response selection task. We train a dialogue dependency parser to find the most probable parent utterance for each utterance in a session. We name such relation between utterances as "reply-to". Then, we empirically design an algorithm to extract dialogue threads, which is represented by a path of dependency relations according to the parsed trees. The extracted threads are sorted by the distance between the final utterance in each thread and the response in ascending order, following the intuition that the closer utterances are more relevant. After that, we propose the model named Thread-Encoder based on a pre-trained language model. Each encoder in the model can distill the critical information from each dialogue thread or the candidate response. Finally, another attention layer is used to calculate the matching score with thread representations and the candidate representation. The candidate with the highest matching score will be selected as the final response.

We collect the training data for dialogue dependency parser from a dialogue disentanglement dataset (Kummerfeld et al., 2019) in the technical domain. And we do response selection experiments among UbuntuV2, DSTC7 and DSTC8*. These datasets consist of dialogues in the same domain but under different settings, including two-party dialogues and multi-party dialogues. The results demonstrate our model's strong capability to represent multi-turn dialogues on all of these datasets.

Our main contributions are as follows:

- As far as we know, we are the first to incorporate dialogue dependency information into response selection task, demonstrating that the dependency relations in the dialogue history are useful in predicting dialogue responses (Sec 5).

- Based on the predicted dependencies, we design a straight-forward but effective algorithm to extract several threads from the dialogue history (Sec 2.1). The results show the algorithm is better than other simple segmentation methods on the response selection task.

- We propose the Thread-Encoder model, incorporating dialogue dependency information by threads and utilizing the pre-trained language model to generate corresponding representations (Sec 2.2). The experimental results show that our model outperforms the state-of-the-art baselines on DSTC7 and DSTC8* datasets, and is very competitive on UbuntuV2 (Sec 4).

2 Approach

The multi-turn response selection tasks represent each dialogue as a triple $T = \langle C, R, L \rangle$, where $C = \{t_1, t_2, ..., t_n\}$ represents the history turns. R is a candidate response and L is the $0/1$ label indicating whether R is the correct response or a negative candidate. To incorporate the dependency information between the history turns, we design a straight-forward algorithm to extract the dialogue history C into dialogues threads $\langle C_1, C_2, ..., C_M \rangle$ based on the predicted dependencies, along with an elaborately designed model to find the function $f(C_1, C_2, ..., C_M, R)$, which measures the matching score of each (C, R) pair. Both the extraction algorithm and the model will be explained as follows.

2.1 Dialogue Extraction Algorithm

Since it's impossible for the large pre-trained language models to take all of the dialogue history turns as the input under the computational power nowadays, these models usually set a truncate window and only consider the top-k most recent turns or tokens. However, several dialogue threads may exist concurrently in two-party (Du et al., 2017) or multi-party dialogues (Tan et al., 2019). Such coarse-grained truncating operation may not only bring in redundant turns from other dialogue threads, but also exclude the expected turns given earlier in the current dialogue thread, hurting the representation capability of pre-trained language models. Extracting the whole history into self-contained dialogue threads can help preserve more relevant turns and avoid the negative effects of encoding irrelevant turns by a single language model.

Motivated by the above, we aim to analyze the discourse structures in dialogue history at first. We utilize the discourse dependency parsing model for dialogues proposed by Shi and Huang (2019). It is a deep sequential model that achieves the state-of-the-art performance on the STAC corpus. Instead of predicting the predefined relation types between Elementary Discourse Units(EDUs), we borrow the proposed model in this work to find if there exist dependency relations between utterances in the given dialogue history. The model scans through the dialogue history and predicts the most likely parent turn for each turn. It finally constructs a dependency tree for each dialogue history with a confidence score on each edge.

Algorithm 1: The Dialogue Extraction Algorithm

```
Input : The dependency tree $T$ with confidence scores on each edge
        $e_{ji} = (t_i, t_j, P_{ji})$, where $i.j = 1, 2, ..., n$ and $j > i$;
        The threshold for the confidence score $P$.
Output: The threads $C' = \langle C_1, C_2, ..., C_M \rangle$, and each is made up
        of a sequence of turns.
for $e_{ji}$ in $T$ do
    if $P_{ji} < P$ then
        delete $e_{ji}$ from $T$
    end
end
The forest $T' = T$
The set of threads $C' = \emptyset$
for each leaf node in $T'$ do
    $C_{tmp}$ = all the node from the leaf node to the corresponding
    root.
    $C' = C' \cup C_{tmp}$
end
Rank the threads in $C'$ based on the index of the leaf node in
descending order.
```

Then, the dialogue extraction algorithm is designed to extract original long history into dialogue threads according to dependency tree T and confidence scores. The algorithm is depicted in Algorithm 1. e_{ji} is a directed edge with head t_i and tail t_j, indicating that turn j is a reply of turn i with probability P_{ji}. The threshold P is a hyper-parameter. It is noteworthy that we still follow the intuition that the turns closer to the responses are more likely to be useful than others. As a result, the threads are returned in ascending order according to the distance between the last turns in each thread and the response. An illustration of the algorithm is shown in Figure 2. The 7-turn dialogue history is extracted into three threads.

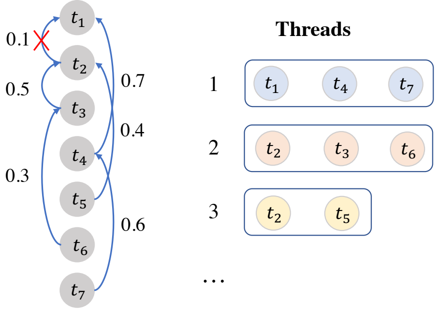


Figure 2: An example of the algorithm when the threshold $P = 0.2$. The figures are the confidence scores of corresponding predicted dependency relations.

2.2 Thread-based Encoder Model

In the work from Humeau et al. (2019), they use a pre-trained language model as the context encoder and generate the embedding for dialogue history. Inspired by this work, we also utilize pre-trained language models to encode natural texts into meaningful representations.

Given the extracted self-contained dialogue threads $\langle C_1, C_2, ..., C_M \rangle$, we utilize a pre-trained language model to encode the content of each dialogue thread in parallel and another pre-trained language model to encode the candidate respectively. If the candidate representation matches well with one or more thread representations, that candidate is probably the correct response.

The architecture of our model **Thread-Encoder** (shown in Figure 3) can be divided into two layers: Encoding Layer and Matching Layer.

2.2.1 Encoding Layer

We use the pre-trained language model released by Humeau et al. (2019). This large pre-trained Transformer model has the same architecture as BERT-base (Devlin et al., 2019). It has 12 layers, 12 attention heads and 768 hidden size. Different from the original one trained on BooksCorpus and Wikipedia, the new language model is further trained on Reddit (Henderson et al., 2019), a large dialogue dataset with around 727M context-response pairs. The pretraining tasks include masked language model and next utterance prediction [1]. Finally, the

[1] "Utterance" and "turn" are interchangeable in this paper.

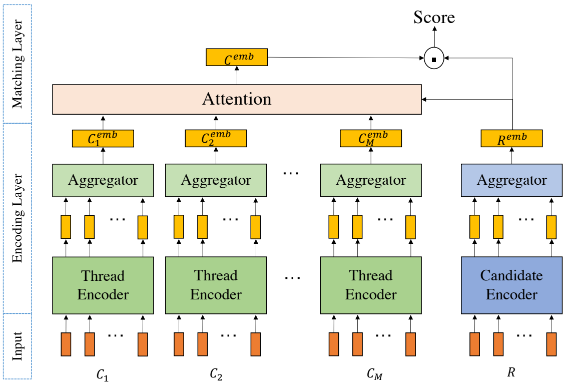


Figure 3: The architecture of Thread-Encoder model. All of the blocks in the same color share parameters.

pre-trained model can be used for a wide range of multi-sentence selection tasks with fine-tuning.

In our model, the encoder layer uses two Transformers, thread encoder $T_1(\cdot)$ and candidate encoder $T_2(\cdot)$, both initialized with the pre-trained weights. $T_1(\cdot)$ is used for encoding threads, and all of the turns in a thread are concatenated into a long sequence in reverse chronological order as the input. $T_2(\cdot)$ is used for encoding the candidate. The inputs to the Transformer encoder are surrounded by the special token $[S]$, consistent with the operations during pretraining.

Above the Transformer encoder is an aggregator $agr(\cdot)$ that aggregates a sequence of vectors produced by the encoder into one or more vectors. In a word, the threads and response can be encoded as follows:

$$\begin{aligned} C_m^{emb} &= agr_1(T_1(C_m)) \\ R^{emb} &= agr_2(T_2(R)), \end{aligned} \tag{1}$$

where $m = 1, 2, ...M$ and M is the number of dialogue threads. For $arg_1(\cdot)$, if we simply use ”average” function for the aggregator, only one representation will be encoded for each thread. We name this model as **Thread-bi**. If we use ”multi-head attention” as the aggregator, multiple representations will be encoded for each thread. We name this model as **Thread-poly**. The aggregator $agr_2(\cdot)$ for candidate representations is the average over input vectors.

2.2.2 Matching Layer

Given the encoded threads $\langle C_1^{emb}, C_2^{emb}, ..., C_M^{emb} \rangle$ and candidate R^{emb}, we further use an attention layer to distill the information from the threads by attending the query R^{emb} to each C_m^{emb}:

$$C^{emb} = \sum_{m=1}^{M} w_m C_m^{emb} \tag{2}$$

where

$$\begin{aligned} s_m &= (R^{emb})^\top \cdot C_m^{emb} \\ w_m &= \exp(s_m) / \sum_{k=1}^{M} \exp(s_k) \end{aligned} \tag{3}$$

The final matching score is given by:

$$S = F(C_1, C_2, ..., C_M, R) = (R^{emb})^\top \cdot C^{emb} \tag{4}$$

We consider the other correct responses in a mini-batch as the negative candidates to accelerate the training process (Mazaré et al., 2018). The whole model is trained to minimize the cross-entropy loss as follows:

$$loss = -\frac{1}{A} \sum_{a=1}^{A} \sum_{b=1}^{A} L_{ab} \log(S_{ab}) \tag{5}$$

where A is the batch size. L_{ab} equals 1 when $a = b$, otherwise 0. S_{ab} is the matching score in Eq. 4.

3 Experimental Setup

In this section, we introduce the datasets, baselines and implementation details of our model[2].

3.1 Datasets

Our experiments are performed on three datasets: UbuntuV2, DSTC 7 and DSTC 8*.

- **UbuntuV2** (Lowe et al., 2017) consists of two-party dialogues extracted from the Ubuntu chat logs.
- **DSCT7** (Gunasekara et al., 2019) refers to the dataset for DSTC7 subtask1 consisting of two-party dialogues.
- **DSCT8*** refers to the dataset for DSTC8 subtask 2, containing dialogues between multiple parties. We remove the samples without correct responses in the given candidate sets [3].

More details of these three datasets are in Table 1.

[2] The codes and data resources can be found in https://github.com/JiaQiSJTU/ResponseSelection.

[3] We do this to eliminate the controversy of solving no correct response in different ways and try to focus on dialogue context modeling.

	UbuntuV2	DSTC7	DSTC8*
Train	957,101	100,000	89,813
Valid	19,560	5,000	7,660
Test	18,920	1,000	7,174
#Candidates	10	10	100
#Correct Response	1	1	1
#Turns	3-19	3-75	1-99

Table 1: The statistics of the datasets used in this paper.

3.2 Baselines

We introduce several state-of-the-art baselines to compare with our results as follows.

- **DAM** (Zhou et al., 2018) is a hierarchical model based entirely on self and cross attention mechanisms.

- **ESIM-18** (Dong and Huang, 2018) and **ESIM-19** (Chen and Wang, 2019) are two sequential models, which are the modifications and extensions of the original ESIM (Chen et al., 2017) developed for natural language inference. The latter one ranked top on D-STC7.

- **IMN** (Gu et al., 2019) is a hybrid model with sequential characteristics at matching layer and hierarchical characteristics at aggregation layer.

- **Bi-Encoder** (Bi-Enc), **Poly-Encoder** (Poly-Enc) and **Cross-Encoder** (Cross-Enc) (Humeau et al., 2019) are the state-of-the-art models based on pre-trained model.

3.3 Implementation Details

According to Section 2.1, we firstly transform the dialogue disentanglement dataset (Kummerfeld et al., 2019). Turns are clustered if there exists a “reply-to” edge, and we obtain 4,444 training dialogues from the original training set and 480 test dialogues from the original valid set and test set. Only 7.3% of turns have multiple parents. Since the parsing model can only deal with dependency structure with a single parent, we reserve the dependency relation with the nearest parent in these cases. We trained a new parser on this new dataset. The results on the new test set are shown in Table 2. It shows that in-domain data are useful for enhancing the results for dialogue dependency prediction.

	Precision	Recall	F1
Trained on STAC	67.37	64.43	65.86
Trained on the new dataset	71.44	68.32	69.85

Table 2: The results of the dialogue dependency parser.

For the response selection task, we implemented our experiments based on ParlAI [4]. Our model is trained with Adamax optimizer. The initial learning rate and learning rate decay are $5e-5$ and 0.4 respectively. The candidate responses are truncated at 72 tokens, covering more than 99% of them. The last 360 tokens in the concatenated sequence of each thread are reserved. The BPE tokenizer was used. We set the batch size as 32. The model is evaluated on valid set every 0.5 epoch. The training process terminates when the learning rate is 0 or the hits@1 on validation no longer increases within 1.5 epochs. The threshold in the Algorithm 1 is set to 0.2 and we preserve at most top-4 threads for each sample, avoiding the meaningless single turns while ensuring the coverage of original dialogue contexts. The results are averaged over three runs. For UbuntuV2 and DSTC7 training set, we do data augmentation: each utterance of a sample can be regarded as a potential response and the utterances in the front can be regarded as the corresponding dialogue context.

Our experiments were carried out on 1 to 4 N-vidia Telsa V100 32G GPU cards. The evaluation metrics for response selection are hits@k and M-RR, which are widely used and the codes can be found in ParlAI.

4 Results and Analysis

Here we show results on dialogue thread extraction and response selection of the three datasets, and give some discussions on our model design.

4.1 Extraction Results

We first evaluate the extraction results on Ubuntu-V2, DSTC7 and DSTC8* with three metrics: The average number of threads (**avg#thd**) is to show how many dialogue threads are discovered in each dialogue, which ranges from 1 to 4. We didn’t take all of the extracted threads into consideration, serving as a hard cut for the trade-off between information loss and memory usage of the model. The average number of turns in each thread (**avg#turn**) and the average standard deviation of the number

[4] https://github.com/facebookresearch/ParlAI

Dataset		avg#thd	avg#turn	std#turn	1-thd(%)	2-thd(%)	3-thd(%)	4-thd(%)
UbuntuV2	train	1.42	3.24	0.29	68.92	22.19	6.65	2.24
	valid	1.39	3.09	0.25	70.78	20.90	6.47	1.85
	test	1.39	3.13	0.25	70.69	20.89	6.18	2.23
DSTC7	train	1.40	4.45	0.38	67.75	25.37	5.48	1.40
	valid	1.39	4.42	0.37	67.76	25.74	5.44	1.06
	test	1.45	4.42	0.42	65.00	26.10	7.70	1.20
DSTC8*	train	3.82	24.70	5.39	2.96	2.59	3.85	90.61
	valid	3.80	24.46	5.37	3.32	3.09	3.64	89.95
	test	3.81	24.53	5.34	3.09	2.76	4.06	90.09

Table 3: Statistics on extraction results. avg#thd refers to the average number of threads per dialogue, avg#turn refers to the average number of turns in each thread, and std#turn refers to the average standard deviation of the number of turns in each thread per dialogue. 1-thd to 4-thd refers to the percentage of the number of dialogues with 1 to 4 threads in corresponding datasets.

of turns in each thread (**std#turn**) are to measure the length of each thread. Dialogues context is not well separated if the length of each thread varies a lot (i.e., the std#turn is too high).

We apply the dialogue extraction algorithm in Section 2.1 on the three datasets. The statistics of extracted threads are in Table 3. Firstly, we can find that the average number of threads is around 3.81 for DSTC8* dataset while around 1.40 for the other two datasets, which well aligns with the empirical observation that two-party dialogues tend to have more concentrated discussions with a smaller number of threads while multi-party dialogues usually contain more threads to accommodate conversation with high diversity. Also, as is listed in Table 1, the number of turns for DSTC8* dataset is usually larger than UbuntuV2 and DSTC7 dataset, which naturally leads to more leaf nodes hence a larger number of threads. Secondly, the average length of threads is around 24.50 for DSTC8* dataset while around 4.0 for DSTC7 dataset and UbuntuV2 and the standard deviation for DSTC8* dataset is also larger. It shows that when the number of dialogue threads increases, the standard deviation of the length of each thread also tends to increase since some dialogue threads may catch more attentions while others may be ignored. In summary, DSTC8* is a more challenging multi-party dialogue dataset for dialogue context modeling than two-party dialogue datasets, including UbuntuV2 and DSTC7.

4.2 Response Selection Results

The response selection results of our Thread-Encoder models, including Thread-bi and Thread-poly, are shown in Table 4 for UbuntuV2 and D-STC7 datasets, and in Table 6 for DSTC8*.

Since UbuntuV2 is too large, we only fine-tuned on this dataset for three epochs due to limited computing resources. The performance of our model is similar to Bi-Enc and Poly-Enc on UbuntuV2. Although the Cross-Enc rank top on UbuntuV2, it is too time-consuming and not practical (Humeau et al., 2019). It runs over 150 times slower than both Bi-Enc and Poly-Enc. Our model, Thread-bi, takes the top four threads (see Section 4.3.2 for more details) into consideration with the inference time overhead similar to Bi-Enc and Poly-Enc. Besides, the reason why our model seems slightly worse than Poly-Enc is that UbuntuV2 is an easier dataset with fewer turns and threads according to Table 1 and Table 3. Consequently, our model degenerates towards Bi-Enc and Poly-Enc, and all four models (Bi-Enc, Poly-Enc, Thread-bi, Thread-poly) actually yield similar results, with p-value greater than 0.05.

Due to the huge advancement of pre-trained models over other models shown on UbuntuV2 and DSTC7, we mainly compared the competitive state-of-the-art pre-trained models on DSTC8* dataset for through comparison as shown in Table 6. Our models achieve the new state-of-the-art results on both DSTC7 and DSTC8* dataset proving that threads based on dependency relation between turns are helpful for dialogue context modeling. We can see that using multiple vectors works much better than using only one representation. The gap between these two aggregation methods is not clear on UbuntuV2 and DSTC7, but much more significant on DSTC8* where the dialogues between multiple participants are much more complicated. This finding hasn't been shown in Humeau's work (2019). Besides, our model can enhance both kinds of pre-trained dialogue models on the multi-

	UbuntuV2				DSCT7			
Model	hits@1	hits@2	hits@5	MRR	hits@1	hits@10	hits@50	MRR
DAM	-	-	-	-	34.7	66.3	-	35.6
ESIM-18	73.4	85.4	96.7	83.1	50.1	78.3	95.4	59.3
ESIM-19	73.4	86.6	97.4	83.5	64.5	90.2	**99.4**	73.5
IMN	77.1	88.6	97.9	-	-	-	-	-
Bi-Enc	83.6	-	98.8	90.1	70.9	90.6	-	78.1
Poly-Enc	83.9	-	98.8	90.3	70.9	91.5	-	78.0
Cross-Enc	**86.5**	-	**99.1**	**91.9**	71.7	92.4	-	79.0
Thread-bi	83.8	92.4	98.5	90.0	**73.3***	92.5	99.3	80.2*
Thread-poly	83.6	**92.5**	98.5	90.0	73.2*	**93.6***	99.1	**80.4***

Table 4: Results on UbuntuV2 and DSTC7 dataset. Scores marked with $\star$ are statistically significantly better than the state-of-the-art with $p < 0.05$ according to t-test.

turn response selection task by comparing Thread-bi with Bi-enc and Thread-poly with Poly-enc.

It should be noted that the inherent properties of these three datasets are different according to Section 4.1. UbuntuV2 and DSTC7 datasets are dialogues between two parties, while DSTC8* dataset involves more complicated multi-party dialogue. This reveals that Thread-Encoder not only works under simple scenarios such as private chats between friends, but also acquires further enhancement under more interlaced scenarios such as chaos chat rooms.

Model	#Para	Train(h)	Test(#dialog/s)
Bi-Enc	256.08M	10.22	6.79
Poly-Enc	256.13M	12.34	4.78
Thread-bi	256.08M	16.36	4.73
Thread-poly	256.13M	17.09	4.77

Table 5: Total number of parameters, training time (h) and testing speed(#dialogues per second) on DSTC8* main models.

The number of parameters, training time and testing speed are shown in Table 5. It takes more epochs for our model to convergence, while the testing speed is similar to Poly-Enc.

4.3 Discussions on Model Design

To further understand the design of our full model, we did several ablations on DSTC8*. All of the ablation results as listed in Table 6. The descriptions and analysis are in following subsections.

4.3.1 Different ways to generate threads

We evaluate some reasonable alternative methods to extract dialogue threads from the history, i.e."Thread Type" in Table 6.

- **Full-hty** concatenate the full dialogue history in one thread. Our model degrades to Bi-Enc and Poly-Enc.
- **Dist-seg** segments the turns based on their distance to the next response. This idea is based on the intuition that the adjacent turns are possible to have strong connections. For example, if we use 4 threads, the dialogue in Figure 2 will be segmented into $\langle\langle t_6, t_7\rangle, \langle t_4, t_5\rangle, \langle t_2, t_3\rangle, \langle t_1\rangle\rangle$.
- **Dep-extr** refers to the threads extraction procedure as explained in Algorithm 1.

Comparing in group ID-$\{1, 5, 7\}$ and ID-$\{2, 11, 12\}$, we get the following observations: (1) Our extraction operations help with the response selection as both ID-5 and ID-11 have significant improvement despite the distance-based extraction method is a strong baseline. The dependency relations capture salient information in dialogue more accurately and yields better performance. (2) Segmenting dialogues simply based on distance may hurt the storyline for each sub dialogue as ID-7 is worse than ID-$\{1, 5\}$, which hurts the representation ability of language models. (3) The information loss caused by Dist-seg can be partially made up by "poly" settings as ID-12 lies between ID-2 and ID-11. Generating multiple representations by aggregators may help to get multiple focuses in each thread. Thus interleaved sub-dialogues can be captured more or less. The gap between Dist-seg and Dep-extr will definitely be widened by improving the performance of sub-dialogue extraction.

4.3.2 The number of threads to use

After deciding the way for extraction, the number of threads (i.e., #Thread in Table 6) to use is another key hyper-parameter for this model design.

We tested our model using the number of threads ranging from 1 to 4. The results are shown in

ID	Method	Aggregation Type	Thread Type	#Thread	hits@1	hits@5	hits@10	hits@50	MRR
1	Bi-Enc	Average	Full-hty	1	22.2	43.0	54.2	88.7	32.9
2	Poly-Enc	Attention	Full-hty	1	32.5	54.1	64.4	91.4	43.1
3	Thread-bi	Average	Dep-extr	1	20.2	39.6	51.1	86.1	30.5
4	Thread-bi	Average	Dep-extr	2	22.6	42.5	53.1	87.9	32.9
5	Thread-bi	Average	Dep-extr	3	23.4	43.0	54.9	88.2	33.8
6	Thread-bi	Average	Dep-extr	4	22.9	43.3	55.1	88.5	33.5
7	Thread-bi	Average	Dist-seg	3	21.7	43.2	55.2	88.8	32.8
8	Thread-poly	Attention	Dep-extr	1	29.4	49.1	59.4	88.7	39.5
9	Thread-poly	Attention	Dep-extr	2	32.0	53.2	63.2	91.1	42.5
10	Thread-poly	Attention	Dep-extr	3	33.1	54.1	64.2	92.0	43.5
11	Thread-poly	Attention	Dep-extr	4	**33.5***	**54.5***	**64.5**	91.7	**44.0***
12	Thread-poly	Attention	Dist-seg	4	33.2	53.5	63.6	**92.3***	43.4

Table 6: Main results of DSTC8* (underlined) and ablation tests on DSTC8*. Scores marked with * are statistically significantly better than Poly-Enc with $p < 0.05$ according to t-test.

ID-$\{3 \sim 6\}$ and ID-$\{8 \sim 11\}$ from Table 6, we draw following conclusions. First, by comparing the results with only 1 thread, we can see ID-3 and ID-8 are worse than Bi-enc and Poly-enc respectively. It shows that there does exist many cases that correct candidates that do not respond to the nearest dialogue threads. Considering only the nearest sub-dialogue is not enough. Second, with the increasing number of threads from 1 to 4, the results go up and down for Thread-bi. The peak value is achieved when #Thread equals 3. Although more than 90% of dialogues can be extracted into 4 threads according to Table 3, the results doesn't go up with one more thread. Some redundant dialogue threads far from the current utterances may bring noises for response selection. Also, the negative effects of redundant dialogue threads for Thread-poly reflect on the limited improvements and even decreases on hits@50 between ID-10 and ID-11. Designing a metric to filter the extracted dialogue threads automatically is our future work.

5 Related Work

Related work contains dialogue dependency parsing and multi-turn response selection.

5.1 Dialogue dependency parsing

Discourse parsing has been researched by scientists especially in linguistics for decades. Asher and Lascarides (2005) proposed the SDRT theory with the STAC Corpus (Asher et al., 2016) which made a great contribution to the discourse parsing on multi-party dialogues. Shi and Huang (2019) proposed a sequential neural network and achieved the state-of-the-art results on this dataset. Another similar task is dialogue disentanglement (Du et al., 2017). This task isn't focusing on developing discourse theories but trying to segment the long dialogues according to topics. It takes each turn in the dialogue as a unit, and only care about whether there is a relation between two turns, which is called "reply-to" relation. Due to the scarcity of annotated dialogues across domains under SDRT theory, the predicted dependency relations had never been used for down-streaming tasks, such as response selection and dialogue summarization. In this paper, we take advantage of both the simplicity of the "reply-to" relation and the sequential parsing methods (Shi and Huang, 2019) to do dialogue dependency parsing. Developing general discourse parsing with relations types and take relation types into consideration may be future work.

5.2 Multi-turn response selection

Multi-turn response selection task was proposed by Lowe et al. (2015) and the solutions for this task can be classified into two categories: the sequential models and the hierarchical models. To begin with, the sequential models (Lowe et al., 2015) were directly copied from the single-turn response selection task since we can regard the multiple history turns as a long single turn. Considering the multi-turn characteristic, Wu et al. (2017) proposed the sequential matching network (SMN), a new architecture to capture the relationship among turns and important contextual information. SMN beats the previous sequential models and raises a popularity of such hierarchical models, including DUA (Zhang et al., 2018), DAM (Zhou et al., 2018), IOI (Tao et al., 2019), etc. The ESIM (Dong and Huang, 2018), which is mainly based on the self and cross attention mechanisms and incorporates different kinds of pre-trained word embedding. It changed the inferior position of the sequential model, making it hard to say which kind of architecture is better.

Due to the popularity of the pre-trained language models such as BERT (Devlin et al., 2019) and GPT (Radford et al., 2018), the state-of-the-art performance on this task was refreshed (Vig and Ramea, 2019). Work such as (Whang et al., 2019) and (Humeau et al., 2019) further shows that the response selection performance can be enhanced by further pretraining the language models on open domain dialogues such as Reddit (Henderson et al., 2019), instead of single text corpus such as BooksCorpus (Zhu et al., 2015). These models can be also regarded as the sequential models because they concatenate all the history turns as the input to the model while ignoring the dependency relations among the turns. Inspired by these works, we incorporate the dependency information in the dialogue history into the response selection model with the pre-trained language model on dialogue dataset.

In this work, we focus on the effectiveness of exploiting dependency information for dialogue context modeling and follow the data preprocessing steps in two-party dialogue datasets, including UbuntuV2 and DSTC7, which have no special designs for speaker IDs. In the papers for DSTC8 response selection track, such as (Gu et al., 2020), many heuristic rules based on speaker IDs are used for data preprocessing, which greatly helps to filter out unrelated utterances. However, they also definitely lead to losing some useful utterances. These hard rules will hurt the completeness of the meaning in each thread and are not suitable for us. As a result, the results on the response selection task for DSTC8 dataset are not comparable. We will take advantage of the speaker information into both extraction and dialogue understanding models as our future work.

6 Conclusion

As far as we know, we are the first work bringing the dependency information of dialogues into the multi-turn response selection task. We proposed the dialogue extraction algorithm and Thread-Encoder model, which becomes the state-of-the-art on several well-known ubuntu datasets. In the future, we will move on to develop a more general dialogue dependency parser and better incorporate dependency information into dialogue context modeling tasks.

Acknowledgement

This research was supported by the SJTU-CMBCC Joint Research Scheme, SJTU Medicine-Engineering Cross-disciplinary Research Scheme, and NSFC grant 91646205.

References

Nicholas Asher, Julie Hunter, Mathieu Morey, Farah Benamara, and Stergos D. Afantenos. 2016. Discourse structure and dialogue acts in multiparty dialogue: the STAC corpus. In *LREC*.

Nicholas Asher and Alex Lascarides. 2005. *Logics of Conversation*. Studies in natural language processing. Cambridge University Press.

Qian Chen and Wen Wang. 2019. Sequential attention-based network for noetic end-to-end response selection. *CoRR*, abs/1901.02609.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *ACL*, pages 1657–1668.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *ACL*, pages 3696–3709.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Jianxiong Dong and Jim Huang. 2018. Enhance word representation for out-of-vocabulary on ubuntu dialogue corpus. *CoRR*, abs/1802.02614.

Wenchao Du, Pascal Poupart, and Wei Xu. 2017. Discovering conversational dependencies between messages in dialogs. In *AAAI*, pages 4917–4918.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tür. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *CoRR*, abs/1907.01669.

Jia-Chen Gu, Tianda Li, Quan Liu, Xiaodan Zhu, Zhen-Hua Ling, and Yu-Ping Ruan. 2020. Pre-trained and attention-based neural networks for building noetic task-oriented dialogue systems. *CoRR*, abs/2004.01940.

Jia-Chen Gu, Zhen-Hua Ling, and Quan Liu. 2019. Interactive matching network for multi-turn response selection in retrieval-based chatbots. In *CIKM 2019*, pages 2321–2324.

Chulaka Gunasekara, Jonathan K Kummerfeld, Lazaros Polymenakos, and Walter Lasecki. 2019. Dstc7 task 1: Noetic end-to-end response selection. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 60–67.

Matthew Henderson, Pawel Budzianowski, Iñigo Casanueva, Sam Coope, Daniela Gerz, Girish Kumar, Nikola Mrksic, Georgios Spithourakis, Pei-Hao Su, Ivan Vulic, and Tsung-Hsien Wen. 2019. A repository of conversational datasets. *CoRR*, abs/1904.06472.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *CoRR abs/1905.01969*, 2:2–2.

Seokhwan Kim. 2019. Dynamic memory networks for dialogue topic tracking.

Seokhwan Kim, Michel Galley, R. Chulaka Gunasekara, Sungjin Lee, Adam Atkinson, Baolin Peng, Hannes Schulz, Jianfeng Gao, Jinchao Li, Mahmoud Adada, Minlie Huang, Luis Lastras, Jonathan K. Kummerfeld, Walter S. Lasecki, Chiori Hori, Anoop Cherian, Tim K. Marks, Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, and Raghav Gupta. 2019. The eighth dialog system technology challenge. *CoRR*, abs/1911.06394.

Jonathan K. Kummerfeld, Sai R. Gouravajhala, Joseph Peper, Vignesh Athreya, R. Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros C. Polymenakos, and Walter S. Lasecki. 2019. A large-scale corpus for conversation disentanglement. In *ACL*, pages 3846–3856.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *SIGDIAL*, pages 285–294.

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *D&D*, 8(1):31–65.

Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training millions of personalized dialogue agents. In *EMNLP*, pages 2775–2779.

Feng Nan, Ran Ding, Ramesh Nallapati, and Bing Xiang. 2019. Topic modeling with wasserstein autoencoders. In *ACL*, pages 6345–6381.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Liliang Ren, Jianmo Ni, and Julian J. McAuley. 2019. Scalable and accurate dialogue state tracking via a hierarchical sequence generation. In *EMNLP-IJCNLP*, pages 1876–1885.

Zhouxing Shi and Minlie Huang. 2019. A deep sequential model for discourse parsing on multi-party dialogues. In *AAAI*, pages 7007–7014.

Ming Tan, Dakuo Wang, Yupeng Gao, Haoyu Wang, Saloni Potdar, Xiaoxiao Guo, Shiyu Chang, and Mo Yu. 2019. Context-aware conversation thread detection in multi-party chat. In *EMNLP-IJCNLP*, pages 6455–6460.

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019. One time of interaction may not be enough: Go deep with an interaction-over-interaction network for response selection in dialogues. In *ACL*, pages 1–11.

Jesse Vig and Kalai Ramea. 2019. Comparison of transfer-learning approaches for response selection in multi-turn conversations. In *Workshop on DSTC7*.

Heyuan Wang, Ziyi Wu, and Junyu Chen. 2019. Multi-turn response selection in retrieval-based chatbots with iterated attentive convolution matching network. In *CIKM*, pages 1081–1090.

Taesun Whang, Dongyub Lee, Chanhee Lee, Kisu Yang, Dongsuk Oh, and Heuiseok Lim. 2019. Domain adaptive training BERT for response selection. *CoRR*, abs/1908.04812.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *ACL*, pages 496–505.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *SIGIR*, pages 55–64.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019. Dialogpt: Large-scale generative pre-training for conversational response generation. *CoRR*, abs/1911.00536.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018. Modeling multi-turn conversation with deep utterance aggregation. In *COLING*, pages 3740–3752.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In *ACL*, pages 1118–1127.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, pages 19–27.