Optimizing Chatbot Fallback Intent Selections with Reinforcement Learning

Jeremy David Curuksu [1 2]

Abstract

Large language models used in GPT-4 and Alexa are limited by their ability to assess the validity of their own answers i.e., to fall back on a clarification intent when needed. Reinforcement learning can be used specifically to address this fallback selection problem, by adapting to semantic pitfalls of a given language model in a given environment. This is demonstrated in a simplified environment where the chatbot learns when best to ask for clarifications. After training it identifies correct intents in < 2 steps on average in over 99% of dialogues.

1. Introduction

Developing intelligent chatbots is a challenging AI problem which often requires deep learning large language models (LLM) to be trained on massive amounts of data. Even with recent developments combining LLMs with reinforcement learning based on human preferences (Christiano et al., 2017), chatbots such as GPT-4 struggle to calibrate their own uncertainty (Kadavath et al., 2022; Lin et al., 2022) i.e., the probability that their responses are valid and coherent. Eventually, we will need AI systems that are honest, meaning that accurately and faithfully evaluate their level of confidence in their own knowledge and reasoning (Kadavath et al., 2022). When building a chatbot using services such as the Alexa Skills Kit, LLMs are pre-trained on large amount of data to convert utterances to text and recognize the intent of the text (Kumar et al., 2017). The developer customizes an *interaction model* which is a set of intents each associated with utterances, prompts and slots (Fig.1, Appendix A), and a *dialog model* which identifies the steps of a multi-turn conversation between the chatbot and the user in order to collect information needed to fulfill each intent. Although the most recent chatbots such as GPT-4 are fined tuned using human feedback by reinforcement learning (Christiano et al., 2017), most chatbot development services still require the developer to define the different ways a user might interact with the chatbot (dialogue model), and include a native fallback intent invoked when the pretrained LLMs can't deduce the user's intent i.e., for situations when the user input is not recognized (AWS, 2023). Both types of chatbots (with or without fine tuning by human feedback) struggle to calibrate their own uncertainty (Kadavath et al., 2022). The developer can refine and curate heuristic rules i.e., the dialogue model, assuming an ideal semantic parsing from the pretrained LLM. But LLMs are far from perfect and known to sometimes hallucinate (Ouyang et al., 2022) or misinterpret utterances without awareness of their mistake, and without asking for clarifications either.

Reinforcement Learning (RL) has been successfully applied in recent years to autonomous decision making in video games (Vinyals et al., 2019), strategic board games (Silver et al., 2018), robotics (Ibarz et al., 2021), self-driving (Sallab et al., 2017), and financial trading (Wu et al., 2020). It has also been applied to design conversational chatbots rewarded based on the long-term success of dialogues as measured by linguistic properties called *validity* and *coherence* (Li et al., 2016), and to manage chatbots with pre-engineered language features fine tuned with human feedback (Christiano et al., 2017; Kumar et al., 2017; Serban et al., 2017). To my knowledge, it has not yet been applied specifically to learn an optimal fallback selection mechanism i.e., to address semantic parsing pitfalls of a given (pre-trained) LLM encoder in a given, user-defined interaction model.

Figure 1. Typical components of a chatbot interaction model.

In this paper, an RL-based language model is developed specifically to learn selection policies between the core intents and the fallback intent without the need for any predetermined heuristic rules. It autonomously identifies optimal selection policies to mitigate the ambiguity of utterances encoded by LLM with the use of the fallback intent given

[1]Center for Data Science, New York University, NY, USA [2]Amazon.com LLC, New York, NY, USA. Correspondence to: Jeremy Curuksu <curukj@amazon.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

a specific interaction environment (i.e., observable set of intents, utterances, prompts and slots), by exploring possible sequences of interactions in this environment. It bypasses the need for defining heuristic rules and does not assume a perfectly trained LLM either. The LLM is used to define the RL agent state and the RL agent adapts to it by exploring the interaction environment and learning fallback selection policies that best mitigate the LLM semantic pitfalls in this environment. The agent learns, from scratch, to select the proper intents or to fall back on asking for clarifications, as is demonstrated in simulations involving deterministic users (section 4.1) and adaptive users (section 4.2).

2. Reinforcement Learning for Chatbots

In a RL process, a goal is defined for an agent which makes decisions in an *environment*. This goal is translated into a mathematical formula called a reward function, which rewards or penalizes the agent when it takes an action, helping the agent reach the predefined goal. At each step t, the agent receives a representation of the environment state s_t, takes an action a_t which brings the agent in a new state s_{t+1}, and receives a numerical reward r_{t+1} for having taken a_t in s_t , as shown in Fig.2.

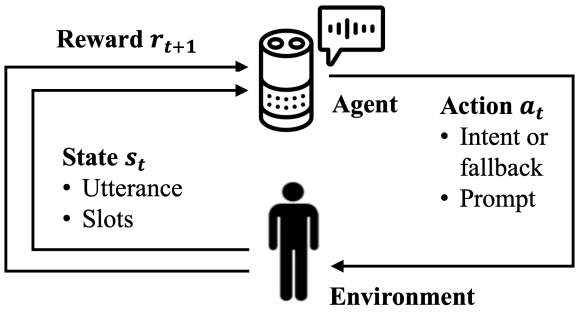


Figure 2. Components of the current RL decision process.

A sequence of actions is called a policy; the purpose of reinforcement learning is to identify an optimal policy to reach the predefined goal. More details can be found in (Sutton & Barto, 2018).

In the context of natural language processing (NLP), any NLP encoder can transform an utterance into a numerical vector (Kumar et al., 2017) to define a state s_t in the environment (details in section 3.3). Upon observing s_t, the agent takes two actions: it selects one of the core intents or the fallback intent (action 1), and it decides to prompt the user for a slot or to not ask for any slot (action 2).

In contrast to supervised and semi-supervised learning, with reinforcement learning it is possible to mitigate multiple learning goals simultaneously, by adding multiple terms in the reward function each addressing a particular sub-goal (Sutton & Barto, 2018). Since asking for clarifications is better than moving forward with a wrong intent, the proposed RL-based dialogue model is designed to learn the optimal tradeoff between speed and validity/coherence. These goals are encoded in the reward function detailed in section 3.5: it contains terms to incentivize the agent to close the dialogue in a small number of steps, to identify the correct intents or mitigate by asking for clarifications, and to fill in slots required for fulfillment. To demonstrate the impact of a RL dialogue model compared to heuristic models, a multi-agent RL simulation was also implemented where two RL agents interact with each other, one agent emulating a chatbot and one agent emulating a user (details in section 3.2). The two agents share a common objective (same reward function). Not surprisingly, a cooperation was observed between the two agents (details in section 4.2), which helped further improve speed and coherence by learning directly from the sampled dialogues, without any heuristic rule.

A custom interaction model was used for all results reported in this paper and is summarized in Appendix A. It contains typical components of an interaction model in the Alexa Skills Kit, such as a hierarchical modular organization. For example, the intents */pizza* and */dessert* are found within the intent */food*, and each intent is associated with a custom set of possible utterances and required slots needed for fulfillment. The interaction model was reduced to minimal complexity to demonstrate the impact of RL, yet preserves the typical ambiguity that makes speech recognition and NLP challenging in LLMs: some utterances that belong to different intents are almost identical, for example *Open the food module to find pizza* and *Open the food module to find dessert* contain 5 tokens and differ by only 1 token (80% identical). And some slots are 100% identical between different intents, such as *Are you looking for pick up or delivery* for all intents within */food*. This ambiguity was preserved to evaluate whether RL could learn fallback mitigation policies i.e., learn when best to ask for clarifications.

3. Methodology

3.1. Reinforcement Learning Goals

When a developer designs a chatbot, the goal is generally to make the interaction with the user simple and natural, mimicking human conversational patterns, and to fulfill the intents of the user coherently and efficiently (Kumar et al., 2017; Li et al., 2016). More specifically, an optimal dialogue model is (i) *valid*: the agent identifies the right intents among candidate intents, (ii) *coherent*: the agent identifies appropriate prompts during the dialogue i.e., it solicits data required for fulfillment (slots) and does not solicit data not required for fulfillment, and (iii) *efficient*: the agent minimizes the time it takes to fulfill intents. These goals are translated into a RL reward function in section 3.5.

3.2. Simulated Environment

In a first test case (results in section 4.1), the simulated users select intents randomly (uniform sampling) and also randomly select utterances within the subset of five utterances defined for each intent in the interaction model. Each RL episode corresponds to a dialogue between the user and the agent given the user has chosen one intent. If the agent guesses the intent correctly, the dialogue proceeds deterministically to filling remaining slots (if any) and to fulfillment. If the agent guesses the intent incorrectly, the user selects another of the five utterances corresponding to the same intent. As long as the total number of steps (including greetings/farewells and slot filling) is less than 10, the dialogue continues until fulfillment is reached. If the total number of steps reaches 10, the user ends the dialogue and complains about the chatbot's lack of efficiency and poor experience.

In the second test case (results in section 4.2), a multi-agent framework was implemented where two agents interact with each other, one agent emulating the chatbot exactly as described above, and one agent emulating the user. Each agent becomes the *environment* from the perspective of the other agent. The only difference in this test case is when the chatbot asks for clarifications or guesses the intent incorrectly, the user does not select one of the five utterances randomly but instead learns to select utterances which lead to higher reward. The behavior of each agent is learned directly by reinforcement from the sampled dialogues. Both agents aim to maximize the same reward function. Thus, a cooperation between the two agents is expected (Silver et al., 2018), which may result in improved speed and coherence. The multi-agent test case was created because in practice, a human user who regularly interacts with a chatbot tends to use words that the chatbot understands better. That is, a human user does not maintain a purely uniform and random choice of words as emulated in test case 1.

In both test cases, a chatbot dialogue model is obtained directly by learning from sampled dialogues. After training, it can be queried to select intents and prompts without having to define heuristic rules. In this paper a simulated environment was used, but after training the RL agent could also continually explore and learn from real-world human feedback (Christiano et al., 2017; Ouyang et al., 2022).

3.3. State Space

Any NLP encoder can be used to transform each utterance sent by the user into a numerical vector, which in turn can define a state s_t in the RL environment. A Word2Vec unsupervised learning model (as implemented in (Gupta & Khare, 2017)) was pre-trained on the entire corpus of the interaction model to represent each word as a numerical embedding vector. During a dialogue, the embedding vectors computed in real time for every word in an utterance were averaged to define a sentence-level embedding vector. Each word embedding had 100 dimensions so each utterance was also encoded by a vector of 100 dimensions after averaging over all words in the utterance. A separate binary vector stored which of the 6 slots had been filled in each dialogue and was appended to the NLP embedding vector, resulting in a RL state of 106 dimensions. This NLP encoding represents a simplified version of LLMs and will thus have obvious semantic pitfalls, which the chatbot will need to learn from and adapt to.

3.4. Action Space

The action space has two dimensions: at each step, the chatbot selects a core intent or the fallback intent, and prompts the user to fill a slot or does not prompt the user for any slot. The custom interaction model was defined by 6 core intents and 6 possible slots across all intents (Appendix A) so the action is defined as a vector of two integers (a, b), where a is the selected intent and b is the selected slot. a and b can take values between 0 and 6, where 0 corresponds to *clarification* and *no slot*, respectively. As shown in Appendix A, each core intent requires a specific combination of slots. The navigation intent requires no slot at all.

3.5. Reward Function

The reward function aims at rewarding or penalizing the agent when it takes an action to help the agent mitigate multiple learning goals and identify an optimal dialogue model. Each sub-goal is encoded by a specific component in the reward function:

$$r_t = \lambda_1 r_t^1 + \lambda_2 r_t^2 + \lambda_3 r_t^3$$

where λ_1, λ_2, λ_3 are weights (hyperparameters) which can be fine-tuned to boost the impact of each term relative to one another. As introduced in section 3.1, an optimal dialogue model needs to be valid (r_t^1), coherent (r_t^2) and efficient (r_t^3). These three components are detailed below.

3.5.1. Validity of selected intents: r_t^1

The component r_t^1 rewards the agent depending on whether it identifies the correct intent. The agent can select either of the 6 core intents, or fall back on the clarification intent. If the agent guesses the user intent correctly, it receives a positive value ($+5$). If the agent guesses the intent incorrectly, it receives a negative value (-5). If the agent falls back on the clarification intent, it is neither rewarded nor penalized by this component i.e., $r_t^1 = 0$. The validity of the selected intents is further reinforced at the end of each dialogue by a large positive value ($+10$) if the intent has been fulfilled and the number of steps is less than 10, or by a large negative value (-10) otherwise.

3.5.2. COHERENCE OF SELECTED SLOTS: r_t^2

The component r_t^2 rewards the agent depending on whether it identifies the correct slots. The agent can prompt the user for either of the 6 slots, or do not prompt the user for any slot. If the agent identifies a valid slot during the dialogue i.e., if it asks for slot-data required for fulfillment, it receives a positive value ($+5$). In contrast, if the agent prompts the user for slot-data not required for fulfillment, it receives a negative value (-5). If the agent does not prompt the user for any slot, it is neither rewarded nor penalized by this component i.e., $r_t^2 = 0$.

3.5.3. EFFICIENCY OF DIALOGUE: r_t^3

The component r_t^3 systematically penalizes the agent by a negative value -1 at every step of the dialogue, so the agent is incentivized to close the dialogue quickly.

By simultaneously mitigating all three goals of validity, coherence and efficiency, the agent tries to minimize the time it takes to identify the right intent and solicit slots required to fulfill the intent. Given some utterances may be ambiguous, the agent may fall back on the clarification intent, in particular once it has learned that an utterance is frequently mixed up between two different intents. For example, in the current interaction model the intents */food/dessert* and */food/pizza* may be easier to differentiate with utterances of the forms *Can you help me find some cake for dessert? vs. I am looking for Italian pizza*, than with the forms *Open the food module to find dessert vs. Open the food module to find pizza*. In parallel, given some slots are common to multiple intents, the agent may explore original strategies such as prompting the user for a particular slot (e.g., *Are you looking for pick up or delivery?*) most likely to be needed based on a given user utterance (*Open the food module to find ...*), even though the utterance may be ambiguous and require clarifications to determine the actual intent.

4. Results

The chatbot was trained by simulating interactions $(s_t, a_t, r_{t+1}, s_{t+1})$ with users, where each episode corresponds to a dialogue between the user and the chatbot given the user has chosen an intent. All RL simulations were carried out using the proximal policy optimization (PPO) algorithm for a total number of user-chatbot interactions varying from 130,000 to 150,000 and representing a total of 30,000 dialogues. Each simulation was repeated 3 times and run on 36 CPUs using C5 9XL Amazon EC2 instances, taking approximately 5h each. An efficient exploration of state-action space was ensured by applying an ϵ-greedy exploration schedule: a search over different ϵ-schedules showed that linearly switching ϵ from 10% to 1% over the first 60,000 interactions led to the best results.

4.1. Analysis of User-Chatbot Interactions in Dialogues Simulated with RL

To evaluate the learning performance of the chatbot with an adaptive RL dialogue model, the total accumulated reward was computed for each dialogue. The accumulated reward in a given dialogue measures how good the policy followed in this dialogue was. Thus, when compared between dialogues (Fig.3) it measures the relative value of the policies learned. Three independent trials of 30,000 dialogues were produced to assess sensibility to the random ϵ-greedy exploration schedule. Fig.3 suggests the RL chatbot dynamically explores the custom interaction model and identifies a dialogue model that mitigates the multiple learning goals of validity (identify the right intent), coherence (solicit relevant slots), and efficiency. In all trials, the RL chatbot transitions from a phase of random exploration in the first 10,000 episodes, where the accumulated reward ranges from -120 to $+20$, to a phase (episodes 10,000 to 25,000) which trades off exploitation of learned policies with partial exploration: the accumulated reward is spread across a smaller range of values and skewed toward higher values ranging from -60 to $+20$. A third phase is then observed across the final 5,000 dialogues where the accumulated reward ranges almost exclusively from -40 and $+20$. This indicates that the RL chatbot now systematically avoids certain behaviors when interacting with the user.

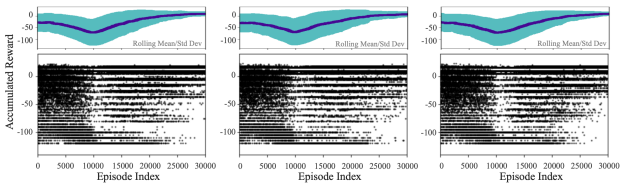


Figure 3. Accumulated rewards across 3×30,000 dialogues sampled with single-agent RL.

Table 1 shows the average proportion of *successful* dialogues (i.e., intent fulfilled within 10 steps) and the average number of steps in successful dialogues, per intent. Standard errors in parentheses were computed over the three independent trials shown in Fig.3. In the first 5,000 dialogues, 68% are successful and these take > 4 interactions to complete on average. In contrast, in the final 5,000 dialogues (phase 3 observed in Fig.3), 99% are successful and these take at most 2 steps on average to infer the right intent. Fig.3 confirms that sub-optimal policies are still occasionally followed. This indicates that the RL chatbot has not yet *fully* converged by the end of these simulations.

Appendix B shows a few examples sampled at the beginning and at the end of the simulations. In some dialogues observed in phase 3 of Fig.3, the chatbot was able to take

Table 1. Validity and efficiency of sampled dialogues.

	SINGLE RL		MULTI RL	
	0-5K	25-30K	0-5K	25-30K
% SUCCESS	68 (.8)	99 (.1)	67 (1.4)	100 (0)
NUMBER OF STEPS IN SUCCESSFUL DIALOGUES:				
/NAVIGATION	4.1 (.2)	1.0 (.0)	4.1 (.1)	1.0 (.0)
/PIANO	4.2 (.1)	1.8 (.2)	4.1 (.1)	1.3 (.1)
/DENTIST	4.2 (.0)	1.6 (.2)	4.1 (.2)	1.3 (.0)
/PIZZA	4.1 (.1)	1.7 (.3)	4.3 (.0)	1.3 (.0)
/ADVIL	4.3 (.2)	1.7 (.3)	4.2 (.1)	1.3 (.0)
/DESSERT	4.4 (.1)	2.3 (.5)	4.3 (.1)	1.4 (.1)

the initiative to prompt the user for a valid slot even when it felt back on asking the user for clarification of intent. The chatbot has thus learned some original policies which correctly infer slots required for fulfillment *even when the exact intent cannot yet be precisely determined*. This strategy spontaneously emerged by reinforcement learning and allows the chatbot to be more efficient i.e., to complete a dialogue with a smaller number of steps without sacrificing validity and coherence.

4.2. Nash Equilibrium in Dialogues Simulated with Multi-Agent RL

In this second test case, the chatbot and the user are both RL agents. Multi-agent RL is an active field of research (Silver et al., 2018; Vinyals et al., 2019). A key challenge is that from the perspective of one agent, other agents are part of the environment, making the environment non-stationary. Metastable states can be tracked under particular conditions known as the *Nash equilibrium* i.e., when competing agents coexist in a shared environment and each agent lacks incentive to further change its policy. For example, AlphaZero (Silver et al., 2018) was designed to reach a Nash equilibrium in the games of Go and Chess by learning policies optimal *when used by both players* alternatively. After millions of self-played games, the policies learned by AlphaZero on Go and Chess are optimal for both players. This is possible because in a two-player game such as Go, each agent tries to maximize a reward function which is perfectly symmetric to the other agent's reward function.

In this paper, the Nash equilibrium is even easier to reach because both agents try to maximize the *same* reward function. That is, they both aim to make the dialogue more valid, coherent and efficient. The only difference compared to the single-agent test case is that the user is also an RL agent whose action is to select an utterance within the subset of five utterances defined for a given intent. As can be seen in Fig.4, the adaptive RL dialogue model converges toward optimal policies (reward in range -20 to $+20$) faster than

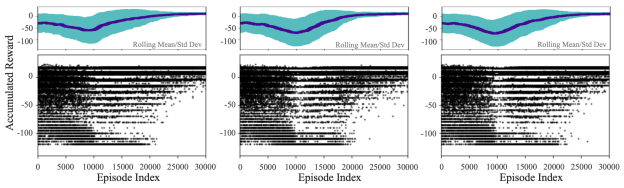


Figure 4. Accumulated rewards across 3×30,000 dialogues sampled with multi-agent RL.

in the single-agent test case (Fig.3) where the user selects utterances randomly. A second result observed in multi-agent simulations is that 100% of the policies followed by the end of the simulations have a reward ranging between -20 and $+20$. The standard deviation observed in the final 5,000 dialogues is significantly smaller in Fig.4 compared to Fig.3. Table 1 also reports that 100% of dialogues complete successfully. Thus, the cooperation of the user has eliminated the sub-optimal policies that were still occasionally observed in the end of the single-agent simulations.

Table 1 also indicates a systematic improvement in *efficiency*, for every intent. It takes 1.3 steps on average to infer the right intent, compared to 1.8 steps on average in single-agent simulations. This makes sense because when the user sends utterances better understood by the chatbot, the chatbot less often needs to ask for clarifications. Appendix B shows some examples of dialogues sampled at the end of multi-agent simulations.

These results indicate that the RL agent has identified reproducible policies to interact with the user and infer the right intent (validity), solicit relevant slots (coherence), and minimize the time it takes to fulfill intents (efficiency). The resulting RL dialogue model can now be queried to select intents and prompts.

5. Conclusion

Reinforcement learning was applied to optimize chatbot fallback intent selections in user interactions sampled from a custom interaction model. The chatbot converged to policies which fulfill intents in 99% of dialogues and identified the correct intent in 1.8 steps on average. When the user cooperated with the chatbot, the correct intent was identified in 1.3 steps on average in 100% of dialogues sampled.

In addition to select between intents, the RL chatbot also learned to fill in slots as fast as possible. In particular, it identified an original strategy to increase the speed of fulfillment without sacrificing coherence, by filling in some valid slots even when the utterance is still too ambiguous to determine the exact intent.

Any interaction model could be used to fine tune a dialogue model that optimizes fallback intent selections by reinforcement learning. Once trained, the RL agent can be queried to select intents and prompts without defining heuristic rules. An adaptive RL dialogue model can also continue to learn from human feedback, in contrast to heuristics.

Future research will focus on identifying RL dialogue models in more complex interaction models, where optimizing fallback policies for the chatbot to calibrate its own uncertainty may reveal more challenging convergence issues. When training the RL agent each episode was a dialogue between the user and the chatbot given the user had chosen an intent. All intents were known and well-defined. But human preferences have many facets including mixed and unclear intents, and intents evolving in time. Teaching an RL agent to fall back on clarifications based on clearly defined intents may not generalize well to environments where the user intent can be mixed and unclear. Measuring performance in such environment will also be challenging.

A cooperation was observed between the users and the chatbot, which in turn led to better dialogues. In practice, a human user who regularly interacts with a chatbot does not maintain a random choice of words as emulated in the single-agent test case. Instead, a human tends to use words that the chatbot understands better, as emulated in the multi-agent test case. This is because a human never sample decisions from a purely uniform random distribution. In other words a human is not a robot: a human speaker has been conditioned, through evolution and a lifetime of reinforcement, to adapt utterances when it is self-beneficial to do so.

These results demonstrate that RL can be combined with LLMs specifically to learn when best to fall back on the clarification intent, and can replace heuristics to develop adaptive dialogue models.

References

AWS. Amazon lex v2 developer guide. Technical report, Amazon Web Services, 2023.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Gupta, S. and Khare, V. Blazingtext: Scaling and accelerating word2vec using multiple gpus. In *Proceedings of the Machine Learning on HPC Environments*, pp. 1–5, 2017.

Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., and Levine, S. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Dodds, Z. H., DasSarma, N., and Tran-Johnson, E. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Kumar, A., Gupta, A., Chan, J., Tucker, S., Hoffmeister, B., Dreyer, M., Peshterliev, S., Gandhe, A., Filiminov, D., and Rastrow, A. Just ask: building an architecture for extensible self-service spoken language understanding. *arXiv preprint arXiv:1711.00549*, 2017.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

Lin, S., Hilton, J., and Evans, O. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*, 2022.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., and Ray, A. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*, 2017.

Serban, I. V., Sankar, C., Germain, M., Zhang, S., Lin, Z., Subramanian, S., Kim, T., Pieper, M., Chandar, S., and Ke, N. R. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*, 2017.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., and Lillicrap, T. A general reinforcement learning algorithm that masters chess, shogi and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

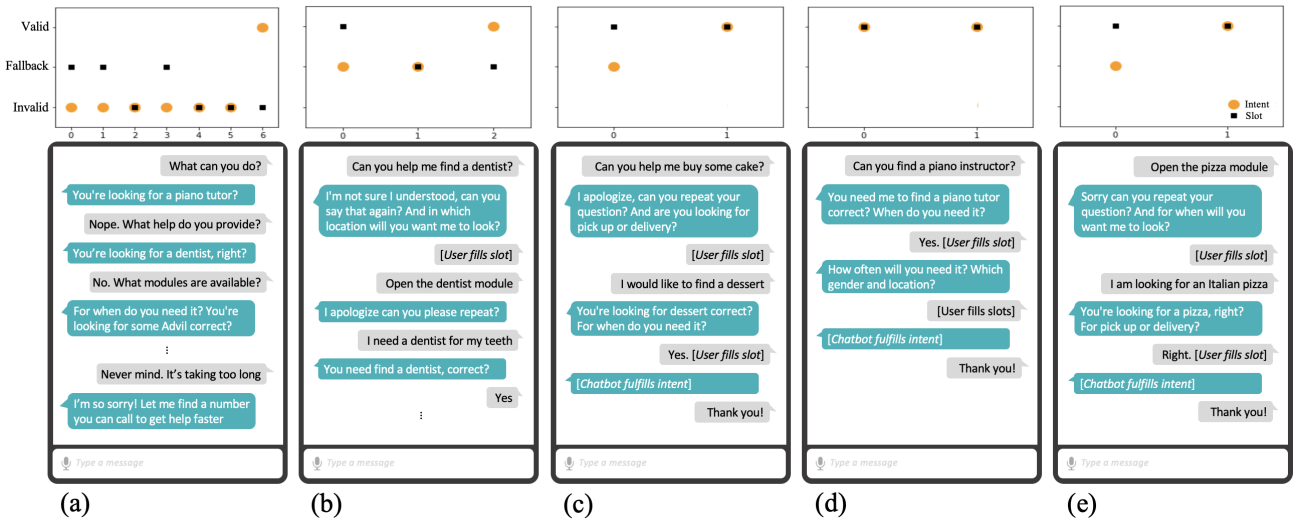
Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., and Georgiev, P. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019.

Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., and Fujita, H. Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, 538:142–158, 2020.

INTENT	CORRESPONDING SLOTS
/NAVIGATION	NO SLOT
/TUTOR/PIANO	LOCATION, AGE, GENDER, WHEN, HOW OFTEN
/DOCTOR/DENTIST	LOCATION, WHEN
/PHARMACY/ADVIL	LOCATION, WHEN, HOW OFTEN, PICKUP OR DELIVERY
/FOOD/PIZZA	WHEN, PICKUP OR DELIVERY
/FOOD/DESSERT	WHEN, PICKUP OR DELIVERY
/CLARIFICATION	NO SLOT

A. Appendix A.

Map of intents and slots in the custom interaction model used in this paper. The complete JSON file mapping all utterances and prompts to all intents and slots (resulting in approximately 1230 possible combinations of utterances and prompts for the RL agent to explore i.e., 7 intents $\times$ 5 utterances $\times$ 7 slots $\times$ 5 prompts) is available upon request to the author.



B. Appendix B.

Sample of user-chatbot interactions before training (a), after an optimal dialogue model has been identified by single agent RL (b, c), and after an optimal dialogue model has been identified by multi-agent RL (d, e). The validity of every action taken at every step by the chatbot is shown above each sampled dialogue.