DIALOG STATE TRACKING WITH ATTENTION-BASED SEQUENCE-TO-SEQUENCE LEARNING

Takaaki Hori[1], *Hai Wang*[2], *Chiori Hori*[1],
Shinji Watanabe[1], *Bret Harsham*[1],
Jonathan Le Roux[1], *John R. Hershey*[1]

[1]Mitsubishi Electric Research Laboratories
[2]Toyota Technological Institute at Chicago

Yusuke Koji[3], *Yi Jing*[3],
Zhaocheng Zhu[4], *Takeyuki Aikawa*[3]

[3]Information Technology R&D Center,
Mitsubishi Electric Corporation
[4]Peking University

ABSTRACT

We present an advanced dialog state tracking system designed for the 5th Dialog State Tracking Challenge (DSTC5). The main task of DSTC5 is to track the dialog state in a human-human dialog. For each utterance, the tracker emits a frame of slot-value pairs considering the full history of the dialog up to the current turn. Our system includes an encoder-decoder architecture with an attention mechanism to map an input word sequence to a set of semantic labels, i.e., slot-value pairs. This handles the problem of the unknown alignment between the utterances and the labels. By combining the attention-based tracker with rule-based trackers elaborated for English and Chinese, the F-score for the development set improved from 0.475 to 0.507 compared to the rule-only trackers. Moreover, we achieved 0.517 F-score by refining the combination strategy based on the topic and slot level performance of each tracker. In this paper, we also validate the efficacy of each technique and report the test set results submitted to the challenge.

Index Terms— Dialog state tracking, attention model, sequence-to-sequence learning, encoder-decoder, long short-term memory

1. INTRODUCTION

Recently, spoken dialog systems have been widely used for many human-machine interfaces such as smart phones and car navigation systems. Spoken language understanding (SLU) technology, which predicts the intention of spoken user utterances, is a key component of dialog systems [1, 2]. SLU is usually designed as a system that converts a word sequence automatic speech recognition (ASR) result to a semantic representation that can be interpreted by a dialog manager. The semantic representation can be a concept tag for each utterance such as a dialog act and a set of slot-value pairs such as named entities and the category classes they belong to. To tackle SLU problems, classifier-based approaches are often trained using on a large amount of labeled data from the target domain. Support vector machines (SVMs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) have been applied to utterance-level tagging, which is purely an utterance classification problem [3, 4, 5]. Furthermore, contextual information such as previous utterances, topic, and speaker are also considered using long short-term memory (LSTM) RNNs and role-dependent LSTMs [6, 7]. For named entity or slot-value extraction, we can also apply classifiers for labeling each word in a given utterance. Conditional random fields (CRFs) and LSTM RNNs are available for this type of sequence labeling problem.

However, the classifier approaches mentioned above can only deal with relatively simple tasks, where the user's goal is clear and most dialogs are accomplished within a few turns. A sufficient amount of labeled data training must be available. A dialog system capable of responding in a human-like way must be capable of understanding ambiguous and complicated user goals, and be able to cope with users' changing goals over the course of a long dialog. In this case, the SLU problem is not well-defined.

Dialog state tracking [8] is an SLU technique intended to improved understanding for such extended dialogs. For this task, we need a good semantic representation of dialog states and a better way to predict the state at each moment of a given conversation. The 4th and 5th Dialog State Tracking Challenges (DSTC4 and DSTC5) [9, 10] are based on human-human dialogs.

The training data for DSTC4 and DSTC5 consists of dialogs which are divided into sub-dialog segments, each segment consisting of one or more spoken utterances. In the training data, all of the utterances in a segment are tagged with an identical set of slot-value pairs. The main goal of DSTC4 and DSTC5 is to design a *tracker* system to predict a set of slot-value pairs for each sub-dialog segment using the transcript (or the translated result in DSTC5) and the dialog context prior to the current utterance. Since the explicit alignment between slot-value pairs and individual utterances is not available in the training phase, the tracker needs to have a mechanism to acquire such alignments through the training step. Furthermore, since slot values do not appear exactly in the utterances as defined in the ontology database, the tracker needs to acquire different ways of expressing each slot value from training data. It is also necessary to include slot values from the dialog history, when the values are not present but are implied by demonstrative adjectives or pronouns in the current segment. In DSTC4, most classifier-based approaches performed poorly while elaborated rule-based trackers showed substantially better performance. Accordingly, there is much room for improvement in solving this task using machine learning approaches.

In this paper, we present our tracker system for the DSTC5 main task. DSTC5 is similar to the previous challenge, DSTC4 [9], but the training data are spoken in English while the development and test data are spoken in Chinese. Machine translation results from English-to-Chinese and Chinese-to-English systems are also provided in the training data and development/test data, respectively. To achieve a high performance for DSTC5, first we examine rule-based trackers for English and Chinese, which are similar to the best system in DSTC4 [11]. Secondly we investigate machine learning approaches, where we evaluate utterance classification methods using context-sensitive LSTM RNNs [6],

After that we propose an encoder-decoder-based tracker with an attention mechanism to map an input word sequence to a set of semantic labels, i.e., slot-value pairs, considering the dialog history.

This model handles the problem of the unknown alignment between the utterances and the labels. Finally we combine the attention-based tracker with rule-based trackers elaborated for English and Chinese. In the experiments, we compare the performance scores of different trackers and their combinations, and report the final test-set results we submitted to the challenge.

2. TASK AND SYSTEM

This section summarizes the dialog state trackers we built at Mitsubishi Electric Research Laboratories (MERL) and Mitsubishi Electric Corporation (MELCO) for DSTC5. First we review the main task of DSTC5 and clarify the problems to be solved in this task. Then, we introduce the system architecture consisting of multiple trackers based on rules and machine learning.

2.1. DSTC5 main task

The goal of the DSTC5 main task is to track dialog states for sub-dialog segments. A tracker has to predict a set of slot-value pairs which best matches the content of the current segment, in the context of the dialog history up to and including the current turn.

Unlike DSTC4 which handled English-only dialogs, DSTC5 aims at cross-language dialog state tracking, using English dialogs as training data and Chinese dialogs as development data. The test set for the challenge is also in Chinese. This scenario simulates building a tracker in a target language with given resources in a source language and their translations generated by machine translation to the target language.

The training and development sets contain manual annotations, which represent the segment boundaries, topic, and dialog state of each segment, and the speaker, transcription, and translation of each utterance. The dialog state is given as a set of slot-value pairs in a frame structure. All the slots and values used for annotations are defined in an ontology database provided by the organizer. Details are given in [10].

A dialog state tracker has to output the predicted dialog state for every utterance in a given dialog session. While all the transcripts and segment information from the beginning of the session to the current turn can be used, no information from the future turns is allowed to be used to analyze the state at a given turn. Although the fundamental goal of this tracking is to analyze the state for each sub-dialog level, the execution should be done at the utterance level regardless of the speaker from the beginning to the end of a given session in sequence. In a real deployment, it would be very useful to predict the parts of the dialog state as early as possible, so DSTC5 includes a task that evaluates the capabilities of trackers based on utterance level predictions.

Compared to well-defined SLU tasks, we need to consider the following problems.

(1) Slot values do not appear exactly in the utterances as defined in the ontology database. The ontology defines topical concepts and entity names with only their canonical forms, and does not contain any information about how they can be expressed by humans in a spoken dialog. For example, value "Pricerange" for slot "INFO" is implied by the utterance "how much does it cost in that hotel?". Value "Singapore Changi Airport" for slot "FROM" is implied by "how do you plan to travel from the airport to your hotel?" Accordingly, this is different from named entity recognition tasks based on word sequence labeling. The tracker needs to match many spoken forms to each slot-value pair.

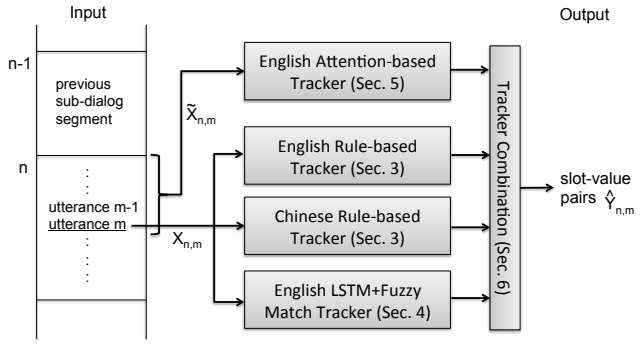

Fig. 1. The MERL/MELCO system for DSTC5

(2) Slot values are often implicitly represented by the speaker. In this case, the value is usually mentioned with a demonstrative adjective or pronoun in the current segment, but the actual name of the value exists in the previous segments. For example, the utterance "I think it's very romantic when we go there." has slot-value pair "PLACE: Siloso Beach". But the keyword "Siloso Beach" actually appeared four segments earlier. The tracker needs to include the implicit slot values from the dialog history.

(3) Zero or more slots and values are assigned to each sub-dialog. Each frame can be filled with multiple slots and each slot can be filled with multiple values. This is not a simple classification problem based on one-to-one mapping. We need to consider one-to-many mapping.

(4) There is no explicit alignment given between slot-value pairs and individual utterances. An identical set of slot-value pairs are assigned to each utterance in a sub-dialog segment. There is no information about which slot-value pair is derived from which utterance or keyword. The tracker needs to infer such alignments to predict the slot-value pairs correctly.

(5) There is insufficient data provided for DSTC5. The size of the training data annotated for DSTC5 is not sufficient to train large-scale machine learning-based models. Because the training data does not cover all slot values in the ontology database, a lot of unseen slot values need to be predicted for the development and test sets. It is allowed to use external data in addition to the provided data, however, external data does not contain any annotations corresponding to DSTC5.

2.2. The MERL/MELCO System

Figure 1 shows our DSTC5 tracking system. We use a combination of three types of trackers: rule-based trackers, an LSTM+Fuzzy match tracker, and an attention-based tracker, where there are separate English and Chinese versions of the rule-based tracker.

The input data are given at the utterance level or sub-dialog level. In the case of sub-dialog level, the range of input is limited up to the current utterance because we cannot use future information. The English and Chinese rule-based trackers and LSTM+Fuzzy match tracker process each utterance considering the history while the attention-based tracker processes a partial or whole sub-dialog segment at once. At the m-th turn of segment n, the system combines the slot-value pairs predicted by the four trackers and outputs the final result $Y_{n,m}$. We explain each tracker and the combination strategy in the following sections.

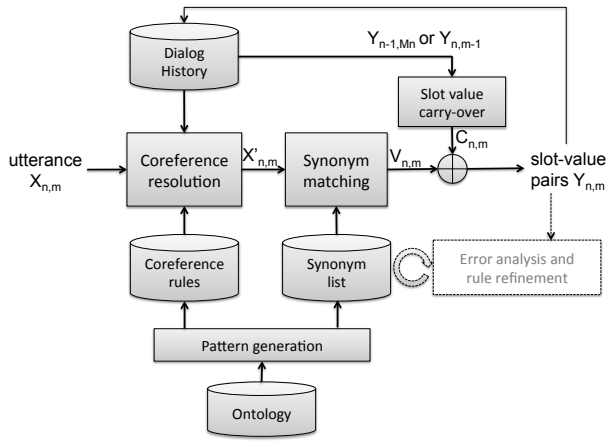

Fig. 2. Architecture of rule-based tracker

3. RULE-BASED TRACKERS

Our rule-based trackers make use of several techniques similar to the best system of DSTC4 [11]. Figure 2 illustrates the architecture of our rule-based trackers. The trackers include a set of rules for coreference resolution, synonym-based matching, and slot value carry-over.

Let $X_{n,m}$ be the m-th utterance in the n-th sub-dialog segment. $X_{n,m}$ is first processed with a coreference resolution module, which replaces demonstrative expressions in $X_{n,m}$ with the entity names of slot values from the dialog history. The output of coreference resolution, $X'_{n,m}$, is then used for pattern matching with synonyms defined in the synonym list. The synonym list has a set of synonym patterns for each slot-value pair, which can be written in the form of regular expressions. If a synonym pattern matches $X'_{n,m}$, the slot-value pair that the synonym matches is added to V_m, which is the set of slot-value pair candidates. In the step of slot value carry-over, a subset of the set of slot-value pairs predicted for the previous utterance, $Y_{n,m-1}$ (or Y_{n-1,M_n} if $m = 1$, where M_n denotes the number of utterances in the $(n-1)$-th segment), is selected and processed, leading to a set of slot-value pairs $C_{n,m}$. Then $C_{n,m}$ is added to V_m to obtain a prediction result $Y_{n,m}$ for the current utterance $X_{n,m}$.

Most of the rules are initially generated automatically from the ontology database. After that, they are refined manually by analyzing the errors generated by the system for the training data. The following subsections describe each module of the rule-based trackers.

3.1. Coreference resolution

Coreference resolution is the task of finding all expressions that refer to the same entity in a text [12]. It is an important step for many higher level NLP tasks such as document summarization [13], question answering [14], and information extraction [15]. For dialog state tracking, it can be used to solve the problem (2) mentioned in Section 2.1. However, publicly available tools for coreference resolution [16] perform poorly on the DSTC5 data since they are built based on well-written text and not applicable to the machine-translated conversational-style text of DSTC5. Accordingly, we take the same approach as [11], in which we use a set of templates to decide which mention is referring to which slot value in the dialog history.

The following examples show how our templates are defined:

```
" (the|this|that|these|those|your|my|our) hotel" : {
  "type" : "HOTEL",
  "slot" : ["PLACE"]
}
"(?<! is|are) (here|there) (?!is|'s|are|'re)" : {
 "type" : "*",
 "slot" : ["TO", "FROM", "PLACE", "STATION"]
}
```

The first example indicates that when the regular expression (the first element) has matched to the utterance, the system searches for a value with type "HOTEL" and filling a "PLACE" slot in the history. If there are multiple entities that satisfy the rules, the system basically chooses the most recent one. Then, the matched phrase in the utterance is replaced with the value name. For example, an utterance "*The hotel looks great.*" may be converted to "*Hilton Singapore Hotel looks great.*" The second example indicates that when the regular expression has matched with "*here*" or "*there*", the system searches for a value with any type filling a "TO", "FROM", "PLACE", or "STATION" slot in the history. To avoid matching with non-mention phrases such as "*there is ..*" and "*are there ...?*", positive/negative look-behind/look-ahead patterns are allowed to be used in the regular expression. We made 64 templates for English and Chinese trackers, respectively.

3.2. Synonym matching

We use a synonym list to increase the matching possibility for each slot value. When we only use the list of value names defined in the ontology file, the recall rate is very low even though we use a fuzzy string match technique. By expanding the expressions that can be uttered by speakers for each value, we can increase the recall rate without decreasing the precision rate. This technique alleviates problem (1) in Section 2.1.

Our rule trackers use a synonym list written as in the following example:

```
"112 Katong": {
  "PLACE": [
    [[" one twelve "], 2],
    [[" one twelve ", " katong "], 3],
    [[" one twelve katong "], 4],
    [[" one one two "], 4],
    [[" one one two ", " katong "], 5]]
}
```

The value "112 Katong" is the name of a shopping mall that can fill a "PLACE" slot if one of the synonyms in the list matches the utterance. Since numbers such as "112" can be expressed in different ways, the synonym list contains "one twelve" and "one one two". A synonym pattern can have multiple phrases, where the pattern is considered to be matched only if all the phrases have matched to the utterance. For example, the second pattern `[" one twelve ", " katong "]` matches utterances that include both phrases "one twelve" and "katong" in either order. For each synonym pattern, we also define a score that is used to decide the priority. We usually set the score based on the matching length. In the matching process, matches with higher scores are take priority over those with lower scores, and already matched portions in the utterance are not used for other pattern matches. This avoids detecting multiple values from the same portion of the utterance.

The initial list of synonyms is automatically generated from the ontology database, where we extract unigrams, bigrams, trigrams and separated word pairs from the word sequence of each slot value name defined in the ontology. Before extracting these patterns, we

normalize the English text into lowercase and convert Arabic numerals and symbols into English words (e.g., "313@" becomes "three one three at"). To avoid conflicts during pattern matching, we exclude any pattern that matches any slot value other than its own slot value. We also exclude patterns consisting only of common words, where the common word set is obtained from a large text corpus using TF-IDF measure or frequency of each word in the corpus. After that, these synonym patterns are refined manually through a cycle of tracking the training data and modifying the synonym patterns by analyzing frequent errors (both false positives and false negatives).

Our Chinese rule-based tracker was built in the same manner. Finally, 12,727 English synonyms and 17,196 Chinese synonyms were prepared for 1,690 values in the ontology.

3.3. Slot value carry-over

The rule-based tracker keeps detected slot-value pairs until the end of the current segment, and carries some specific slot-value pairs beyond the segment boundary to the next segment. We decide which slot-value pairs should be kept based on the topics of the previous and current segments and each slot type. The most likely slot for each triplet (previous topic, current topic, previous slot) is chosen based on the number of occurrences in the training data, where the value of the previous slot is given to the most likely slot when the previous topic changes to the current topic.

For example, carry-over rule

$$(\text{SHOPPING}, \text{TRANSPORTATION}, \text{PLACE}) \rightarrow \text{TO}$$

indicates that if the previous topic is "SHOPPING", the current topic is "TRANSPORTATION", and a value such as "City Square Mall" exists in a "PLACE" slot of Y_{n-1,M_n}, then a slot value pair "TO : City Square Mall" is added to $C_{n,1}$ at the beginning of the current segment. This mechanism also helps to alleviate problem (2) mentioned in Section 2.1.

4. LSTM+FUZZY MATCH TRACKER

The LSTM+Fuzzy match tracker has a two-pass process, in which the first pass predicts only slot types using an LSTM RNN, and the second pass detects slot values for the slot type candidates predicted in the first pass, where word and character based fuzzy string matching is used to detect the values.

For the first pass, we use an LSTM RNN depicted in Fig. 3. The network has an input layer that takes each input word, a projection layer that reduces the word vector in a low-dimensional space, a hidden layer with recurrent connections that keeps context information, and an output layer that estimates posterior probabilities of output labels. In the hidden layer, we use a set of LSTM cells instead of regular network units.

Considering a sequence of M utterances, $(u_m)_{m=1,\ldots,M}$, we denote each utterance's word sequence as $(w_{m,t})_{t=1,\ldots,T_m}$ and its output label as y_m. Since multiple slot types can be assigned to each utterance in DSTC5, we consider as output label the concatenation of the multiple slot types into one label, e.g., "INFO" and "PLACE" slot types are converted to "INFO+PLACE".

The input vector $x_{m,t}$ is prepared as

$$x_{m,t} = \text{Onehot}(w_{m,t}), \tag{1}$$

where word $w_{m,t}$ in vocabulary $\mathcal{V}$ is converted by 1-of-N coding using function OneHot(w), i.e. $x_{m,t} \in \{0,1\}^{|\mathcal{V}|}$.

The input vector is projected to the D dimensional vector

$$x'_{m,t} = W_{pr} x_{m,t} + b_{pr} \tag{2}$$

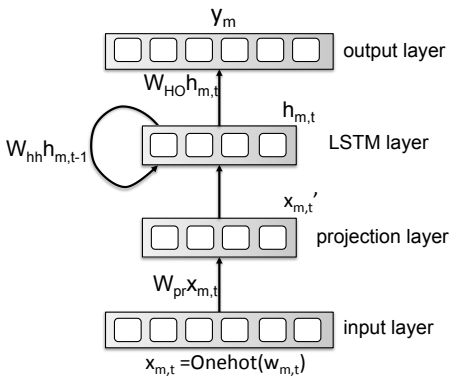

Fig. 3. LSTM RNN.

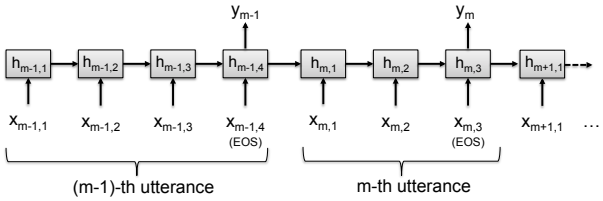

Fig. 4. Forward propagation of LSTM RNN.

and fed to the recurrent hidden layer, where W_{pr} and b_{pr} are the projection matrix and the bias vector.

At the hidden layer, activation vector $h_{m,t}$ is computed using LSTM cells according to [17][18].

$$h_{m,t} = \text{LSTM}(h_{m,t-1}, x'_{m,t}) \tag{3}$$

The output vector is computed at the end of each utterance as

$$p_m = \text{softmax}(W_{HO} h_{m,T_m} + b_{HO}), \tag{4}$$

where W_{HO} and b_{HO} are the transformation matrix and the bias vector to classify the input vector into different categories. softmax() is a softmax function that converts the classification result into label probabilities, i.e., $p_m \in [0,1]^{|\mathcal{L}|}$, $\sum_{o \in \mathcal{L}} p_m[o] = 1$, where $\mathcal{L}$ denotes the label set, and $p_m[o]$ indicates the component of p_m for label o, which corresponds to label probability $P(o|h_{m,T_m})$. The estimated label for utterance m is obtained as

$$\hat{y}_m = \underset{o \in \mathcal{L}}{\text{argmax}}\, p_m[o]. \tag{5}$$

To inherit the context information from the previous utterances, the hidden and cell activations at the beginning of each utterance are initialized with those at the final position T_{m-1} of the previous utterance, i.e.,

$$h_{m,0} = h_{m-1,T_{m-1}}, \quad c_{m,0} = c_{m-1,T_{m-1}}, \tag{6}$$

where $m > 1$, $h_{1,0} = c_{1,0} = 0$, and $c_{m,t}$ denotes the cell activation vector.

Figure 4 illustrates a propagation process of the context-sensitive slot-type prediction. Words are sequentially fed to the LSTM and an output label corresponding to a set of slots is generated at the end of the utterance, where the symbol "EOS" marks the end of a sentence. This model considers the entire context from the beginning of the

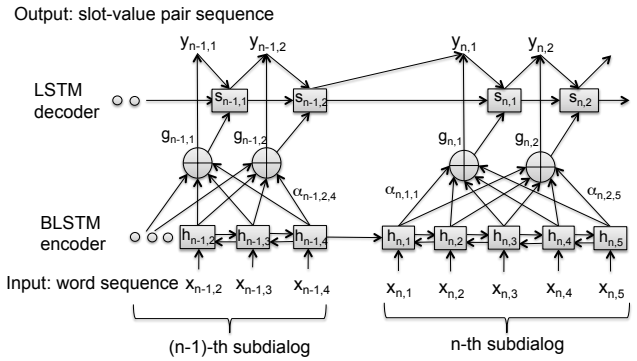

Fig. 5. Dialog state tracker with an attention-based encoder-decoder

dialog, and the label probabilities can be inferred using not only the sentence-level intentions but also the dialog-level context.

In the second pass, the tracker uses fuzzy matching to detect values for the slots predicted in the first pass. The second pass employs the character-level fuzzy matching from the official baseline tracker and a word-level fuzzy matching. We set two thresholds, one for the edit distance of character-level matching and one for the percentage of words of the slot value which appear in the target utterance.

5. ATTENTION-BASED TRACKER

This section proposes a dialog state tracker based on attention models to handle the problem of the unknown alignment between the utterances and the labels. The model predicts each output label by attending to salient phrases relevant to the label in the input sequence. This means that the model also predicts the alignment by the attention mechanism to choose correct labels.

Figure 5 shows the proposed tracker, where the word sequence from the beginning of the sub-dialog segment is the input and the slot-value pair sequence assigned to the segment is the output. The architecture of the network is based on the attention-based encoder decoder [19]. Although this model considers the whole prior context of the dialog, the attention mechanism is limited to the current sub-dialog segment. For DSTC5, a set of slot-value pairs for each segment is converted to a sequence in the alphabetical order, but we give a constraint to the model so that it does not output the same label twice in the sequence.

In the encoder, we use a bidirectional LSTM. Let $h^{(f)}_{n,j}$ and $h^{(b)}_{n,j}$ be the forward and backward hidden activation vectors, respectively. We can consider both contexts by concatenating the forward and backward activation vectors as

$$h_{n,j} = \begin{bmatrix} h^{(f)}_{n,j} \\ h^{(b)}_{n,j} \end{bmatrix}. \tag{7}$$

The attention mechanism is realized by using *attention weights* to the hidden activation vectors throughout the input sequence. Accordingly, important phrases are emphasized with these weights to predict the next output label.

Let $\alpha_{n,i,j}$ be an attention weight between the i-th output label and the j-th input word in the n-th segment. A summary vector of the n-th segment for the i-th output is obtained as a weighted sum of hidden activation vectors, i.e.,

$$g_{n,i} = \sum_{j=1}^{L_n} \alpha_{n,i,j} h_{n,j}. \tag{8}$$

The decoder network is an Attention-based Recurrent Sequence Generator (ARSG) [19] that generates an output label sequence with summary vectors $g_{n,i}$. The network also has an LSTM network, where the decoder state can be updated as

$$s_{n,i} = \text{LSTM}(s_{n,i-1}, y_{n,i}, g_{n,i}), \tag{9}$$

where LSTM() represents a function of LSTM layer in the decoder network. Then, the output label probability is computed as

$$p_{n,i} = \text{softmax}(W_{SO} s_{n,i-1} + W_{GO} g_{n,i} + b_{SO}) \tag{10}$$

and the output label is decided by

$$\hat{y}_{n,i} = \underset{o \in \mathcal{L}}{\text{argmax}}\, p_{n,i}[o]. \tag{11}$$

The attention weights are computed in the same manner as [20]

$$\alpha_{n,i,j} = \frac{\exp(e_{n,i,j})}{\sum_{k=1}^{L_n} \exp(e_{n,i,k})} \tag{12}$$

and

$$e_{n,i,j} = w^{\intercal} \tanh(W s_{n,i-1} + V h_{n,j} + b), \tag{13}$$

where w and b are vectors, W and V are matrices, and $e_{n,i,j}$ is a scalar.

In the training phase, we feed in an entire sub-dialog segment at once and compute the cross-entropy loss over the output sequence, where we exclude output labels that already appeared in the sequence from the softmax and cross-entropy computation. In the test phase, we make a partial sub-dialog segment from the beginning of the segment to the current utterance and feed it to the network. By adding the current utterance to the previous partial segment, we can repeatedly obtain utterance-level input and output. We apply the beam search to generate the output label sequence.

6. TRACKER COMBINATION

This section describes tracker combination methods. There are many ways to decide which tracker fills each slot. The first approach is to make a union of slot types obtained from multiple trackers, and fill each slot in the union with a value (or values) from one tracker. In the first approach, four trackers overwrite the values of each slot in a predefined order. According to our preliminary experiments, a better tracker should be positioned later in the ordered trackers. This means that slot values are always overwritten by a better tracker if it has values for the slot.

The second approach is to filter out unreliable slots from each tracker and combine the remaining slot-values in the same manner as the first approach. We found that the multiple trackers have different performances depending on topic and slot. The rule-based trackers have better performances in slot types with concrete values such as "PLACE" and "STATION" while the attention-based tracker has a better performance in slot types with abstract values such as "INFO" and "TYPE_OF_PLACE." This means that we can select for each tracker unreliable slot types where it poorly predicts the slot values, using the training and development data.

7. EXPERIMENTS

We evaluated four trackers and their combination using the DSTC5 data sets. Frame level accuracy and precision, recall, and F1-score of slot-value pairs were computed for the tracker's output. These performance measures were obtained for two evaluation schedules,

Table 1. Evaluation results for development set

	Schedule 1				Schedule 2			
tracker	accuracy	precison	recall	F1-score	accuracy	precison	recall	F1-score
(a) Baseline EN	0.0411	0.1769	0.1105	0.1360	0.0543	0.2536	0.1431	0.1830
(b) Rule EN	0.0589	**0.5217**	0.2793	0.3638	0.0995	**0.5117**	0.3129	0.3883
(c) Rule ZH	0.0909	0.4831	0.3243	0.3881	0.1312	0.5040	0.3845	0.4362
(d) LSTM+Fuzzy EN	0.0471	0.3851	0.1785	0.2439	0.0543	0.3522	0.2168	0.2684
(e) Attention EN	0.0616	0.4629	0.2079	0.2869	0.0633	0.4698	0.2229	0.3024
(f) Union(b,c)	0.1000	0.4758	0.3773	0.4209	0.1312	0.4967	0.4560	0.4754
(g) Union(d,b,c)	0.0918	0.4423	0.3878	0.4132	0.1086	0.4605	0.4765	0.4683
(h) Union(e,b,c)	0.0968	0.4582	0.4570	**0.4570**	0.1312	0.4816	0.5358	0.5073
(i) Filter(e,b,c)	**0.1014**	0.4484	**0.4653**	0.4567	**0.1403**	0.4917	**0.5460**	**0.5174**

Table 2. Evaluation results for test set

		Schedule 1				Schedule 2			
entry	tracker	accuracy	precison	recall	F1-score	accuracy	precison	recall	F1-score
1	(f-)	**0.0583**	**0.4008**	0.2776	0.3280	**0.0765**	**0.4127**	0.3284	0.3658
2	(i-)	0.0407	0.3554	0.3267	0.3405	0.0413	0.3569	0.3575	0.3572
3	(i)	0.0515	0.3682	**0.3735**	**0.3708**	0.0635	0.3768	**0.4140**	**0.3945**
4	(i')	0.0552	0.3717	0.3583	0.3649	0.0681	0.3806	0.4026	0.3913
5	(h)	0.0454	0.3473	0.3677	0.3572	0.0559	0.3510	0.4043	0.3758

in which Schedule 1 evaluates the performance at utterance level while Schedule 2 evaluates the performance at sub-dialog level. See details of the data sets and the evaluation procedure in [10].

We built rule-based trackers for English and Chinese, where Jieba [21] was used to segment each Chinese utterance into words. The LSTM and attention models were implemented using Chainer [22]. In the training phase of the LSTM and attention models, we initialized the projection layer W_{pr} with the 50 dimensional word vectors obtained by word2vec [23], where the wikipedia text of Singapore-related keywords were used for training the word2vec. In the case of the attention model, we limited the output labels into the most common 200 labels in the training data, because slot-value pairs do not appear sufficiently in the training data. The 200 labels cover 80% of slot-value pairs in the training set and 78% in the development set. The number of hidden units was set to 50 in all the LSTM layers of the attention model.

7.1. Evaluation with development set

Table 1 shows the results for the development set. (a) Baseline EN indicates the official baseline tracker for English. (b) Rule EN and (c) Rule ZH correspond to our English and Chinese rule-based trackers, respectively. The rule-based systems substantially outperformed the official baseline system that used only fuzzy string matching with the slot values. The Chinese tracker showed better performance than the English one in most measures. This is because the development set is originally Chinese and the tracker was not directly affected by translation errors. (d) LSTM+Fuzzy EN represents the LSTM+Fuzzy match tracker for English. The LSTM tracker did not perform well for DSTC5.

(e) Attention EN indicates the proposed attention-based tracker. It yielded a better performance than the LSTM+Fuzzy tracker. Although it did not outperform the rule-based trackers, the attention model approach is still promising because the model was trained only with the training data without any information from the ontology or any manual refinement. Furthermore, the combination of the attention model with the rule-based trackers, i.e., tracker (h), significantly improved the performance over the rule-only tracker combination (f). Finally, we achieved 0.517 F-score for the development set using the filter-based combination (i).

7.2. Final test set results

Table 2 shows the test set results we submitted to DSTC5. Entry 1 corresponds to the rule-only tracker combination (f-). Entry 2 (i-) is a filter-based combination. After submitting both of these entries, we improved the Rule ZH system (c), so we have marked both of these systems with a - symbol to indicate that they do not correspond exactly to the improved systems f and i in Table 1. Entries 3-5 used the improved Rule ZH system.

Entry 3 is the filter-based combination (i) of the two rule trackers and the attention-based tracker. Entry 3 yielded the best F1-score for the test set. Entry 4 (i') is a filter-based combination but which minimizes the effect of translation errors by fixing Chinese translations in the ontology. This system gave almost the same performance as Entry 3. Entry 5 used the union combination of trackers (h). In both the development and test sets, the filter-based combination is better than the union-based combination. Compared to the results from other teams, our trackers are in the second place out of the nine teams' submissions measured by any evaluation metric.

8. CONCLUSIONS

We designed a dialog state tracking system for DSTC5 that combines an encoder-decoder architecture including an attention mechanism with rule-based trackers elaborated for English and Chinese. By refining the combination strategy based on the topic and slot level performance of each tracker, we achieved 0.517 F-score on the development set. For the test set, we obtained 0.395 F-score with the combined tracker, which is in the second place out of the nine teams.

Our results show that adding a modern neural network component to a dialog tracker can improve performance, even given very limited training data. Such a system could be deployed as an initial prototype and data collection tool, leading to expanded training data and improved performance.

9. REFERENCES

[1] Dan Jurafsky and James H Martin, *Speech & Language Processing*, Pearson Education, 2000.

[2] Renato De Mori, "Spoken language understanding: a survey.," in *ASRU2007*, 2007, pp. 365–376.

[3] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu, "Recurrent neural networks for language understanding.," in *Interspeech2013*, 2013, pp. 2524–2528.

[4] Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao, "Recurrent conditional random field for language understanding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2014)*. IEEE, 2014, pp. 4077–4081.

[5] Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi, "Spoken language understanding using long short-term memory neural networks," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 189–194.

[6] Chiori Hori, Takaaki Hori, Shinji Watanabe, and John R. Hershey, "Context sensitive spoken language understanding using role dependent LSTM layers," in *Machine Learning for SLU & Interaction NIPS 2015 Workshop*, 2015.

[7] Chiori Hori, Takaaki Hori, Shinji Watanabe, and John R. Hershey, "Context-sensitive and role-dependent spoken language understanding using bidirectional and attention LSTMs," in *INTERSPEECH 2016*, 2016.

[8] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black, "The dialog state tracking challenge," in *Proceedings of the SIGDIAL 2013 Conference*, 2013, pp. 404–413.

[9] Seokhwan Kim, Luis Fernando D'Haro, Rafael E. Banchs, Jason Williams, and Matthew Henderson, "The Fourth Dialog State Tracking Challenge," in *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS)*, 2016.

[10] Seokhwan Kim, Luis Fernando D'Haro, Rafael E. Banchs, Jason Williams, Matthew Henderson, and Koichiro Yoshino, "The Fifth Dialog State Tracking Challenge," in *Proceedings of the 2016 IEEE Workshop on Spoken Language Technology (SLT)*, 2016.

[11] Franck Dernoncourt, Ji Young Lee, Trung H Bui, and Hung H Bui, "Robust dialog state tracking for large ontologies," *arXiv preprint arXiv:1605.02130*, 2016.

[12] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky, "Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task," in *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, 2011, pp. 28–34.

[13] Josef Steinberger, Mijail Kabadjov, and Massimo Poesio, "Coreference applications to summarization," in *Poesio, M., Stuckardt, R., & Versley, Y. (Eds.). Anaphora Resolution: Algorithms, Resources, and Applications.* Springer, 2016.

[14] Hai Wang and Mohit Bansal Kevin Gimpel David McAllester, "Machine comprehension with syntax, frames, and semantics," *Volume 2: Short Papers*, pp. 700–706, 2015.

[15] Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam, "Event coreference for information extraction," in *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, Stroudsburg, PA, USA, 1997, ANARESOLUTION '97, pp. 75–81, Association for Computational Linguistics.

[16] "Stanford deterministic coreference resolution system," in *"http://nlp.stanford.edu/projects/coref.shtml"*.

[17] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013, pp. 6645–6649.

[19] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., pp. 577–585. Curran Associates, Inc., 2015.

[20] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[21] "Jieba," in *"https://github.com/fxsjy/jieba"*.

[22] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton, "Chainer: a next-generation open source framework for deep learning," in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

[23] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, 2013.