Deep Reinforcement Learning for Task-Oriented Dialogue

Nnamdi Iregbulem
nnamdi@stanford.edu

Sahil Yakhmi
syakhmi@stanford.edu

Abstract

Despite major advances in NLP over the past decade, most deployed task-oriented chatbots are implemented as a hand-coded Finite State Machine (FSM) architecture, where user utterances matching pre-defined patterns transition the dialogue from one state to the next. The open-source framework Rasa (rasa.com) has popularized a new architecture for chatbots with flexible dialogue states and transitions that can be learned [1] rather than pre-defined. However, there exists a training data problem within this new architecture. Rasa requires training data for both NLU (intent recognition and parsing) and dialogue state management, but hand-annotating examples for training is a time-consuming and expensive process. Moreover, there is no simple way to leverage the conversation data generated by chatbots deployed in the field. In this paper, we demonstrate the viability of end-to-end Deep Reinforcement Learning for automatic and continuous learning for a chatbot agent. We describe the process of building a machine-learned action policy model for a chat agent to approximate the behavior of a deployed system, heuristics and conversational cues to extract rewards from unannotated conversation logs, and finally the use of Reinforcement Learning to improve performance over our baseline.

1 Introduction

Recent advancements in machine learning and neural networks specifically have fostered an explosion of interest and activity around the creation of dialogue agents capable of holding meaningful text-based interactions with human users. One of the core use cases for such a dialogue agent is chatbots, frequently deployed in customer service contexts where a virtual agent replaces a human, lowering the costs of call centers and customer support operations.

The chatbot context is special in that the virtual agent is not simply trying to hold a conversation with human users, but also generally has a goal—a task to be completed, for example, helping a user check a flight itinerary.

However, a significant number of commercial chatbots continue to rely on relatively simplistic approaches such as regex matching and hand-coded Finite State Machines (FMSs). The open-source framework Rasa (rasa.com) has popularized a new architecture for chatbots with flexible dialogue states and transitions that can be learned [1] rather than predefined. However Rasa retains discrete components for understanding text (NLU), tracking state (DST), and action policy (DM), and requires difficult to obtain hand-annotated training data examples for all components. End-to-End and Reinforcement Learning can offer an improvement upon the previous paradigm.

We explore semi-automated Reinforcement Learning (RL) with conversational cues as an avenue for further performance gains in task-oriented dialogue. Demonstrating the ability of deep reinforcement learning to improve chatbot conversations quality and task completion challenges the modeler to design reward functions that reinforce desired behaviors, and demote undesired ones.

Although we used scaffolding from class assignments, no other external code was used in the project.

2 Related Work

Even prior to the recent rise of neural network-based approaches to dialogue, researched has explored the applicability and potential of reinforcement learning for this task. Singh et al. [6] propose a dialogue system designed in a Markov Decision Process framework, enabling the application of reinforcement learning to train a dialogue policy.

Previous research in this space has attempted to combine reinforcement learning with traditional architectures for neural network-based natural language generation to varying degrees of success. (Li et. al., 2016) construct a virtual chatbot capable of generating coherent open domain conversational dialogue using deep reinforcement learning to reinforce supplemental objectives such as ease of answering, information flow, and semantic coherence in simulated dialogues.[4] These go beyond typical cross-entropy-based loss functions which were found to be insufficient in preventing generic ("I don't know") or repetitive utterances which commonly plague sequence-to-sequence models using maximum likelihood estimation per (Sutskever et. al., 2014).

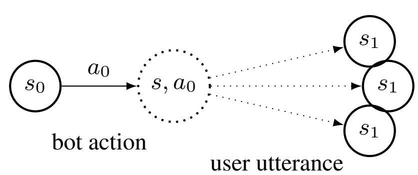
(Shah et. al., 2016) explores a similar task-oriented domain as ours, and explores fully generic techniques that require only a known goal state, but simulates a direct user feedback function [-1, 1] for each system dialog act—an unrealistic assumption in most real-world settings.[5].

Reinforcement learning has been shown to be especially helpful in highly noisy contexts, such as conversation, where the "correct" response is highly debatable and multiple potential utterances could be considered valid. (Su et. al.) demonstrate a two-stage learning process whereby a neural network is trained in traditional supervised fashion from a dataset of dialogue and subsequently improved via reinforcement learning. This reinforcement-based augmentation improves performance in both simulated dialogue and conversation with human subjects, who assign dialogue with the reinforcement agent a higher quality score on a 6-point Likert scale relative to the standalone supervised model.[7]

In addition to conversational dialogue, recent work attempts to build task-oriented dialogue systems, which are of particular relevance to chatbot contexts. (Wen et. al.) use data from the restaurant search domain to train an end-to-end goal-oriented dialogue system that both generates plausible dialogue with interspersed "slots" which it fills in with relevant contextual information using slot-value pairs from an external database. The model achieves a high level in both identifying queries from the user to be answered and correctly retrieving the relevant information.[8]

3 Approach

Our approach was to model task-oriented dialogue as a Markov Decision Process (MDP) with infinite state space, allowing for the application of Reinforcement Learning (RL) techniques.



In this model of task-oriented dialogue, at every point in a conversation just before the chatbot speaks, the state can be thought of as the entire dialogue up until that point. At each time-step, the bot chooses one of a finite set of pre-programmed actions to take. These actions might involve performing a computation, or interacting with a database, for example. After the bot performs an action, the dialogue will then transition non-deterministically to the next state, based on the probability distribution over possible user responses:

$$P_a(s, s') = P(s_{t+1} = s' | s_t = s, a_t = a) \tag{1}$$

Our overarching task is to learn an optimal policy, such that we can estimate:

$$Q_{opt}(s, a) = ... \tag{2}$$

We begin by learning a $Q_{baseline}(s, a)$ distribution over possible actions in a supervised fashion. The goal of this first stage is not to compute the optimal Q distribution, but rather to obtain a baseline that approximates the behavior of an existing real-world system. We used an LSTM-based model with a softmax output layer and cross-entropy loss to fit a model to data obtained from a real-world chatbot system. The construction and performance of this baseline model is reported in section 4.1.

We selected the architecture of this model based on the Seq2Seq architecture and state-space representation used in (Li et. al., 2016) [4], with the modification of targeting a probability distribution of actions rather than natural language output, and adapted from open-domain dialogue to task-oriented dialogue. As in (Li et. al., 2015) [3], in this configuration, the model receives the prior two conversation turns as input. In our second baseline experiment, we extended this state representation and model architecture in a novel way to fit the task-oriented domain. This approach is detailed in section 4.3.

After learning this baseline system, we next attempt to approximate Q_{opt} by running a Q-learning style training algorithm with (s, a, r) triples extracted algorithmically from conversation logs. These reward heuristics attempt to capture whether or not a bot was successful in achieving the user's desired outcome, and seek to reinforce success and demote incorrect actions. This work is detailed in section 4.4.

4 Experiments

4.1 Baseline: DSTC

4.1.1 Data

Although the majority of our experiments were performed on a real-world dataset obtained in partnership with Yalochat, Inc., we constructed our baseline model prior to obtaining access to this dataset using the Dialogue State Tracking Challenge 2 (DSTC2) dataset. This dataset, consisting of dialogues between users and an automatic restaurant search system, is annotated with dialogue acts / NLU representations via a crowd-sourced procedure.

Our data consisted of 28168 utterances in the training set, and 3054 utterances in the test set, or 2118 total conversations.

We performed substantial preprocessing of the datasets labels to better simulate the task we hoped to perform on the Yalochat dataset. This resulted in 14 classes, each presented with an example from the dataset in Table 6. Example data is shown in Table 5.

4.1.2 Evaluation Method

We did not perform an end-to-end evaluation of the DSTC Baseline experiment, as our task was simply to prove that an LSTM sequence model can fit to the behavior of an existing, hand-engineered dialogue agent.

Our evaluation of this baseline focused on measuring how well the model approximated the behavior of the existing system. We calculated accuracy, precision and recall for each possible action.

4.1.3 Experiment Details

For this baseline model, we chose a simple model that consists of 50-dim Glove word embeddings, fed into a vanilla LSTM (hidden size: 50) encoder, whose last hidden state is fed to a linear output layer, possibly with one or more hidden layers. Softmax is then calculated over the output layer, generating a probability distribution over each possible action within the action space:

$$Model(s_t) = Q_t \in \mathbb{R}^{\|A\|} \tag{3}$$

where

$$s_t = (b_{t-1,1}, b_{t-1,2}, ...b_{t-1,n}, < /s >, u_{t-1,1}, u_{t-1,2}, ...u_{t-1,m}) \tag{4}$$

where b_t represents tokens of a bot utterance, </s> is a separator token and u_t represents tokens of a user response to the bot utterance (i.e., the last two turns of dialogue).

With a learning rate of 0.001, we found that cross entropy loss quickly converged to zero within a minute or two on the training dataset, and prediction accuracy was quite high on the dev set.

4.1.4 Results

Overall labeling accuracy of our baseline model compared to the DSTC dialogue system actions are presented in Table 1. These statistics only represent fit to the existing system agent, not correctness. Precision and recall for each class are presented in Table 7.

Table 1: DSTC Baseline Results

Dataset	dev
Accuracy	98.1%
Weighted F1	0.980

4.2 Baseline: Yalochat

4.2.1 Data

For subsequent experiments we used data obtained in partnership with Yalochat, Inc. We obtained 30 days of conversation data from Yalochat, Inc. of real-world chatbot logs from the Aeroméxico Facebook / WhatsApp chatbot.

Due to the unprocessed nature of this dataset, we faced many challenges in preparing the data for experiments. We initially planned to use 2.5 months of data, however a bug in the company's logging systems corrupted over half of the data. Moreover, of the data we were able to obtain, a significant percentage of the data was found to be logging of bogus events, or otherwise affected by logging bugs.

Starting with approximately 75,000 conversations, we filtered the data for:

1. Bogus event data (comprising the majority of log data by bytes)
2. Conversations where one side was entirely missing

After this filtering, we were left with only 14,647 conversations and 284,027 utterances, of which we set aside 1,464 conversations for the dev and test sets each, and left 11,719 for training data.

For each conversation, we obtained a transcript of the dialogue, where each utterance was labeled with the agent (user or bot), and each user utterance was labeled with the state transition that the production Aerobot performed after processing this input (internally, Yalochat models dialogue as a hand-designed FSM). Initially, there were approximately 140 unique transition labels.

After examining the training data, we found a slight conceptual mismatch between our MDP model of dialogue and Yalochat's recorded state transitions. In particular, the recorded Yalo bot action could be conditionally based upon arbitrary logic e.g., validation logic or a database lookup. Because modeling these tasks was a nongoal for our model, we merged many of the system actions. For example, we mapped both *no-found-flight-by-pnr* and *find-flight-by-pnr* to the same label, as the model would not be able to perform the associated database lookup. After adjusting for this and other issues, we were left with 98 labels.

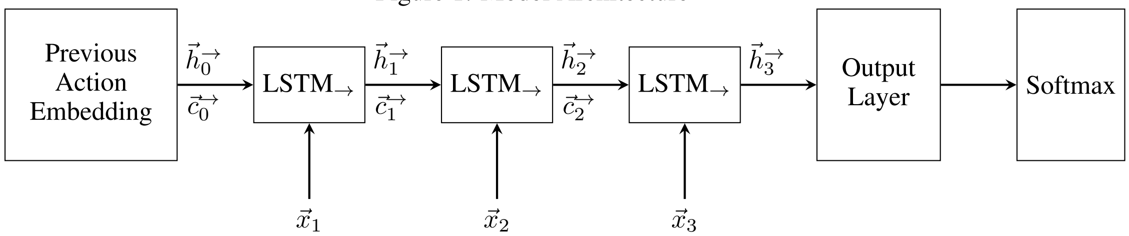
4.2.2 Evaluation method

Because this was not our final baseline model, we chose not to perform end-to-end evaluation for the correctness of the labels produced by the model, instead performing only the same kind of multi-class fit analysis as we performed for the DSTC Baseline model.

4.2.3 Experiment Details

For this baseline model, we had to make a few minor adjustments to the DSTC model for it to run on the new dataset. In particular, we needed to swap the English-language Glove word embeddings

Figure 1: Model Architecture



we used in the prior model for Spanish-language word embeddings. We elected to use GloVe-based word embeddings trained by the Spanish Billion Word Corpus and Embeddings project, which were available only in a 300-dim variant.[2] We kept the LSTM hidden state size the same at 50 (although future work would involve tweaking this hyperparameter), and our output layer grew in size to 98.

4.2.4 Results

Compared to the state transition labels from the original Yalo system, our model achieved relatively good accuracy:

Table 2: Yalochat Baseline Results

Dataset	dev	test
Accuracy	91.4%	91.4%
Weighted F1	0.913	0.910

Classification statistics for each action on the test data are reported in Table 8.

4.3 Baseline + Improved State Representation: Yalochat

Building on the previous baseline model, we made some improvements to the model architecture and state representation. This experiment became our primary baseline for future reinforcement learning experiments.

4.3.1 Evaluation method

We chose to perform an end-to-end evaluation as a Win/Loss ratio computed over diffs on test data between the original Yalo system and our ML system. In this human eval task raters were asked to rate system A or system B as better, or if the change was neutral (both acceptable, or both incorrect). Agreement between multiple raters was required. We initially intended for this rating task to be done through crowdsourced labeling, however we quickly realized that the task was too difficult without significantly cleaning up the data, and writing a comprehensive set of rating guidelines. Instead we rated diffs on a blind basis within our team. Designing a robust crowdsourced task is part of future planned work.

In addition, we also computed classification evaluation statistics comparing the output of our model to the original Yalo system, as described in Section 4.2.2.

4.3.2 Experiment Details

Noticing that the last two turns of dialogue did not appear to be enough to capture dialog state, and that in many situations, this state could ambiguously represent the conversation context, we experimented with adding the previous bot action to our state representation:

$$s_t = (a_{t-1}, b_{t-1,1}\,, b_{t-1,2}\,, ...b_{t-1,n}\,, < /s >, u_{t-1,1}\,, u_{t-1,2}\,, ...u_{t-1,m}\,) \tag{5}$$

For each action, we learned an embedding matrix with which to initialize the hidden state h_0 and cell state c_0 of the LSTM encoder. Essentially we modified our earlier Markov assumption that the dialogue state depends on the last two turns of dialogue text to an assumption that the dialogue state depends on the action that generated the bot's utterance as well, encoding more information about the prior conversation state. See Figure 1.

4.3.3 Results

Classification Statistics

Compared to the state transition labels from the original Yalo system, our model achieved slightly higher accuracy than our previous baseline model, converging faster in training as well:

Table 3: Yalochat Enhanced Baseline Results

Dataset	dev	test
Accuracy	93.1%	92.7%
Weighted F1	0.930	0.924

Multi-class results are presented in Table 9.

End-to-End

A random sample of 50 diffs with the Yalo test data resulted in 10 wins, 18 losses, yielding a W/L ratio of 0.56. This result suggests that despite relatively good fit to the existing bot's behaviors, the learned model introduced more regressions than it was able to fix through generalization over a (buggy) legacy architecture.

4.4 Reinforcement Learning: Yalochat

4.4.1 Data

Our next experiment sought to improve upon the baseline models using Reinforcement Learning, using automated conversational analysis heuristics to generate "good" (s, a) examples to reward, and "bad" (s, a) examples to demote. We focused on actions we could give a positive reward, and kept our heuristics relatively simple and unoptimized, to avoid overfitting to the train / dev datasets and provide a good foundation for future generalization. As a result, our heuristics were relatively low precision. We implemented three such heuristics:

1. Default state (unrecognized input) => Rephrased Action Sequences
2. Q&A Dissatisfaction (explicit negative feedback) => Rephrased Action Sequences
3. Q&A / Action Equivalence (jump directly from correct FAQ answer to system action)

Example dialogues are presented in Figure 2.

4.4.2 Evaluation method

We used the end-to-end evaluation methodology described in Section 4.3.1. In addition to evaluating diffs between the original Yalo system and our RL models, we also evaluated diffs between our enhanced baseline model and the RL models.

4.4.3 Experiment Details

We utilized a simplified version of Q-Learning for our experiments. In the general case of Q-Learning with function approximation, we calculate a target Q value according to a reward function $R(s, a, s')$ summed with discounted future rewards:

$$Q_{target} \approx R(s, a, s') + \gamma \max_{a'} Q(s', a') \tag{6}$$

We would then use an optimization algorithm to minimize the difference between Q_target and Q for each RL example. In the case of our experiments, we determined it would be infeasible to approximate $\max_{a'} Q(s', a')$ without simulating the user agent as well (future work). As such we elected to fully discount future rewards (including the final goal states) with $\gamma = 0$. Although this limits the power of our approach, it dramatically simplifies training.

We approximated Q_{target} as a one-hot probability distribution based on pattern match with our reward heuristics. The resulting algorithm is similar to direct supervised learning of the Q function. Because our reward heuristics provided a distribution of examples that was both sparse and biased, we faced an additional problem: even with a low learning rate, the model would quickly forget what it had learned in the baseline training in order to maximize the RL reward. We solved this problem by alternating training one batch of RL examples with one batch of examples from the baseline training set. Additionally, we used a smaller learning rate of 0.00005, $\frac{1}{20}$ of the rate we used in training the baseline, to encourage the training process to "fine-tune" rather than start from scratch.

4.4.4 Results

Classification Statistics

Compared to the state transition labels from the original Yalo system, our model achieved high accuracy, but lower than the enhanced baseline system:

Table 4: Yalochat RL Results

Dataset	dev	test	dev-baseline	test-baseline
Accuracy	91.8%	91.6%	93.1%	92.7%
Weighted F1	0.916	0.912	0.930	0.924

The purpose of the RL experiments was to fix bugs in the original system, so a lower accuracy is not necessarily cause for concern. Multiclass results are presented in Table 10.

End-to-End

A random sample of 100 diffs with the existing Yalo bot on the test dataset resulted in 34 wins and 30 losses, yielding a slightly positive W/L ratio of 1.13. We also evaluated a random sample of 100 diffs between the enhanced baseline, and the RL system, resulting in 45 wins and 28 losses, yielding a positive W/L ratio of 1.61.

5 Analysis

5.1 Effectiveness of Reinforcement

In analyzing our RL system's performance, there was one question of primary interest—did our model learn the behaviors we were trying to reinforce?

Qualitatively, selecting arbitrary comparable examples from the dev set and analyzing the model's perforce shows that although reinforcement did not solve always solve the problem, it did boost the probability of the desired action on unseen data. One example is presented below (correct action: *FS-Arrival*—flight search), with several more in Figure 8:

$$s = (\quad \text{QnA-Question}, \quad \text{"Escribe tu pregunta en un mensaje a continuación . Por ejemplo, : ¿ Puedo viajar con mi perro ?"} \quad \text{"Cuanto vale el doblete de Oaxaca a Tijuana"} \quad) \tag{7}$$

a	FS-Arrival	QnA-QuestionScript	buenfin1a-yes	default
$Model_{RL}(s, a)$	0.918	0.078	0.001	0.001
$Model_{baseline}(s, a)$	0.000	0.989	0.005	0.002

Nevertheless, the RL process also created losses through overgeneralization, as in the following example (from the W/L test set eval), in which a nonsense input is newly interpreted as a flight search:

$$s = (\quad \text{default}, \quad \text{"Todavía no puedo responder esa pregunta , <name> ."}, \quad \text{"Jajajaa"} \quad) \tag{8}$$

5.2 Other Systematic Losses

One downside of our model architecture is that it was not always good at distinguishing who said what. For example, the model learned an association between the word "gracias" and a successful resolution of a conversation (*buenfin1a-yes*). However, in the following example, it likely misinterpreted the bot's utterance as the user's. A solution is discussed in the conclusion.

$$s = \left(\text{humanLoop}, \text{"¡ Gracias por contactarnos ! Aerobot sigue a tus órdenes ."}, \text{"En las condiciones que me indica , claramente en el punto 7.3 esta señalada la responsabilidad del transportista"} \right) \tag{9}$$

In addition, the model:

1. Consistently underlabels "Bullying" (cursing) for lack of a good distribution of examples
2. Has poor performance for actions with ambiguous semantics (e.g., Welcome, Hi, Menu)
3. May have trouble learning "default" as a catch-all action when no other action is appropriate

6 Conclusion

This paper presents the results of multiple experiments leveraging deep reinforcement learning for task-oriented dialogue in a real-world setting (the Aeroméxico Facebook Messenger / WhatsApp chatbot).

In our experiments, we have demonstrated the viability of approximating the behavior (action policy) of a hand-tuned, chatbot FSM with a deep neural architecture. Moreover, we have demonstrated the effectiveness of semi-automated reinforcement learning in tuning the baseline neural model to a level of performance above that of the original hand-tuned system.

These experiments represent a step in the direction of a self-optimizing chat agent, whose conversation flows do not have to be fully designed by a UX team. Although our use of an ML-based action policy is inspired by Rasa, we have shown the viability of end-to-end learning of the action policy on raw text inputs (without the use of an intermediate NLU knowledge representation), as well as the viability of reinforcement learning as a scalable learning technique within this architecture.

6.1 Future work

6.1.1 Model Architecture Improvements

One of the biggest drawbacks to the current model architecture is its handling of OOV tokens. Our domain is full of nonstandard spellings, emojis, and slang terms, and we believe our task would greatly benefit from a hybrid character/word encoding architecture. This would help with understanding confirmation codes, names, flight numbers, and a host of other common inputs.

In addition, we plan to experiment with more complex architectures than a vanilla LSTM conditioned on previous bot action. For example, we could encode bot actions, bot utterances, and user utterances into separate vectors, and feed them through a convolutional or fully connected layer to obtain our action distribution.

6.1.2 Entity Extraction

Many bot actions require data (for example a flight search requires dates and a destination), and ideally this data could be extracted directly from encoded representation of conversation state. We plan to experiment with an attention mechanism to perform slot filling.

6.1.3 Simulation

We should be able to achieve better distribution of rewards within our state space through simulation of the user agent with a separate neural model, and we can increase the power of our approach by guiding our RL agent toward future goals states. This would have substantial benefits for learning.

7 Additional information

Mentor: Annie Hu

References

[1] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open source language understanding and dialogue management. *CoRR*, abs/1712.05181, 2017.

[2] Cristian Cardellino. Spanish Billion Words Corpus and Embeddings, March 2016.

[3] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *CoRR*, abs/1510.03055, 2015.

[4] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *CoRR*, abs/1606.01541, 2016.

[5] Pararth Shah, Dilek Hakkani-Tur, and Larry Heck. Interactive reinforcement learning for task-oriented dialogue management. 2016.

[6] Satinder Singh, Michael Kearns, Diane Litman, and Marilyn Walker. Reinforcement learning for spoken dialogue systems. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 956–962, Cambridge, MA, USA, 1999. MIT Press.

[7] Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. Continuously learning neural dialogue management. *CoRR*, abs/1606.02689, 2016.

[8] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. A network-based end-to-end trainable task-oriented dialogue system. *CoRR*, abs/1604.04562, 2016.

8 Appendix

Table 5: Sample of DSTC Conversation

Agent	Text	Label
system	Hello , welcome to the Cambridge restaurant system? You can ask for restaurants by area , price range or food type . How may I help you?	welcomemsg
user	expensive restaurant	inform
system	What part of town do you have in mind?	request
user	south noise	inform
system	What kind of food would you like?	request
user	expen	
system	What kind of food would you like?	request
user	french	inform
system	I'm sorry but there is no french restaurant in the south of town	canthelp
user	italian	inform
user	frankie and bennys is a great restaurant serving expensive italian food in the south of town .	offer

Table 6: Classes

Class	Example
affirm	yea
thankyou	thank you good bye
inform	cheap restaurant mediterranean food
request	address and phone number
deny	not european fusion
negate	no british food
hello	hi im looking for a moderately priced restaurant in the south part of town
bye	bye
confirm	is it in the north part of town
repeat	can you repeat that
reqmore	more
reqalts	how about french
restart	start again
ack	thatll do dear whats the address of that venue

Table 7: DSTC Baseline Results

Class	# in dev Set	Precision	Recall
affirm	56	96.5%	98.2%
thankyou	218	99.1%	99.5%
inform	584	99.5%	99.7%
request	419	99.0%	99.0%
deny	2	N/A	0%
negate	19	82.6%	100%
hello	7	100%	71.4%
bye	14	100%	85.7%
confirm	7	100%	85.7%
repeat	1	33.3%	100%
reqmore	1	100%	100%
reqalts	90	96.7%	96.7%
restart	0	N/A	N/A
ack	2	N/A	0%

Table 8: Yalo Baseline Results

Action	Count	Precision	Recall	F1
FS-Script-Search-Arrival-Airpots	1514	0.974	0.976	0.975
FS-Arrival	1127	0.980	0.995	0.987
humanLoop	1074	0.880	0.891	0.886
FS-Script-Search-Departure-Airports	1063	0.994	0.992	0.993
FS-PassengersAdults	1018	0.917	0.923	0.920
QnA-QuestionScript	700	0.840	0.910	0.874
FS-Rates	620	0.986	0.995	0.990
FS-PassengersChildren	581	0.993	1.000	0.997
Menu	353	0.773	0.773	0.773
default	343	0.740	0.755	0.747
Welcome	333	0.872	0.775	0.820
QnA-Question	291	0.993	0.990	0.991
QnA-AnswerCorrect	194	0.949	0.954	0.951
FS-SelectDate-Round-Trip-Error	165	0.801	0.806	0.804
Hi	112	0.452	0.375	0.410
menu-flight-status	110	1.000	1.000	1.000
FS-PassengersChildrenNumber	105	0.981	1.000	0.991
QnA-Answer0	97	0.990	0.979	0.984
YourWelcome	89	0.863	0.921	0.891
buenfin1a-yes	86	0.493	0.407	0.446
CheckIn-LastName	66	0.985	0.985	0.985
QnA-Answer1	62	0.984	0.968	0.976
FS-SelectDate-Error	58	0.450	0.310	0.367
CheckIn-PNR	49	0.904	0.959	0.931
QnA-Answer2	45	0.953	0.911	0.932
CheckIn-Start	40	0.950	0.950	0.950
ask-last-name	39	1.000	0.974	0.987
find-flight-by-pnr	30	0.903	0.933	0.918
QnA-Answer1-Help	29	0.641	0.862	0.735
ask-departure-date-flight	23	0.955	0.913	0.933
sfn_pnr	23	0.958	1.000	0.979
FS-PassengersGroups	20	1.000	0.850	0.919
Checkin-SearchPNR	19	0.439	0.947	0.600
sfn_lastname	19	1.000	0.947	0.973
save-destination-city	18	0.400	0.667	0.500
wait-for-destination-iata	18	0.077	0.056	0.065
CheckIn-international-2	17	1.000	1.000	1.000
create_subscription	16	1.000	1.000	1.000
CheckIn-LastNameAgain	13	0.750	0.692	0.720
FS-JustChildren	13	1.000	1.000	1.000
Identity	13	0.222	0.308	0.258
humanCheckIn	10	N/A	0.000	N/A
no-have-number-flight	10	1.000	1.000	1.000
search-weekend-destinations	10	N/A	0.000	N/A
sfn_lambda	8	0.500	0.250	0.333
ask_lastname	7	1.000	1.000	1.000
init	6	N/A	0.000	N/A
Bullying	5	N/A	0.000	N/A
ByeByeMessageTransition	5	1.000	0.200	0.333
CheckIn-MoreN/Ao	5	N/A	0.000	N/A
search_reservation	5	0.222	0.800	0.348
CheckIn-PNRAgain	4	1.000	0.750	0.857
push_notification_default	4	0.000	0.000	N/A
search-beach-destinations	4	N/A	0.000	N/A
tracking-flight	4	0.200	0.250	0.222
world-trace-script	4	N/A	0.000	N/A
parse-departure-date-flight	3	N/A	0.000	N/A

Table 9: Yalo Baseline + Improved State Representation Results

Action	Count	Precision	Recall	F1
FS-Script-Search-Arrival-Airpots	1514	0.973	0.978	0.975
FS-Arrival	1127	0.989	0.991	0.990
humanLoop	1074	0.953	0.953	0.953
FS-Script-Search-Departure-Airports	1063	0.992	0.992	0.992
FS-PassengersAdults	1018	0.943	0.938	0.940
QnA-QuestionScript	700	0.852	0.919	0.884
FS-Rates	620	0.982	0.994	0.988
FS-PassengersChildren	581	0.997	1.000	0.998
Menu	353	0.809	0.754	0.780
default	343	0.774	0.787	0.780
Welcome	333	0.865	0.769	0.814
QnA-Question	291	0.993	0.993	0.993
QnA-AnswerCorrect	194	0.920	0.943	0.931
FS-SelectDate-Round-Trip-Error	165	0.876	0.903	0.890
Hi	112	0.456	0.509	0.481
menu-flight-status	110	0.991	1.000	0.995
FS-PassengersChildrenNumber	105	0.990	0.990	0.990
QnA-Answer0	97	0.969	0.979	0.974
YourWelcome	89	0.913	0.944	0.928
buenfin1a-yes	86	0.542	0.453	0.494
CheckIn-LastName	66	0.970	0.985	0.977
QnA-Answer1	62	0.952	0.968	0.960
FS-SelectDate-Error	58	0.439	0.500	0.468
CheckIn-PNR	49	0.870	0.959	0.913
QnA-Answer2	45	0.977	0.933	0.955
CheckIn-Start	40	0.974	0.925	0.949
ask-last-name	39	0.974	0.974	0.974
find-flight-by-pnr	30	0.935	0.967	0.951
QnA-Answer1-Help	29	0.889	0.828	0.857
ask-departure-date-flight	23	1.000	0.870	0.930
sfn_pnr	23	0.958	1.000	0.979
FS-PassengersGroups	20	1.000	0.850	0.919
Checkin-SearchPNR	19	0.425	0.895	0.576
sfn_lastname	19	1.000	0.947	0.973
save-destination-city	18	0.400	0.778	0.528
wait-for-destination-iata	18	0.133	0.111	0.121
CheckIn-international-2	17	0.944	1.000	0.971
create_subscription	16	1.000	1.000	1.000
CheckIn-LastNameAgain	13	0.778	0.538	0.636
FS-JustChildren	13	1.000	1.000	1.000
Identity	13	0.545	0.462	0.500
humanCheckIn	10	0.000	0.000	N/A
no-have-number-flight	10	1.000	1.000	1.000
search-weekend-destinations	10	N/A	0.000	N/A
sfn_lambda	8	0.000	0.000	N/A
ask_lastname	7	1.000	1.000	1.000
init	6	N/A	0.000	N/A
Bullying	5	N/A	0.000	N/A
ByeByeMessageTransition	5	1.000	0.200	0.333
CheckIn-MoreN/Ao	5	N/A	0.000	N/A
search_reservation	5	0.500	1.000	0.667
CheckIn-PNRAgain	4	1.000	0.750	0.857
push_notification_default	4	N/A	0.000	N/A
search-beach-destinations	4	0.000	0.000	N/A
tracking-flight	4	N/A	0.000	N/A
world-trace-script	4	N/A	0.000	N/A
parse-departure-date-flight	3	0.333	0.667	0.444

Figure 2: RL Heuristic Example Dialogues

1. Default state => Action Sequences

    User: D los cabo a tapachula vuelo redondo => *Default*
    Bot:: Todavía no puedo responder esa pregunta , <name> .
    Bot: <name> , puedo ayudarte con estas opciones , selecciona una o escribe tu consulta en un mensaje . En cualquier momento puedes regresar a este menú, sólo escribe la palabra Aerobot .
    User: Cotizar un vuelo => *FS-Arrival*

2. Q&A Dissatisfaction => Action Sequences

    User: Ando buskando buelo alos cabos => *QnA-QuestionScript*
    Bot: Dame unos segundos ...
    Bot: Constantemente tenemos promociones para viajar . Solo debes ingresar a <url> o suscribirte a nuestro newsletter para que te enteres antes que nadie de las promociones .
    Bot: ¿ Te ayudó mi respuesta ?
    User: No => *QnA-QuestionScript*
    ...
    Bot: En cualquier momento puedes volver a hablar conmigo , sólo escribe la palabra Aerobot . Elige una opción o escribe tu consulta .
    User: Cotizar un vuelo => *FS-Arrival*

3. Q&A Answer / Action Equivalence

    User: Vuelo a Parral cuanto saldria desde cd Juárez => *QnA-QuestionScript*
    Bot: Dame unos segundos ...
    Bot: Para cotizar tu vuelo ... Puedes hacerlo aquí mismo , basta con que escribas *Aerobot y luego en el menú, escribas la información del vuelo que estás buscando*.
    Bot: ¿ Te ayudó mi respuesta ?
    User: Si => *QnA-AnswerCorrect*

Table 10: Yalo RL Results

Action	Count	Precision	Recall	F1
FS-Script-Search-Arrival-Airpots	1514	0.974	0.979	0.977
FS-Arrival	1127	0.875	0.991	0.929
humanLoop	1074	0.961	0.943	0.952
FS-Script-Search-Departure-Airports	1063	0.992	0.992	0.992
FS-PassengersAdults	1018	0.936	0.942	0.939
QnA-QuestionScript	700	0.861	0.849	0.855
FS-Rates	620	0.984	0.994	0.989
FS-PassengersChildren	581	0.998	0.995	0.997
Menu	353	0.792	0.776	0.784
default	343	0.753	0.633	0.688
Welcome	333	0.830	0.775	0.801
QnA-Question	291	0.954	0.993	0.973
QnA-AnswerCorrect	194	0.948	0.948	0.948
FS-SelectDate-Round-Trip-Error	165	0.865	0.897	0.881
Hi	112	0.458	0.438	0.447
menu-flight-status	110	1.000	1.000	1.000
FS-PassengersChildrenNumber	105	0.981	0.981	0.981
QnA-Answer0	97	0.969	0.979	0.974
YourWelcome	89	0.914	0.955	0.934
buenfin1a-yes	86	0.580	0.465	0.516
CheckIn-LastName	66	0.985	0.985	0.985
QnA-Answer1	62	0.984	0.968	0.976
FS-SelectDate-Error	58	0.491	0.448	0.468
CheckIn-PNR	49	0.918	0.918	0.918
QnA-Answer2	45	0.955	0.933	0.944
CheckIn-Start	40	0.927	0.950	0.938
ask-last-name	39	1.000	0.974	0.987
find-flight-by-pnr	30	0.931	0.900	0.915
QnA-Answer1-Help	29	0.839	0.897	0.867
ask-departure-date-flight	23	1.000	0.870	0.930
sfn_pnr	23	0.958	1.000	0.979
FS-PassengersGroups	20	1.000	0.850	0.919
Checkin-SearchPNR	19	0.425	0.895	0.576
sfn_lastname	19	0.947	0.947	0.947
save-destination-city	18	0.407	0.611	0.489
wait-for-destination-iata	18	0.167	0.111	0.133
CheckIn-international-2	17	1.000	1.000	1.000
create_subscription	16	1.000	1.000	1.000
CheckIn-LastNameAgain	13	0.692	0.692	0.692
FS-JustChildren	13	1.000	1.000	1.000
Identity	13	0.714	0.769	0.741
humanCheckIn	10	0.000	0.000	inf
no-have-number-flight	10	1.000	0.100	0.182
search-weekend-destinations	10	0.500	0.100	0.167
sfn_lambda	8	0.545	0.750	0.632
ask_lastname	7	1.000	1.000	1.000
init	6	inf	0.000	nan
Bullying	5	inf	0.000	nan
ByeByeMessageTransition	5	1.000	0.200	0.333
CheckIn-MoreInfo	5	inf	0.000	nan
search_reservation	5	0.500	0.800	0.615
CheckIn-PNRAgain	4	1.000	0.750	0.857
push_notification_default	4	inf	0.000	nan
search-beach-destinations	4	0.500	0.250	0.333
tracking-flight	4	0.200	0.250	0.222
world-trace-script	4	inf	0.000	nan
parse-departure-date-flight	3	0.000	0.000	inf

Figure 3: Reinforcement Examples

$s = ($ Menu, "<name> , puedo ayudarte con estas opciones , selecciona una o escribe tu consulta en un mensaje . En cualquier momento puedes' regresar a este menú, sólo escribe la palabra Aerobot ." "Como puedo hacer para comprar un boleto reservar un boleto" $)$ (10)

a	QnA-QuestionScript	FS-Arrival	Welcome	wait-for-destination-iata
$Model_{RL}(s,a)$	0.797	0.194	0.004	0.000
$Model_{baseline}(s,a)$	0.996	0.000	0.001	0.001

$s = ($ init, "<name> , puedo ayudarte con estas opciones , selecciona una o escribe tu consulta en un mensaje . En cualquier momento puedes' regresar a este menú, sólo escribe la palabra Aerobot ." "Mexicali a ciudad México" $)$ (11)

a	FS-Arrival	QnA-QuestionScript	Welcome	buenfin1a-yes
$Model_{RL}(s,a)$	0.998	0.001	0.000	0.000
$Model_{baseline}(s,a)$	0.000	0.956	0.027	0.005

$s = ($ QnA-Question, "Escribe tu pregunta en un mensaje a continuación . Por ejemplo, : ¿ Puedo viajar con mi perro ?" "¿ cuanto me sale un vuelo desde boca de río a Guadalajara ?" $)$ (12)

a	QnA-QuestionScript	FS-Arrival	Menu	default
$Model_{RL}(s,a)$	0.968	0.021	0.003	0.001
$Model_{baseline}(s,a)$	0.998	0.000	0.000	0.000