A Network-based End-to-End Trainable Task-oriented Dialogue System

Tsung Hsien Wen
University of Cambridge

Lina M. Rojas-Barahona
University of Cambridge

Pei-Hao Su
University of Cambridge

Stefan Ultes
University of Cambridge

A Network-based End-to-End Trainable Task-oriented Dialogue System

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, Steve Young
Cambridge University Engineering Department,
Trumpington Street, Cambridge, CB2 1PZ, UK
{thw28,mg436,nm480,lmr46,phs26,su259,djv27,sjy11}@cam.ac.uk

Abstract

Teaching machines to accomplish tasks by conversing naturally with humans is challenging. Currently, developing task-oriented dialogue systems requires creating multiple components and typically this involves either a large amount of handcrafting, or acquiring labelled datasets and solving a statistical learning problem for each component. In this work we introduce a neural network-based text-in, text-out end-to-end trainable dialogue system along with a new way of collecting task-oriented dialogue data based on a novel pipe-lined Wizard-of-Oz framework. This approach allows us to develop dialogue systems easily and without making too many assumptions about the task at hand. The results show that the model can converse with human subjects naturally whilst helping them to accomplish tasks in a restaurant search domain.

1 Introduction

Building a task-oriented dialogue system such as a hotel booking or a technical support service is difficult because it is application-specific and there is usually limited availability of training data. To mitigate this problem, recent machine learning approaches to task-oriented dialogue system design have cast the problem as a partially observable Markov Decision Process (POMDP) [26] with the aim of using reinforcement learning (RL) to train dialogue policies online through interactions with real users [2]. However, the language understanding [4, 24] and language generation [23] modules still rely on supervised learning and therefore need corpora to train on. Furthermore, to make RL tractable, the state and action space must be carefully designed [26], which may restrict the expressive power and learnability of the model. Also, the reward functions needed to train such models are difficult to design and hard to measure at run-time [16].

At the other end of the spectrum, sequence to sequence learning [18] has inspired several efforts to build end-to-end trainable, non-task-oriented conversational systems [15, 14]. This family of approaches treats dialogue as a source to target sequence transduction problem, applying an encoder network to encode a user query into a distributed vector representing its semantics, which then conditions a decoder network to generate each system response. These models typically require a large amount of data to train. They allow the creation of effective chatbot type systems but they lack any capability for supporting domain specific tasks, for example, being able to interact with databases [17, 25] and aggregate useful information into their responses.

In this work, we propose a neural network-based model for task-oriented dialogue systems by balancing the strengths and the weaknesses of the two research communities: the model is end-to-end trainable but still modularly connected; it does not directly model the user goal, but nevertheless, it still learns to accomplish the required task by providing *relevant* and *appropriate* responses at each turn; it has an explicit representation of database (DB) attributes (slot-value pairs) which it uses to achieve a high task success rate, but has a distributed representation of user intent (dialogue act) to

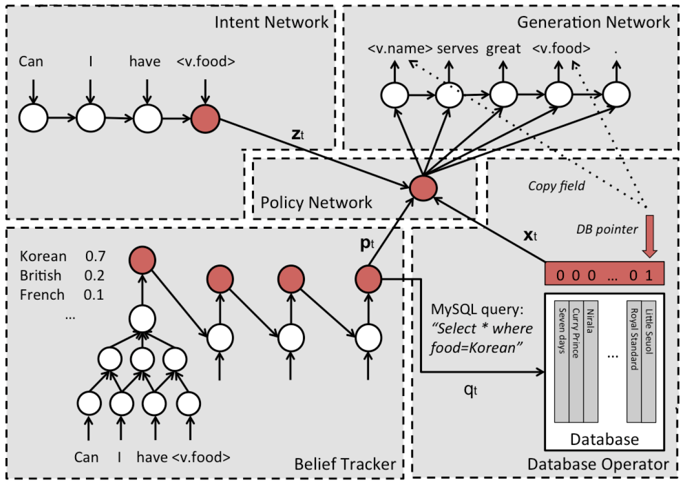

Figure 1: The proposed end-to-end trainable dialogue system framework

allow ambiguous inputs; and it uses delexicalisation[1] and a weight tying strategy [4] to reduce the data required to train the model, but still maintains a high degree of freedom should larger amounts of data become available. We show that the proposed model performs a given task very competitively across several metrics when trained on only a few hundred dialogues.

In order to train the model for the target application, we introduce a novel pipe-lined data collection mechanism inspired by the Wizard-of-Oz paradigm [7] to collect human-human dialogue corpora via crowd-sourcing. We found this process is simple to operate and enables fast data collection online with very low development costs.

2 Model

We treat dialogue as a sequence to sequence mapping problem (modelled by a sequence-to-sequence architecture [18]) augmented with the dialogue history (modelled by a belief tracker [4]) and the current database search outcome (modelled by a database operator), as shown in Figure 1. At each turn, the system takes a sequence of tokens[1] from the user as input and converts it into two internal representations: a distributed representation generated by an intent network and a probability distribution over slot-value pairs called the belief state [26] generated by a set of belief trackers. The database operator then selects the most probable values in the belief state to form a query to the DB, and the search result, along with the intent representation and belief state are transformed and combined by a policy network to form a single vector representing the next system action. This system action vector is then used to condition a response generation network [22] which generates the required system output token by token in skeletal form. The final system response is then formed by substituting the actual values of the database entries into the skeletal sentence structure. A more detailed description of each component is given below.

2.1 Intent Network

The intent network can be viewed as the encoder in the sequence-to-sequence learning framework [18] whose job is to encode a sequence of input tokens $w_0^t, w_1^t, ...w_N^t$ into a distributed vector representation $\mathbf{z}_t$ at every turn t. Typically, a Long Short-term Memory (LSTM) network [5] is used and the last time step hidden layer $\mathbf{z}_t^N$ is taken as the representation,

$$\mathbf{z}_t = \mathbf{z}_t^N = \text{LSTM}(w_0^t, w_1^t, ...w_N^t) \tag{1}$$

Alternatively, a convolutional neural network (CNN) can be used in place of the LSTM to encode the sentence [6, 8],

$$\mathbf{z}_t = \text{CNN}(w_0^t, w_1^t, ...w_N^t) \tag{2}$$

[1]Delexicalisation: we replaced slots and values by generic tokens (e.g. keywords like Chinese or Indian replaced by <v.food> in Figure 1) to allow weight sharing.

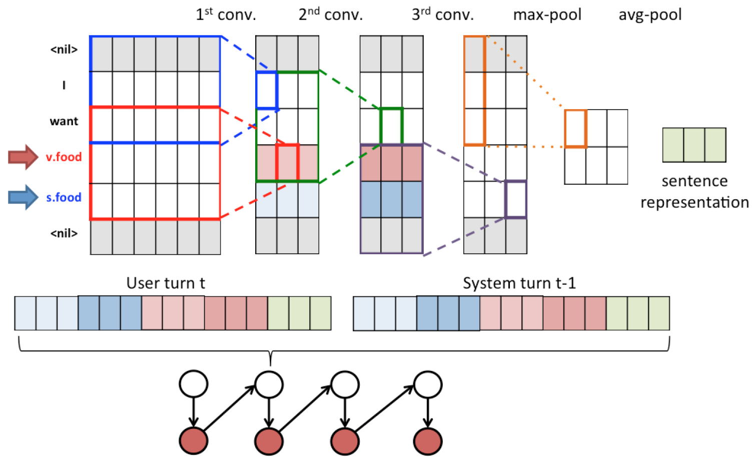

Figure 2: Tied Jordan-type RNN belief tracker with delexicalised CNN feature extractor. The outcome of the CNN feature extractor is a concatenation of top-level sentence (green) embedding and several levels of intermediate ngram-like embeddings (red and blue). We pad zero vectors (in gray) before each convolution operation to make sure the representation at each layer has the same length.

and here we investigate both. Since all the slot-value specific information is delexicalised, the encoded vector can be viewed as a distributed intent representation which replaces the hand-coded dialogue act representation [19] in traditional task-oriented dialogue systems.

2.2 Belief Trackers

Belief tracking (also called Dialogue State tracking) provides the core of a task-oriented spoken dialogue system (SDS) [3]. Current state-of-the-art belief trackers use discriminative models such as recurrent neural networks (RNN) to directly map ASR hypotheses to belief states [4]. Although in this work we focus on text-based dialogue systems, we retain belief tracking at the core of our system because: (1) it enables a sequence of free-form natural language sentences to be mapped into a fixed set of slot-value pairs, which can then be used to query a DB; (2) by keeping track of the dialogue state, it avoids learning unnecessarily complicated long-term dependencies from raw inputs; (3) it uses a smart weight tying strategy that can greatly reduce the data required to train the model, and (4) it provides an inherent robustness which simplifies future extension to spoken systems.

Using each user input as new evidence, the task of a belief tracker is to maintain a multinomial distribution p over values $v \in V_s$ for each informable slot s[2], and a binary distribution for each requestable slot[3]. Each slot in the ontology[4] $\mathbb{G}$ has its own specialised tracker, and each tracker is a Jordan-type RNN[5] with a CNN feature extractor, as shown in Figure 2. Like Mrksic et al (2015) [11], we tie the RNN weights together for each value v but vary features $\mathbf{f}_v^t$ when updating each pre-softmax activation g_v^t. The update equations for a given slot s are,

$$\mathbf{f}_v^t = \mathbf{f}_{v,cnn}^t \oplus p_v^{t-1} \oplus p_\emptyset^{t-1} \tag{3}$$

$$g_v^t = \mathbf{w}_s \cdot \text{sigmoid}(\mathbf{W}_s \mathbf{f}_v^t + \mathbf{b}_s) + b'_s \tag{4}$$

$$p_v^t = \frac{\exp(g_v^t)}{\exp(g_{\emptyset,s}) + \sum_{v' \in V_s} \exp(g_{v'}^t)} \tag{5}$$

where vector $\mathbf{w}_s$, matrix $\mathbf{W}_s$, bias terms $\mathbf{b}_s$ and b'_s, and scalar $g_{\emptyset,s}$ are parameters. $p_\emptyset^t$ is the probability that the user has not mentioned that slot up to turn t and can be calculated by substituting $g_{\emptyset,s}$ for $g_{v,s}$ in the numerator of Equation 5. In order to model the discourse context at each turn, the feature vector $\mathbf{f}_{v,cnn}^t$ is the concatenation of two CNN derived features, one from processing the user

[2] Informable slots are slots that users can use to constrain the search, such as food type or price range.

[3] Requestable slots are slots that users can ask a value for, such as phone number or address.

[4] A small knowledge graph defining the slot-value pairs the system can talk about for a particular task.

[5] We don't use the Jordan connection for requestable slots since they don't need to be tracked.

side u_t at turn t and other from processing the machine response m_{t-1} at turn $t-1$,

$$\mathbf{f}_{v,cnn}^{t} = \text{CNN}_{s,v}^{(u)}(u_t) \oplus \text{CNN}_{s,v}^{(m)}(m_{t-1}) \tag{6}$$

In order to make the tracker aware when delexicalisation is applied to a slot or value, the slot-value specialised CNN operator $\text{CNN}_{s,v}^{(\cdot)}(\cdot)$ extracts not only the top level sentence representation but also intermediate n-gram-like embeddings determined by the position of the delexicalised token in each utterance. If multiple matches are observed, the corresponding embeddings are summed. On the other hand, if there is no match for a particular slot or value, the empty n-gram embeddings are padded with zeros. In order to keep track of the position of delexicalised tokens, both sides of the sentence are padded with zeros before each convolution operation. The number of vectors is determined by the filter size at each layer. The overall process of extracting several layers of position-specific features is visualised in Figure 2.

The belief tracker used here is based on Henderson et al (2014) [4] with some modifications: (1) only probabilities over informable and requestable slots and values are output, (2) the recurrent memory block is removed, since it appears to offer no benefit in this task, and (3) the n-gram feature extractor is replaced by the CNN extractor described above. By introducing slot-based belief trackers, we essentially add a set of intermediate labels into the system as compared to training a pure end-to-end system. Later in the paper we will show that these tracker components are critical for achieving task success. We will also show that the additional annotation requirement that they introduce can be successfully mitigated using a novel pipe-lined Wizard-of-Oz data collection framework.

2.3 Policy Network and Database Operator

Database Operator Based on the output $\mathbf{p}_s^t$ of belief trackers, the DB query q_t is formed by,

$$q_t = \bigcup_{s' \in S_I} \operatorname*{argmax}_{v} \mathbf{p}_{s'}^{t} \tag{7}$$

where S_I is the set of informable slots. This query is then applied to the DB to creates a binary truth value vector $\mathbf{x}_t$ over DB entities where a 1 indicates that the corresponding entity is consistent with the query (and hence it is consistent with the most likely belief state). In addition, if $\mathbf{x}$ is not entirely null, an associated entity pointer is maintained which identifies one of the matching entities selected at random. The entity pointer is updated if the current entity no longer matches the search criteria; otherwise it stays the same. The entity referenced by the entity pointer is used to form the final system response as described in Section 2.4.

Policy network The policy network can be viewed as the glue which binds the system modules together. Its output is a single vector $\mathbf{o}_t$ representing the system action, and its inputs are comprised of $\mathbf{z}_t$ from the intent network, the belief state $\mathbf{p}_s^t$, and the DB truth value vector $\mathbf{x}_t$. Since the generation network only generates appropriate sentence forms, the individual probabilities of the categorical values in the informable belief state are immaterial and are summed together to form a summary belief vector for each slot $\hat{\mathbf{p}}_s^t$ represented by three components: the summed value probabilities, the probability that the user said they "don't care" about this slot and the probability that the slot has not been mentioned. Similarly for the truth value vector $\mathbf{x}_t$, the number of matching entities matters but not their identity. This vector is therefore compressed to a 6-bin 1-hot encoding $\hat{\mathbf{x}}_t$, which represents different degrees of matching in the DB (no match, 1 match, ... or more than 5 matches). Finally, the policy network output is generated by a three-way matrix transformation,

$$\mathbf{o}_t = \tanh(\mathbf{W}_{zo}\mathbf{z}_t + \mathbf{W}_{po}\hat{\mathbf{p}}_t + \mathbf{W}_{xo}\hat{\mathbf{x}}_t) \tag{8}$$

where matrices $\mathbf{W}_{zo}$, $\mathbf{W}_{po}$, and $\mathbf{W}_{xo}$ are parameters and $\hat{\mathbf{p}}_t = \bigoplus_{s \in \mathbb{G}} \hat{\mathbf{p}}_s^t$ is a concatenation of all summary belief vectors.

2.4 Generation Network

The generation network uses the action vector $\mathbf{o}_t$ to condition a language generator [23]. This generates template-like sentences token by token based on the language model probabilities,

$$P(w_{j+1}^t | w_j^t, \mathbf{h}_{j-1}^t, \mathbf{o}_t) = \text{LSTM}_j(w_j^t, \mathbf{h}_{j-1}^t, \mathbf{o}_t) \tag{9}$$

where $\text{LSTM}_j(\cdot)$ is a conditional LSTM operator for one output step j, w_j^t is the last output token (i.e. a word, a delexicalised slot name or a delexicalised slot value), and $\mathbf{h}_{j-1}^t$ is the hidden layer. Once the output token sequence has been generated, the generic tokens are replaced by their actual values: (1) replacing delexicalised slots by random sampling from a list of surface forms, e.g. *<s.food>* to *food* or *type of food*, and (2) replacing delexicalised values by the actual attribute values of the entity currently selected by the DB pointer. This is similar in spirit to the Latent Predictor Network [10] where the token generation process is augmented by a set of pointer networks [21] to transfer entity specific information into the response.

Attentive Generation Network Instead of decoding responses directly from a static action vector $\mathbf{o}_t$, an attention-based mechanism [1] can be used to dynamically aggregate source embeddings at each output step j. In this work we explore the use of an attention mechanism to combine the tracker belief states i.e. $\mathbf{o}_t$ is computed at each output step j by,

$$\mathbf{o}_t^{(j)} = \tanh(\mathbf{W}_{zo}\mathbf{z}_t + \mathbf{W}_{po}\hat{\mathbf{p}}_t^{(j)} + \mathbf{W}_{xo}\hat{\mathbf{x}}_t) \tag{10}$$

where for a given ontology $\mathbb{G}$

$$\hat{\mathbf{p}}_t^{(j)} = \sum_{s\in\mathbb{G}} \alpha_s^{(j)} \left(\mathbf{W}_{po}^s \cdot \hat{\mathbf{p}}_s^t\right) \tag{11}$$

and where the attention weights $\alpha_s^{(j)}$ are calculated by a scoring function,

$$\alpha_s^{(j)} = \text{softmax}\big(\mathbf{r}^\intercal \tanh\left(\mathbf{W}_r \cdot (\mathbf{z}_t \oplus \hat{\mathbf{x}}_t \oplus \hat{\mathbf{p}}_s^t \oplus \mathbf{w}_{j-1}^t \oplus \mathbf{h}_{j-1}^t)\right)\big) \tag{12}$$

where matrix $\mathbf{W}_r$ and vector $\mathbf{r}$ are parameters to learn and $\mathbf{w}_{j-1}^t$ is the embedding of token w_{j-1}^t.

3 Wizard-of-Oz Data Collection

Arguably the greatest bottleneck for statistical approaches to dialogue system development is the collection of appropriate training data, and this is especially true for task-oriented dialogue systems. Serban et al (2015) [13] have catalogued existing corpora for developing conversational agents. Such corpora may be useful for bootstrapping, but for task-oriented dialogue systems in-domain data is essential[6]. To mitigate this problem, we propose a novel crowdsourcing version of the Wizard-of-Oz (WOZ) paradigm [7] for collecting domain-specific corpora.

Based on the given ontology, we designed two webpages on Amazon Mechanical Turk, one for wizards and the other for users (see Appendix A for the designs). The users are given a task specifying the characteristics of a particular entity that they must find (e.g. *a Chinese restaurant in the north*) and asked to type in natural language sentences to fulfill the task. The wizards are given a form to record the information conveyed in the last user turn (e.g. *pricerange=Chinese, area=north*) and a search table showing all the available matching entities in the database. Note these forms contain all the labels needed to train the slot-based belief trackers. The table is automatically updated every time the wizard submits new information. Based on the updated table, the wizard types an appropriate system response and the dialogue continues.

In order to enable large-scale parallel data collection and avoid the distracting latencies inherent in conventional WOZ scenarios, users and wizards are asked to contribute just a single turn to each dialogue. To ensure coherence and consistency, users and wizards must review all previous turns in that dialogue before they contribute their turns. Thus dialogues progress in a pipe-line. Many dialogues can be active in parallel and no worker ever has to wait for a response from the other party in the dialogue. Despite the fact that multiple workers contribute to each dialogue, we observe that dialogues are generally coherent yet diverse. Furthermore, this turn-level data collection strategy seems to encourage workers to learn and correct each other based on previous turns.

In this paper, the system was designed to assist users find a restaurant in the Cambridge, UK area. There are three informable slots (*food*, *pricerange*, *area*) that users can use to constrain the search and six requestable slots (*address*, *phone*, *postcode* plus the three informable slots) that the user can ask a value for once a restaurant has been offered. There are 99 restaurants in the DB. Based on this domain, we ran 3000 HITs (Human Intelligence Tasks) in total for roughly 3 days and collected 1500 dialogue turns. After cleaning the data, we have approximately 680 dialogues in total (some of them are unfinished). The total cost for collecting the dataset was ~ 400 USD.

[6]E.g. technical support for Apple computers may differ completely from that for Windows, due to the many differences in software and hardware.

4 Empirical Experiments

Training Training is divided into two phases. Firstly the belief tracker parameters θ_b are trained using the cross entropy errors between tracker labels $\mathbf{y}_s^t$ and predictions $\mathbf{p}_s^t$, $L_1(\theta_b) = \sum_t \sum_s (\mathbf{y}_s^t)^\intercal \log \mathbf{p}_s^t$. For the full model, we have three informable trackers (*food, pricerange, area*) and seven requestable trackers (*address, phone, postcode, name*, plus the three informable slots).

Having fixed the tracker parameters, the remaining parts of the model are trained using the cross entropy errors from the generation network language model, $L_2(\theta - \theta_b) = \sum_t \sum_j (\mathbf{y}_j^t)^\intercal \log \mathbf{p}_j^t$, where $\mathbf{y}_j^t$ and $\mathbf{p}_j^t$ are output token targets and predictions respectively, at turn t of output step j. We treated each dialogue as a batch and used stochastic gradient decent with a small $l2$ regularisation term to train the model. The collected corpus was partitioned into a training, validation, and testing sets in the ratio 3:1:1. Early stopping was implemented based on the validation set for regularisation and gradient clipping was set to 1. All the hidden layer sizes were set to 50, and all the weights were randomly initialised between -0.3 and 0.3 including word embeddings. The vocabulary size is around 500 for both input and output, in which rare words and words that can be delexicalised are removed. We used three convolutional layers for all the CNNs in the work and all the filter sizes were set to 3. Pooling operations were only applied after the final convolution layer.

Decoding In order to decode without length bias, we decoded each system response m_t based on the average log probability of tokens in the sentence,

$$m_t^* = \underset{m_t}{\operatorname{argmax}} \{\log p(m_t|\theta, u_t)/J_t\} \tag{13}$$

where θ are the model parameters, u_t is the user input, and J_t is the length of the machine response.

As a contrast, we also investigated the MMI criterion [9] to increase diversity and put additional scores on delexicalised tokens to encourage task completion. This *weighted* decoding strategy has the following objective function,

$$m_t^* = \underset{m_t}{\operatorname{argmax}} \{\big(\log p(m_t|\theta, u_t) - \lambda \log p(m_t)\big)/J_t + \gamma R_t\} \tag{14}$$

where λ and γ are weights selected on validation set and $\log p(m_t)$ can be modelled by a standalone LSTM language model. We used a simple heuristic for the scoring function R_t designed to reward giving appropriate information and penalise spuriously providing unsolicited information[7]. We applied beam search with a beamwidth equal to 10, the search stops when an end of sentence token is generated. In order to obtain language variability from the deployed model we ran decoding until we obtained 5 candidates and randomly sampled one as the system response.

Tracker performance Table 1 shows the evaluation of the trackers' performance. Due to delexicalisation, both CNN type trackers and N-gram type trackers [4] achieve high precision, but the N-gram tracker has worse recall. This result suggests that compared to simple N-grams, CNN type trackers can better generalise to sentences with long distance dependencies and more complex syntactic structures.

Table 1: Tracker performance in terms of Precision, Recall, and F-1 score.

Tracker input type	Informable			Requestable		
	Prec.	Recall	F-1	Prec.	Recall	F-1
cnn	99.77%	96.09%	97.89%	98.66%	93.79%	96.16%
ngram	99.34%	94.42%	96.82%	98.56%	90.14%	94.16%

Corpus-based evaluation We evaluated the end-to-end system by first performing a corpus-based evaluation in which the model is used to predict each system response in the held-out test set. Three evaluation metrics were used: BLEU score (on top-1 and top-5 candidates) [12], entity matching rate and objective task success rate [16]. We calculated the entity matching rate by determining whether the actual selected entity at the end of each dialogue matches the task that was specified to

[7]We give an additional reward if a requestable slot (e.g. address) is requested and its corresponding delexicalised slot or value token (e.g. <v.address> and <s.address>) is generated. We give an additional penalty if an informable slot is never mentioned (e.g. food=none) but its corresponding delexicalised value token is generated (e.g. <v.food>). For a more details on scoring, please see Appendix B.

Table 2: Performance comparison of different model architectures based on a corpus-based evaluation.

Encoder	Tracker	Decoder	Match(%)	Success(%)	T5-BLEU	T1-BLEU
Baseline						
lstm	-	lstm	-	-	0.1650	0.1718
lstm	turn recurrence	lstm	-	-	0.1813	0.1861
Variant						
lstm	rnn-cnn, w/o req.	lstm	89.70	30.60	0.1769	0.1799
cnn	rnn-cnn	lstm	88.82	58.52	0.2354	0.2429
Full model w/ different decoding strategy						
lstm	rnn-cnn	lstm	86.34	75.16	0.2184	0.2313
lstm	rnn-cnn	+ weighted	86.04	78.40	0.2222	0.2280
lstm	rnn-cnn	+ att.	90.88	80.02	0.2286	0.2388
lstm	rnn-cnn	+ att. + weighted	90.88	83.82	0.2304	0.2369

the user. The dialogue is then marked as successful if both (1) the offered entity matches, and (2) the system answered all the associated information requests (e.g. *what is the address?*) from the user. We computed the BLEU scores on the template-like output sentences before lexicalising with the entity value substitution.

Table 2 shows the result of the corpus-based evaluation averaging over 5 randomly initialised networks. The *Baseline* block shows two baseline models, the first is a simple turn-level sequence to sequence model [18] while the second one introduces an additional recurrence to model the dependency on the dialogue history following Serban et al (2015) [14]. As can be seen, incorporation of the recurrence improves the BLEU score. However, baseline task success and matching rates cannot be computed since the models do not make any provision for a database.

The *Variant* block of Table 2 shows two variants of the proposed end-to-end model. For the first one, no requestable trackers were used, only informable trackers. Hence, the burden of modelling user requests falls on the intent network alone. We found that without explicitly modelling user requests, the model performs very poorly on task completion ($\sim 30\%$), even though it can offer the correct entity most of the time($\sim 90\%$). More data may help here, however, we found that, as shown by the results for the full model below, the incorporation of an explicit internal semantic representation is more efficient and extremely effective. For the second variant, the LSTM intent network is replaced by a CNN. This achieves a very competitive BLEU score but task success is still quite poor ($\sim 58\%$ success). We think this is because the CNN encodes the intent by capturing several, local features but lacks the global view of the sentence, which may easily result in an unexpected overfit.

The *Full model* block shows the performance of the proposed model with different decoding strategies. The first row shows the result of decoding using the average likelihood term (Equation 13) while the second row uses the *weighted* decoding strategy (Equation 14). As can be seen, the *weighted* decoding strategy does not provide a significant improvement in BLEU score but it does greatly improve task success rate ($\sim 3\%$). The R_t term contributes the most to this improvement because it injects additional task-specific information during decoding. Despite this, the most effective and elegant way to improve the performance is to use the attention-based mechanism (*+att.*) to dynamically aggregate the tracker beliefs (Section 2.4). It gives a slight improvement in BLEU score (~ 0.01) and a big gain on task success ($\sim 5\%$). Finally, we can improve further by incorporating *weighted* decoding with the attention models (*+ att. + weighted*).

As an aside, we used t-SNE [20] to produce a reduced dimension view of the action embeddings $\mathbf{o}_t$, plotted and labelled by the first three generated output words (full model w/o attention). The figure is shown as Figure 3. We can see clear clusters based on the system intent types, even though we did not explicitly model them using dialogue acts.

Human evaluation In order to assess operational performance, we tested our model using paid subjects recruited via Amazon Mechanical Turk. Each judge was asked to follow a given task and to rate the model's performance. We assessed the subjective success rate, and the perceived comprehension ability and naturalness of response on a scale of 1 to 5. The full model with attention and weighted decoding was used and the system was tested on a total of 245 dialogues. As can be seen in Table 3, the average subjective success rate was 98%, which means the system was able to complete the majority of tasks. Moreover, the comprehension ability and naturalness scores both averaged more than 4 out of 5. (See Appendix C for some sample dialogues in this trial.)

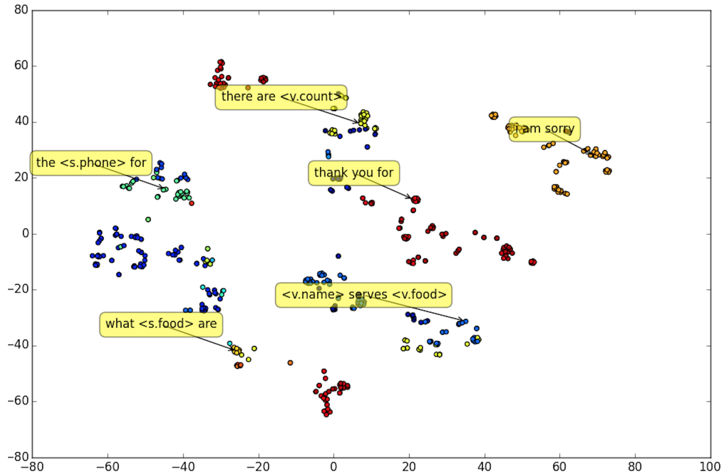

Figure 3: The action vector embedding $\mathbf{o}_t$ generated by the end-to-end model w/o attention. Each cluster is labelled with the first three words the embedding generated.

Table 3: Human assessment of the end-to-end system. The rating for comprehension/naturalness are both out of 5.

Metric	N2N
Success	98%
Comprehension	4.11
Naturalness	4.05
# of dialogues:	245

Table 4: A comparison of the end-to-end system with a rule-based modular system.

Metric	N2N	Modular	Tie
Subj. Success	96.95%	95.12%	-
Avg. # of Turn	3.95	4.54	-
Comparisons(%)			
Naturalness	46.95*	25.61	27.44
Comprehension	45.12*	21.95	32.93
Preference	50.00*	24.39	25.61
Performance	43.90*	25.61	30.49

* p <0.005, # of comparisons: 164

We also ran comparisons between the end-to-end model and a handcrafted, modular baseline system consisting of a handcrafted semantic parser, rule-based policy and belief tracker, and a template-based generator. The result can be seen in Table 4. Over the 164 dialogues tested, the end-to-end system (*N2N*) was considered better than the modular system (*Modular*) on all the metrics compared. Although both systems achieved similar success rates, the end-to-end system (*N2N*) was more efficient and provided a more engaging conversation (lower turn number and higher preference). Moreover, the comprehension ability and naturalness of the end-to-end system were also rated higher, which suggests that the learned system was perceived as being more natural than the hand-designed system.

5 Conclusions and Future Work

This paper has presented a novel neural network-based framework for task-oriented dialogue systems. The model is end-to-end trainable using two supervision signals and a modest corpus of training data. The paper has also presented a novel crowdsourced data collection framework inspired by the Wizard-of-Oz paradigm. We demonstrated that the pipe-lined parallel organisation of this collection framework enables good quality task-oriented dialogue data to be collected quickly at modest cost.

The experimental assessment of the end-to-end dialogue system showed that the learned model can interact efficiently and naturally with human subjects to complete an application-specific task. To the best of our knowledge, this is the first end-to-end neural network-based dialogue system that can conduct meaningful dialogues in a task-oriented application.

However, there is still much work left to do. Our current model is a text-based dialogue system, which can not directly handle noisy speech recognition inputs nor can it ask the user for confirmation when it is uncertain. Indeed, the extent to which this type of model can be scaled to much larger and wider domains remains an open question which we hope to pursue in our further work.

Acknowledgements

Tsung-Hsien Wen and David Vandyke are supported by Toshiba Research Europe Ltd, Cambridge Research Laboratory.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[2] M. Gašić, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young. On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *ICASSP*, 2013.

[3] M. Henderson. Machine learning for dialog state tracking: A review. In *Machine Learning in Spoken Language Processing Workshop*, 2015.

[4] M. Henderson, B. Thomson, and S. Young. Word-based dialog state tracking with recurrent neural networks. In *SIGdial*, 2014.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[6] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *ACL*, 2014.

[7] J. F. Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Transaction on Information Systems*, 1984.

[8] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

[9] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. In *NAACL-HLT*, 2016.

[10] W. Ling, E. Grefenstette, K. M. Hermann, T. Kocisky, A. Senior, F. Wang, and P. Blunsom. Latent predictor networks for code generation. *arXiv preprint:1603.06744*, 2016.

[11] N. Mrkšić, D. Ó Séaghdha, B. Thomson, M. Gašić, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young. Multi-domain Dialog State Tracking using Recurrent Neural Networks. In *ACL*, 2015.

[12] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[13] I. V. Serban, R. Lowe, L. Charlin, and J. Pineau. A survey of available corpora for building data-driven dialogue systems. *CoRR*, abs/1512.05742, 2015.

[14] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau. Hierarchical neural network generative models for movie dialogues. In *AAAI*, 2015.

[15] L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. In *ACL*, 2015.

[16] P.-H. Su, D. Vandyke, M. Gasic, D. Kim, N. Mrksic, T.-H. Wen, and S. J. Young. Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. In *Interspeech*, 2015.

[17] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks. In *NIPS*, 2015.

[18] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[19] D. R. Traum. *Foundations of Rational Agency*, chapter Speech Acts for Dialogue Agents. Springer, 1999.

[20] L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. *JMLR*, 2008.

[21] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In *NIPS*, 2015.

[22] T.-H. Wen, M. Gašić, D. Kim, N. Mrkšić, P.-H. Su, D. Vandyke, and S. Young. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *SIGdial*, 2015.

[23] T.-H. Wen, M. Gašić, N. Mrkšić, P.-H. Su, D. Vandyke, and S. Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*, 2015.

[24] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi. Spoken language understanding using long short-term memory neural networks. In *SLT*, 2014.

[25] P. Yin, Z. Lu, H. Li, and B. Kao. Neural enquirer: Learning to query tables. *arXiv preprint:1512.00965*, 2015.

[26] S. Young, M. Gašić, B. Thomson, and J. D. Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 2013.

A Wizard-of-Oz data collection websites

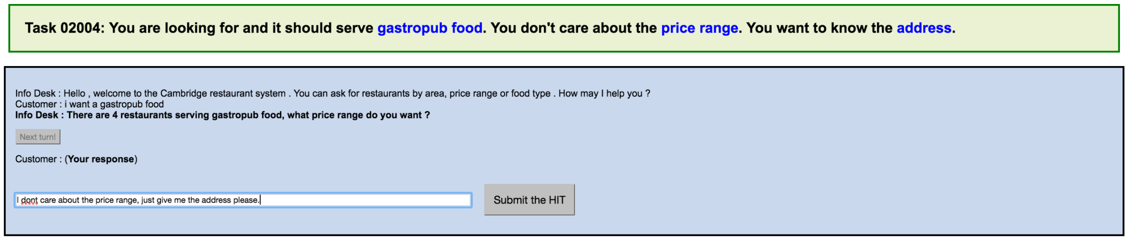

Figure 4: The user webpage. The worker who play users is given a task to follow. For each mturk HIT, he/she needs to type in an appropriate sentence to carry on the dialogue by looking at both the task description and the dialogue history.

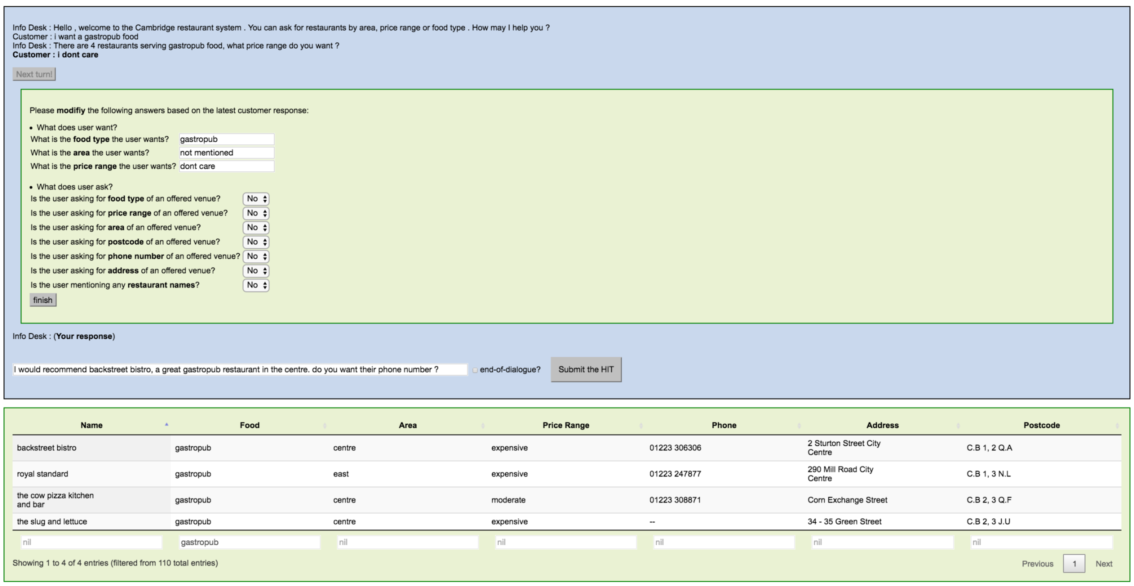

Figure 5: The wizard page. The wizard's job is slightly more complex: the worker needs to go through the dialogue history, fills in the form (top green) by interpreting the user input at this turn, and type in an appropriate response based on the history and the DB result (bottom green). The DB search result is updated when the form is submitted. The form can be divided into informable slots (top) and requestable slots (bottom), which contains all the labels we need to train the trackers.

B Scoring Table

Table 5: Additional R_t term for delexicalised tokens when using weighted decoding (Equation 14). *Not observed* means the corresponding tracker has a highest probability on either *not mentioned* or *dontcare* value, while *observed* mean the highest probability is on one of the categorical values. A positive score encourages the generation of that token while a negative score discourages it.

Delexicalised token	Examples	R_t (*observed*)	R_t (*not observed*)
informable slot token	<s.food>, <s.area>,...	0.0	0.0
informable value token	<v.food>, <v.area>,...	+0.05	-0.5
requestable slot token	<s.phone>,<s.address>,...	+0.2	0.0
requestable value token	<v.phone>,<v.address>,...	+0.2	0.0

C Sample dialogues

Table 6: Some samples of real conversational logs between online judges and the end-to-end system.

Human	: Restaurant in any area that serves Vietnamese food
Machine	: thanh binh serves vietnamese food type , is there anything else i can help you with ?
Human	: What is the phone number and post code
Machine	: the number is 01223 362456 and the area code is c.b 3, 0 a.f . is there anything else i can do for you ?
Human	: Good bye
Machine	: thank you for using the cambridge restaurant system .
Human	: i am looking for moderately priced halal food
Machine	: i am sorry , there are no halal restaurants in the moderate price range . would you like something else ?
Human	: how about european food
Machine	: galleria serves european food type and is in the moderate price range .
Human	: address and postcode
Machine	: the address is 33 bridge street and the area code is c.b 2, 1 u.w . is there anything else i can help you with today ?
Human	: thanks
Machine	: galleria is a european restaurant in the moderate price range . would you like their phone ?
Human	: thank you
Machine	: thank you for using the cambridge restaurant system . good bye
Human	: Hi, I'm hungry for some Indonesian food
Machine	: i am sorry , there are no indonesian restaurants . would you like a different food ?
Human	: Oh no. How about Indian?
Machine	: there are 9 restaurants serving indian types of food . do you have a preference on side of town ?
Human	: East side please
Machine	: sitar tandoori is a indian restaurant in the east side of town . would you like their information ?
Human	: Yes, the address and phone number please
Machine	: sitar tandoori is located at 43 high street cherry hinton cherry hinton and their phone number is 01223 249955 .
Human	: Okay, thank you
Machine	: thank you , goodbye .
Human	: hello
Machine	: what cuisine would you like ?
Human	: afghan
Machine	: i am sorry , there are no afghan restaurants . would you like something else ?
Human	: korean
Machine	: little seoul serves korean food , it is located at 108 regent street city centre .
Human	: is it expensive?
Machine	: little seoul serves korean food and is in the expensive price range .
Human	: what area is that
Machine	: little seoul is located in the centre area of town . would you like their number ?
Human	: yes
Machine	: the telephone number is 01223 308681 . is there anything else i can help you with today ?
Human	: no goodbye
Machine	: thank you for using the cambridge restaurant system , goodbye .
Human	: I want to find a moderately priced restaurant and it should be in the north part of town
Machine	: there are 2 moderate restaurants in the north side of town . would you like the address and telephone number ?
Human	: what is the name address and phone number
Machine	: the nirala is located at 7 milton road chesterton and their phone is 01223 360966 . is there anything else i can do for you today ?
Human	: thank you goodbye
Machine	: thank you for using the cambridge restaurant system . good bye .
Human	: I want to find a restaurant in the east part of town that serves creative food
Machine	: i am sorry , there are no creative restaurants in the east side of town . would you like a different restaurant ?
Human	: how about indian food?
Machine	: curry prince is a indian restaurant in the east of town . would you like their phone number ?
Human	: yes, may i have the address and phone number?
Machine	: curry prince is located at 451 newmarket road fen ditton and their phone is 01223 566388 . is there anything else i can help you with today ?
Human	: no, thank you goodbye
Machine	: thank you for using the cambridge restaurant system . good bye
Machine	: what food are you interested ?
Human	: english
Machine	: i am sorry , there are no english restaurants . would you like to try a different food type ?
Human	: chinese
Machine	: there are 8 restaurants serving chinese type of cuisine . do you have a preference of area ?
Human	: south part
Machine	: the good luck chinese food takeaway is a chinese restaurant in the south of town . would you like the phone number ?
Human	: no. just address and price range
Machine	: the good luck chinese food takeaway is located at 82 cherry hinton road cherry hinton . is there anything else i can help you with ?
Human	: what is their price range?
Machine	: the good luck chinese food takeaway is expensively priced chinese restaurant in the south area . would you like more information ?
Human	: no goodbye
Machine	: thank you for using the cambridge restaurant system . good bye .
Human	: Is there a gastropub anywhere around here?
Machine	: there are 3 gastropub restaurants . would you like more information ?
Human	: Is there one on the west side?
Machine	: i am sorry , there are no gastropub restaurants in the west side of town . would you like something else ?
Human	: How about east?
Machine	: royal standard is a gastropub restaurant in the east of town .
Human	: What's the address?
Machine	: the address is 290 mill road city centre .
Human	: Thanks, bye
Machine	: thank you for using the cambridge restaurant system . good bye