DIALOGUE LEARNING WITH HUMAN-IN-THE-LOOP

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, Jason Weston
Facebook AI Research,
New York, USA
{jiwel,ahm,spchopra,ranzato,jase}@fb.com

ABSTRACT

An important aspect of developing conversational agents is to give a bot the ability to improve through communicating with humans and to learn from the mistakes that it makes. Most research has focused on learning from fixed training sets of labeled data rather than interacting with a dialogue partner in an online fashion. In this paper we explore this direction in a reinforcement learning setting where the bot improves its question-answering ability from feedback a teacher gives following its generated responses. We build a simulator that tests various aspects of such learning in a synthetic environment, and introduce models that work in this regime. Finally, real experiments with Mechanical Turk validate the approach.

1 INTRODUCTION

A good conversational agent (which we sometimes refer to as a learner or bot[1]) should have the ability to learn from the online feedback from a teacher: adapting its model when making mistakes and reinforcing the model when the teacher's feedback is positive. This is particularly important in the situation where the bot is initially trained in a supervised way on a fixed synthetic, domain-specific or pre-built dataset before release, but will be exposed to a different environment after release (e.g., more diverse natural language utterance usage when talking with real humans, different distributions, special cases, etc.). Most recent research has focused on training a bot from fixed training sets of labeled data but seldom on how the bot can improve through online interaction with humans. Human (rather than machine) language learning happens during communication (Bassiri, 2011; Werts et al., 1995), and not from labeled datasets, hence making this an important subject to study.

In this work, we explore this direction by training a bot through interaction with teachers in an online fashion. The task is formalized under the general framework of reinforcement learning via the teacher's (dialogue partner's) feedback to the dialogue actions from the bot. The dialogue takes place in the context of question-answering tasks and the bot has to, given either a short story or a set of facts, answer a set of questions from the teacher. We consider two types of feedback: explicit numerical rewards as in conventional reinforcement learning, and textual feedback which is more natural in human dialogue, following (Weston, 2016). We consider two online training scenarios: (i) where the task is built with a dialogue simulator allowing for easy analysis and repeatability of experiments; and (ii) where the teachers are real humans using Amazon Mechanical Turk.

We explore important issues involved in online learning such as how a bot can be most efficiently trained using a minimal amount of teacher's feedback, how a bot can harness different types of feedback signal, how to avoid pitfalls such as instability during online learing with different types of feedback via data balancing and exploration, and how to make learning with real humans feasible via data batching. Our findings indicate that it is feasible to build a pipeline that starts from a model trained with fixed data and then learns from interactions with humans to improve itself.

[1]In this paper, we refer to a learner (either a human or a bot/dialogue agent which is a machine learning algorithm) as the student, and their more knowledgeable dialogue partner as the teacher.

2 Related Work

Reinforcement learning has been widely applied to dialogue, especially in slot filling to solve domain-specific tasks (Walker, 2000; Schatzmann et al., 2006; Singh et al., 2000; 2002). Efforts include Markov Decision Processes (MDPs) (Levin et al., 1997; 2000; Walker et al., 2003; Pieraccini et al., 2009), POMDP models (Young et al., 2010; 2013; Gašic et al., 2013; 2014) and policy learning (Su et al., 2016). Such a line of research focuses mainly on frames with slots to fill, where the bot will use reinforcement learning to model a state transition pattern, generating dialogue utterances to prompt the appropriate user responses to put in the desired slots. This goal is different from ours, where we study end-to-end learning systems and also consider non-reward based setups via textual feedback.

Our work is related to the line of research that focuses on supervised learning for question answering (QA) from dialogues (Dodge et al., 2015; Weston, 2016), either given a database of knowledge (Bordes et al., 2015; Miller et al., 2016) or short texts (Weston et al., 2015; Hermann et al., 2015; Rajpurkar et al., 2016). In our work, the discourse includes the statements made in the past, the question and answer, and crucially the response from the teacher. The latter is what makes the setting different from the standard QA setting, i.e. we use methods that leverage this response also, not just answering questions. Further, QA works only consider fixed datasets with gold annotations, i.e. they do not consider a reinforcement learning setting.

Our work is closely related to a recent work from Weston (2016) that learns through conducting conversations where supervision is given naturally in the response during the conversation. That work introduced the use of forward prediction that learns by predicting the teacher's feedback, in addition to using reward-based learning of correct answers. However, two important issues were not addressed: (i) it did not use a reinforcement learning setting, but instead used pre-built datasets with fixed policies given in advance; and (ii) experiments used only simulated and no real language data. Hence, models that can learn policies from real online communication were not investigated. To make the differences with our work clear, we will now detail these points further.

The experiments in (Weston, 2016) involve constructing pre-built fixed datasets, rather than training the learner within a simulator, as in our work. Pre-built datasets can only be made by fixing a prior in advance. They achieve this by choosing an omniscient (but deliberately imperfect) labeler that gets π_{acc} examples always correct (the paper looked at values 50%, 10% and 1%). Again, this was not learned, and was fixed to generate the datasets. Note that the paper refers to these answers as coming from "the learner" (which should be the model), but since the policy is fixed it actually does not depend on the model. In a realistic setting one does not have access to an omniscient labeler, one has to learn a policy completely from scratch, online, starting with a random policy, so their setting was not practically viable. In our work, when policy training is viewed as batch learning over iterations of the dataset, updating the policy on each iteration, (Weston, 2016) can be viewed as training only one iteration, whereas we perform multiple iterations. This is explained further in Sections 4.2 and 5.1. We show in our experiments that performance improves over the iterations, i.e. it is better than the first iteration. We show that such online learning works for both reward-based numerical feedback and for forward prediction methods using textual feedback (under certain conditions which are detailed). This is a key contribution of our work.

Finally, (Weston, 2016) only conducted experiments on synthetic or templated language, and not real language, especially the feedback from the teacher was scripted. While we believe that synthetic datasets are very important for developing understanding (hence we develop a simulator and conduct experiments also with synthetic data), for a new method to gain traction it must be shown to work on real data. We hence employ Mechanical Turk to collect real language data for the questions and importantly for the teacher feedback and construct experiments in this real setting.

3 Dataset and Tasks

We begin by describing the data setup we use. In our first set of experiments we build a simulator as a testbed for learning algorithms. In our second set of experiments we use Mechanical Turk to provide real human teachers giving feedback.

3.1 Simulator

The simulator adapts two existing fixed datasets to our online setting. Following Weston (2016), we use (i) the single supporting fact problem from the bAbI datasets (Weston et al., 2015) which consists of 1000 short stories from a simulated world interspersed with questions; and (ii) the WikiMovies dataset (Weston et al., 2015) which consists of roughly 100k (templated) questions over 75k entities based on questions with answers in the open movie database (OMDb). Each dialogue takes place between a teacher, scripted by the simulation, and a bot. The communication protocol is as follows: (1) the teacher first asks a question from the fixed set of questions existing in the dataset, (2) the bot answers the question, and finally (3) the teacher gives feedback on the bot's answer.

We follow the paradigm defined in (Weston, 2016) where the teacher's feedback takes the form of either textual feedback, a numerical reward, or both, depending on the task. For each dataset, there are ten tasks, which are further described in Sec. A and illustrated in Figure 5 of the appendix. We also refer the readers to (Weston, 2016) for more detailed descriptions and the motivation behind these tasks. In the main text of this paper we only consider Task 6 ("partial feedback"): the teacher replies with positive textual feedback (6 possible templates) when the bot answers correctly, and positive reward is given only 50% of the time. When the bot is wrong, the teacher gives textual feedback containing the answer. Descriptions and experiments on the other tasks are detailed in the appendix. Example dialogues are given in Figure 1.

The difference between our simulation and the original fixed tasks of Weston (2016) is that models are trained on-the-fly. After receiving feedback and/or rewards, we update the model (policy) and then deploy it to collect teacher's feedback in the next episode or batch. This means the model's policy affects the data which is used to train it, which was not the case in the previous work.

Figure 1: **Simulator sample dialogues for the bAbI task (left) and WikiMovies (right).** We consider 10 different tasks following Weston (2016) but here describe only Task 6; other tasks are detailed in the appendix. The teacher's dialogue is in black and the bot is in red. (+) indicates receiving positive reward, given only 50% of the time even when correct.

bAbI Task 6: Partial Rewards	
Mary went to the hallway.	
John moved to the bathroom.	
Mary travelled to the kitchen.	
Where is Mary?	kitchen
Yes, that's right!	
Where is John?	bathroom
Yes, that's correct! (+)	

WikiMovies Task 6: Partial Rewards	
What films are about Hawaii?	50 First Dates
Correct!	
Who acted in Licence to Kill?	Billy Madison
No, the answer is Timothy Dalton.	
What genre is Saratoga Trunk in?	Drama
Yes! (+)	
...	

Figure 2: **Human Dialogue from Mechanical Turk (based on WikiMovies)** The human teacher's dialogue is in black and the bot is in red. We show examples where the bot answers correctly (left) and incorrectly (right). Real humans provide more variability of language in both questions and textual feedback than in the simulator setup (cf. Figure 1).

Sample dialogues with correct answers from the bot:	
Who wrote the Linguini Incident ?	richard shepard
Richard Shepard is one of the right answers here.	
What year did The World Before Her premiere?	2012
Yep! That's when it came out.	
Which are the movie genres of Mystery of the 13th Guest?	crime
Right, it can also be categorized as a mystery.	

Sample dialogues with incorrect answers from the bot:	
What are some movies about a supermarket ?	supermarket
There were many options and this one was not among them.	
Which are the genres of the film Juwanna Mann ?	kevin pollak
That is incorrect. Remember the question asked for a genre not name.	
Who wrote the story of movie Coraline ?	fantasy
That's a movie genre and not the name of the writer. A better answer would of been Henry Selick or Neil Gaiman.	

3.2 Mechanical Turk Experiments

Finally, we extended WikiMovies using Mechanical Turk so that real human teachers are giving feedback rather than using a simulation. As both the questions and feedback are templated in the simulation, they are now both replaced with natural human utterances. Rather than having a set of simulated tasks, we have only one task, and we gave instructions to the teachers that they could give feedback as they see fit. The exact instructions given to the Turkers is given in Appendix B. In general, each independent response contains feedback like (i) positive or negative sentences; or (ii) a phrase containing the answer or (iii) a hint, which are similar to setups defined in the simulator. However, some human responses cannot be so easily categorized, and the lexical variability is much larger in human responses. Some examples of the collected data are given in Figure 2.

4 Methods

4.1 Model Architecture

In our experiments, we used variants of the End-to-End Memory Network (MemN2N) model (Sukhbaatar et al., 2015) as our underlying architecture for learning from dialogue.

The input to MemN2N is the last utterance of the dialogue history x as well as a set of memories (context) C=c_1, c_2, ..., c_N. The memory C encodes both short-term memory, e.g., dialogue histories between the bot and the teacher, and long-term memories, e.g., the knowledge base facts that the bot has access to. Given the input x and C, the goal is to produce an output/label a.

In the first step, the query x is transformed to a vector representation u_0 by summing up its constituent word embeddings: $u_0 = Ax$. The input x is a bag-of-words vector and A is the $d \times V$ word embedding matrix where d denotes the emebbding dimension and V denotes the vocabulary size. Each memory c_i is similarly transformed to a vector m_i. The model will read information from the memory by comparing input representation u_0 with memory vectors m_i using softmax weights:

$$o_1 = \sum_i p_i^1 m_i \qquad p_i^1 = \texttt{softmax}(u_0^T m_i) \tag{1}$$

This process selects memories relevant to the last utterance x, i.e., the memories with large values of p_i^1. The returned memory vector o_1 is the weighted sum of memory vectors. This process can be repeated to query the memory N times (so called "hops") by adding o_n to the original input, $u_1 = o_1 + u_0$, or to the previous state, $u_n = o_n + u_{n-1}$, and then using u_n to query the memories again.

In the end, u_N is input to a softmax function for the final prediction:

$$a = \texttt{softmax}(u_N^T y_1, u_N^T y_2, ..., u_N^T y_L) \tag{2}$$

where $y_1, \ldots, y_L$ denote the set of candidate answers. If the answer is a word, y_i is the corresponding word embedding. If the answer is a sentence, y_i is the embedding for the sentence achieved in the same way that we obtain embeddings for query x and memory C.

The standard way MemN2N is trained is via a cross entropy criterion on known input-output pairs, which we refer to as supervised or imitation learning. As our work is in a reinforcement learning setup where our model must make predictions to learn, this procedure will not work, so we instead consider reinforcement learning algorithms which we describe next.

4.2 Reinforcement Learning

In this section, we present the algorithms we used to train MemN2N in an online fashion. Our learning setup can be cast as a particular form of Reinforcement Learning. The policy is implemented by the MemN2N model. The state is the dialogue history. The action space corresponds to the set of answers the MemN2N has to choose from to answer the teacher's question. In our setting, the policy chooses only one action for each episode. The reward is either 1 (a reward from the teacher when the bot answers correctly) or 0 otherwise. Note that in our experiments, a reward equal to 0 might mean that the answer is incorrect or that the positive reward is simply missing. The overall setup is closest to standard contextual bandits, except that the reward is binary.

When working with real human dialogues, e.g. collecting data via Mechanical Turk, it is easier to set up a task whereby a bot is deployed to respond to a large batch of utterances, as opposed to a single one. The latter would be more difficult to manage and scale up since it would require some form of synchronization between the model replicas interacting with each human.

This is comparable to the real world situation where a teacher can either ask a student a single question and give feedback right away, or set up a test that contains many questions and grade all of them at once. Only after the learner completes all questions, it can hear feedback from the teacher.

We use *batch size* to refer to how many dialogue episodes the current model is used to collect feedback before updating its parameters. In the Reinforcement Learning literature, batch size is related to *off-policy* learning since the MemN2N policy is trained using episodes collected with a stale version of the model. Our experiments show that our model and base algorithms are very robust to the choice of batch size, alleviating the need for correction terms in the learning algorithm (Bottou et al., 2013).

We consider two strategies: (i) online batch size, whereby the target policy is updated after doing a single pass over each batch (a batch size of 1 reverts to the usual on-policy online learning); and (ii) dataset-sized batch, whereby training is continued to convergence on the batch which is the size of the dataset, and then the target policy is updated with the new model, and a new batch is drawn and the procedure iterates. These strategies can be applied to all the methods we use, described below.

Next, we discuss the learning algorithms we considered in this work.

4.2.1 Reward-Based Imitation (RBI)

The simplest algorithm we first consider is the one employed in Weston (2016). RBI relies on positive rewards provided by the teacher. It is trained to imitate the correct behavior of the learner, i.e., learning to predict the correct answers (with reward 1) at training time and disregarding the other ones. This is implemented by using a MemN2N that maps a dialogue input to a prediction, i.e. using the cross entropy criterion on the positively rewarded subset of the data.

In order to make this work in the online setting which requires exploration to find the correct answer, we employ an ϵ-greedy strategy: the learner makes a prediction using its own model (the answer assigned the highest probability) with probability $1 - \epsilon$, otherwise it picks a random answer with probability ϵ. The teacher will then give a reward of $+1$ if the answer is correct, otherwise 0. The bot will then learn to imitate the correct answers: predicting the correct answers while ignoring the incorrect ones.

4.2.2 REINFORCE

The second algorithm we use is the REINFORCE algorithm (Williams, 1992), which maximizes the expected cumulative reward of the episode, in our case the expected reward provided by the teacher. The expectation is approximated by sampling an answer from the model distribution. Let a denote the answer that the learner gives, $p(a)$ denote the probability that current model assigns to a, r denote the teacher's reward, and $J(\theta)$ denote the expectation of the reward. We have:

$$\nabla J(\theta) \approx \nabla \log p(a)[r - b] \tag{3}$$

where b is the baseline value, which is estimated using a linear regression model that takes as input the output of the memory network after the last hop, and outputs a scalar b denoting the estimation of the future reward. The baseline model is trained by minimizing the mean squared loss between the estimated reward b and actual reward r, $||r - b||^2$. We refer the readers to (Ranzato et al., 2015; Zaremba & Sutskever, 2015) for more details. The baseline estimator model is independent from the policy model, and its error is not backpropagated through the policy model.

The major difference between RBI and REINFORCE is that (i) the learner only tries to imitate correct behavior in RBI while in REINFORCE it also leverages the incorrect behavior, and (ii) the learner explores using an ϵ-greedy strategy in RBI while in REINFORCE it uses the distribution over actions produced by the model itself.

4.2.3 FORWARD PREDICTION (FP)

FP (Weston, 2016) handles the situation where a numerical reward for a bot's answer is not available, meaning that there are no +1 or 0 labels available after a student's utterance. Instead, the model assumes the teacher gives textual feedback t to the bot's answer, taking the form of a dialogue utterance, and the model tries to predict this instead. Suppose that x denotes the teacher's question and C=c_1, c_2, ..., c_N denotes the dialogue history as before. In *FP*, the model first maps the teacher's initial question x and dialogue history C to a vector representation u using a memory network with multiple hops. Then the model will perform another hop of attention over all possible student's answers in $\mathbb{A}$, with an additional part that incorporates the information of which candidate (i.e., a) was actually selected in the dialogue:

$$p_{\hat{a}} = \texttt{softmax}(u^T y_{\hat{a}}) \quad o = \sum_{\hat{a} \in \mathbb{A}} p_{\hat{a}}(y_{\hat{a}} + \beta \cdot \mathbf{1}[\hat{a} = a]) \tag{4}$$

where $y_{\hat{a}}$ denotes the vector representation for the student's answer candidate $\hat{a}$. β is a (learned) d-dimensional vector to signify the actual action a that the student chooses. o is then combined with u to predict the teacher's feedback t using a softmax:

$$u_1 = o + u \quad t = \texttt{softmax}(u_1^T x_{r_1}, u_1^T x_{r_2}, ..., u_1^T x_{r_N}) \tag{5}$$

where x_{r_i} denotes the embedding for the i^{th} response. In the online setting, the teacher will give textual feedback, and the learner needs to update its model using the feedback. It was shown in Weston (2016) that in an off-line setting this procedure can work either on its own, or in conjunction with a method that uses numerical rewards as well for improved performance. In the online setting, we consider two simple extensions:

- ϵ-greedy exploration: with probability ϵ the student will give a random answer, and with probability $1 - \epsilon$ it will give the answer that its model assigns the largest probability. This method enables the model to explore the space of actions and to potentially discover correct answers.
- data balancing: cluster the set of teacher responses t and then balance training across the clusters equally.[2] This is a type of experience replay (Mnih et al., 2013) but sampling with an evened distribution. Balancing stops part of the distribution dominating the learning. For example, if the model is not exposed to sufficient positive and negative feedback, and one class overly dominates, the learning process degenerates to a model that always predicts the same output regardless of its input.

5 EXPERIMENTS

Experiments are first conducted using our simulator, and then using Amazon Mechanical Turk with real human subjects taking the role of the teacher[3].

5.1 SIMULATOR

Online Experiments In our first experiments, we considered both the bAbI and WikiMovies tasks and varied batch size, random exploration rate ϵ, and type of model. Figure 3 and Figure 4 shows (Task 6) results on bAbI and WikiMovies. Other tasks yield similar conclusions and are reported in the appendix.

Overall, we obtain the following conclusions:

- In general RBI and FP do work in a reinforcement learning setting, but can perform better with random exploration.
- In particular RBI can fail without exploration. RBI needs random noise for exploring labels otherwise it can get stuck predicting a subset of labels and fail.

[2]In the simulated data, because the responses are templates, this can be implemented by first randomly sampling the response, and then randomly sampling a story with that response; we keep the history of all stories seen from which we sample. For real data slightly more sophisticated clustering should be used.

[3] Code and data are available at `https://github.com/facebook/MemNN/tree/master/HITL`.

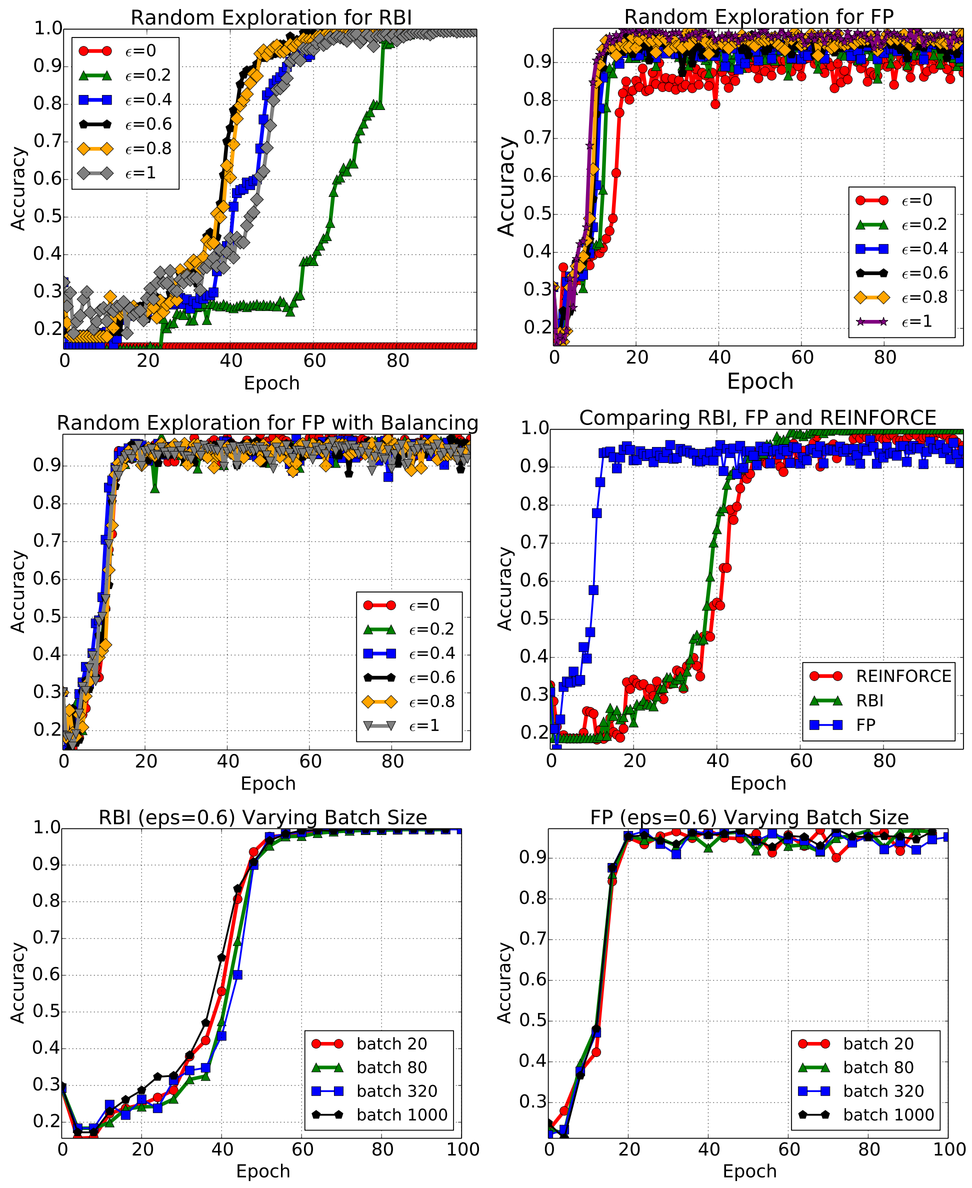


Figure 3: **Training epoch vs. test accuracy for bAbI (Task 6) varying exploration ϵ and batch size.** Random exploration is important for both reward-based (RBI) and forward prediction (FP). Performance is largely independent of batch size, and RBI performs similarly to REINFORCE. Note that supervised, rather than reinforcement learning, with gold standard labels achieves 100% accuracy on this task.

- REINFORCE obtains similar performance to RBI with optimal ϵ.
- FP with balancing or with exploration via ϵ both outperform FP alone.
- For both RBI and FP, performance is largely independent of online batch size.

Dataset Batch Size Experiments Given that larger online batch sizes appear to work well, and that this could be important in a real-world data collection setup where the same model is deployed to gather a large amount of feedback from humans, we conducted further experiments where the batch size is exactly equal to the dataset size and for each batch training is completed to convergence.

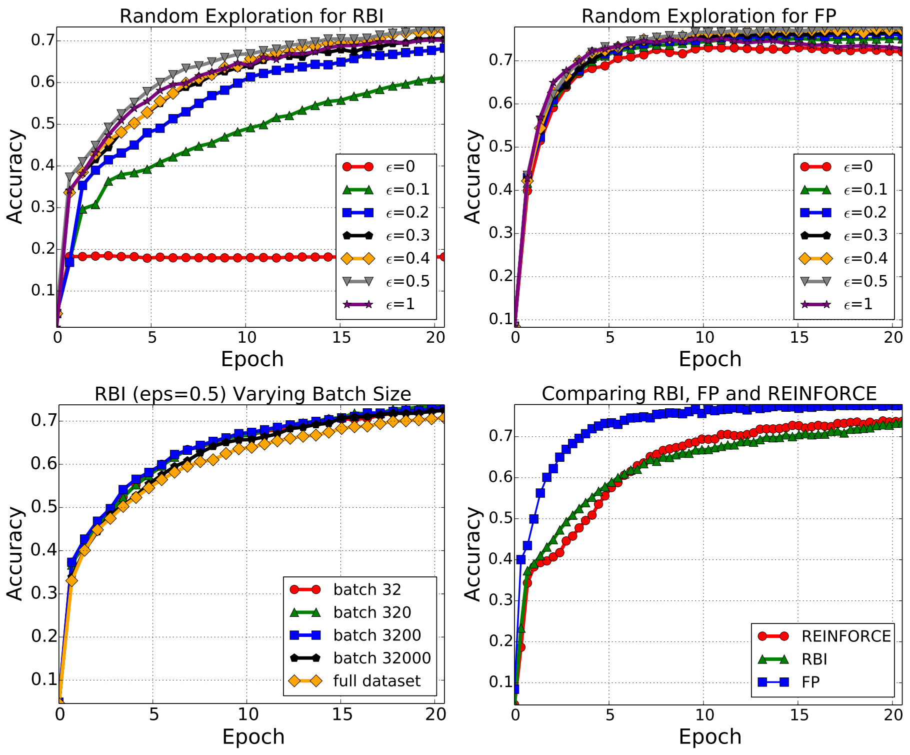


Figure 4: **WikiMovies**: Training epoch vs. test accuracy on Task 6 varying (top left panel) exploration rate ϵ while setting batch size to 32 for RBI, (top right panel) for FP, (bottom left) batch size for RBI, and (bottom right) comparing RBI, REINFORCE and FP with $\epsilon = 0.5$. The model is robust to the choice of batch size. RBI and REINFORCE perform comparably. Note that supervised, rather than reinforcement learning, with gold standard labels achieves 80% accuracy on this task (Weston, 2016).

After the model has been trained on the dataset, it is deployed to collect a new dataset of questions and answers, and the process is repeated. Table 1 reports test error at each iteration of training, using the bAbI Task 6 as the case study (see the appendix for results on other tasks). The following conclusions can be made for this setting:

- RBI improves in performance as we iterate. Unlike in the online case, RBI does not need random exploration. We believe this is because the first batch, which is collected with a randomly initialized model, contains enough variety of examples with positive rewards that the model does not get stuck predicting a subset of labels.
- FP is not stable in this setting. This is because once the model gets very good at making predictions (at the third iteration), it is not exposed to a sufficient number of negative responses anymore. From that point on, learning degenerates and performance drops as the model always predicts the same responses. At the next iteration, it will recover again since it has a more balanced training set, but then it will collapse again in an oscillating behavior.
- FP does work if extended with balancing or random exploration with sufficiently large ϵ.
- RBI+FP also works well and helps with the instability of FP, alleviating the need for random exploration and data balancing.

Overall, our simulation results indicate that while a bot can be effectively trained fully online from bot-teacher interactions, collecting real dialogue data in batches (which is easier to collect and iterate experiments over) is also a viable approach. We hence pursue the latter approach in our next set of experiments.

Iteration	1	2	3	4	5	6
Imitation Learning	0.24	0.23	0.23	0.22	0.23	0.23
Reward Based Imitation (RBI)	0.74	0.87	0.90	**0.96**	**0.96**	**0.98**
Forward Pred. (FP)	**0.99**	**0.96**	**1.00**	0.30	**1.00**	0.29
RBI+FP	**0.99**	**0.96**	**0.97**	**0.95**	0.94	**0.97**
FP (balanced)	**0.99**	**0.97**	**0.97**	**0.97**	**0.97**	**0.97**
FP (rand. exploration $\epsilon = 0.25$)	**0.96**	0.88	0.94	0.26	0.64	**0.99**
FP (rand. exploration $\epsilon = 0.5$)	**0.98**	**0.98**	**0.99**	**0.98**	**0.95**	**0.99**

Table 1: Test accuracy of various models per iteration in the dataset batch size case (using batch size equal to the size of the full training set) for bAbI, Task 6. Results > 0.95 are in bold.

Relation to experiments in Weston (2016) As described in detail in Section 2 the datasets we use in our experiments were introduced in (Weston et al., 2015). However, that work involved constructing pre-built fixed policies (and hence, datasets), rather than training the learner in a reinforcement/interactive learning using a simulator, as in our work. They achieved this by choosing an omniscient (but deliberately imperfect) labeler that gets π_{acc} examples always correct (the paper looked at values 1%, 10% and 50%). In a realistic setting one does not have access to an omniscient labeler, one has to learn a policy completely from scratch, online, starting with a random policy, as we do here. Nevertheless, it is possible to compare our *learnt* policies to those results because we use the same train/valid/test splits.

The clearest comparison comparison is via Table 1, where the policy is learnt using batch iterations of the dataset, updating the policy on each iteration. Weston et al. (2015) can be viewed as training only one iteration, with a pre-built policy, as explained above, where 59%, 81% and 99% accuracy was obtained for RBI for π_{acc} with 1%, 10% and 50% respectively[4]. While π_{acc} of 50% is good enough to solve the task, lower values are not. In this work a random policy begins with 74% accuracy on the first iteration, but importantly on each iteration the policy is updated and improves, with values of 87%, 90% on iterations 2 and 3 respectively, and 98% on iteration 6. This is a key differentiator to the work of (Weston et al., 2015) where such improvement was not shown. We show that such online learning works for both reward-based numerical feedback and for forward prediction methods using textual feedback (as long as balancing or random exploration is performed sufficiently). The final performance outperforms most values of π_{acc} from Weston et al. (2015) unless π is so large that the task is already solved. This is a key contribution of our work.

Similar conclusions can be made for Figures 3 and 4. Despite our initial random policy starting at close to 0% accuracy, if random exploration $\epsilon \geq 0.2$ is employed then after a number of epochs the performance is better than most values of π_{acc} from Weston et al. (2015), e.g. compare the accuracies given in the previous paragraph (59%, 81% and 99%) to Figure 3, top left.

5.2 Human Feedback

We employed Turkers to both ask questions and then give textual feedback on the bot's answers, as described in Section 3.2. Our experimental protocol was as follows. We first trained a MemN2N using supervised (i.e., imitation) learning on a training set of 1000 questions produced by Turkers and using the known correct answers provided by the original dataset (and no textual feedback). Next, using the trained policy, we collected textual feedback for the responses of the bot for an additional 10,000 questions. Examples from the collected dataset are given in Figure 2. Given this dataset, we compare various models: RBI, FP and FP+RBI. As we know the correct answers to the additional questions, we can assign a positive reward to questions the bot got correct. We hence measure the impact of the sparseness of this reward signal, where a fraction r of additional examples have rewards. The models are tested on a test set of $\sim$8,000 questions (produced by Turkers), and hyperparameters are tuned on a similarly sized validation set. Note this is a harder task than the WikiMovies task in the simulator due to the use natural language from Turkers, hence lower test performance is expected.

[4]Note, this is not the same as a randomly initialized neural network policy, because due to the synthetic construction with an omniscient labeler the labels will be balanced. In our work, we learn the policy from randomly initialized weights which are updated as we learn the policy.

Results are given in Table 2. They indicate that both RBI and FP are useful. When rewards are sparse, FP still works via the textual feedback while RBI can only use the initial 1000 examples when $r = 0$. As FP does not use numericalrewards at all, it is invariant to the parameter r. The combination of FP and RBI outperforms either alone.

Model	$r = 0$	$r = 0.1$	$r = 0.5$	$r = 1$
Reward Based Imitation (RBI)	0.333	0.340	0.365	0.375
Forward Prediction (FP)	0.358	0.358	0.358	0.358
RBI+FP	0.431	0.438	0.443	0.441

Table 2: **Incorporating Feedback From Humans via Mechanical Turk.** Textual feedback is provided for 10,000 model predictions (from a model trained with 1k labeled training examples), and additional sparse binary rewards (fraction r of examples have rewards). Forward Prediction and Reward-based Imitation are both useful, with their combination performing best.

We also conducted additional experiments comparing with (i) synthetic feedback and (ii) the fully supervised case which are given in Appendix C.1. They show that the results with human feedback are competitive with these approaches.

6 Conclusion

We studied dialogue learning of end-to-end models using textual feedback and numerical rewards. Both fully online and iterative batch settings are viable approaches to policy learning, as long as possible instabilities in the learning algorithms are taken into account. Secondly, we showed for the first time that the recently introduced FP method can work in both an online setting and on real human feedback. Overall, our results indicate that it is feasible to build a practical pipeline that starts with a model trained on an initial fixed dataset, which then learns from interactions with humans in a (semi-)online fashion to improve itself. Future research should work towards doing this in a never-ending learning setup.

References

Mohammad Amin Bassiri. Interactional feedback and the impact of attitude and motivation on noticing l2 form. *English Language and Literature Studies*, 1(2):61, 2011.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

Leon Bottou, Jonas Peters, Denis X. Quionero-Candela, Joaquin amd Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14:3207–3260, 2013.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*, 2015.

Milica Gašic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. Pomdp-based dialogue manager adaptation to extended domains. In *Proceedings of SIGDIAL*, 2013.

Milica Gašic, Dongho Kim, Pirros Tsiakoulis, Catherine Breslin, Matthew Henderson, Martin Szummer, Blaise Thomson, and Steve Young. Incremental on-line adaptation of pomdp-based dialogue managers to extended domains. *In Proceedings on InterSpeech*, 2014.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. Learning dialogue strategies within the markov decision process framework. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pp. 72–79. IEEE, 1997.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1): 11–23, 2000.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Roberto Pieraccini, David Suendermann, Krishna Dayanidhi, and Jackson Liscombe. Are we there yet? research in commercial spoken dialog systems. In *International Conference on Text, Speech and Dialogue*, pp. 3–13. Springer, 2009.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(02):97–126, 2006.

Satinder Singh, Michael Kearns, Diane J Litman, Marilyn A Walker, et al. Empirical evaluation of a reinforcement learning spoken dialogue system. In *AAAI/IAAI*, pp. 645–651, 2000.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*, 2016.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

Marilyn A. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, 12:387–416, 2000.

Marilyn A Walker, Rashmi Prasad, and Amanda Stent. A trainable generator for recommendations in multimodal dialog. In *INTERSPEECH*, 2003.

Margaret G Werts, Mark Wolery, Ariane Holcombe, and David L Gast. Instructive feedback: Review of parameters and effects. *Journal of Behavioral Education*, 5(1):55–75, 1995.

Jason Weston. Dialog-based language learning. *arXiv preprint arXiv:1604.06045*, 2016.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

Wojciech Zaremba and Ilya Sutskever. Reinforcement learning neural turing machines. *arXiv preprint arXiv:1505.00521*, 362, 2015.

A Further Simulator Task Details

The tasks in Weston (2016) were specifically:
- **Task 1**: The teacher tells the student exactly what they should have said (supervised baseline).
- **Task 2**: The teacher replies with positive textual feedback and reward, or negative textual feedback.
- **Task 3**: The teacher gives textual feedback containing the answer when the bot is wrong.
- **Task 4**: The teacher provides a hint by providing the class of the correct answer, e.g., "No it's a movie" for the question "which movie did Forest Gump star in?".
- **Task 5**: The teacher provides a reason why the student's answer is wrong by pointing out the relevant supporting fact from the knowledge base.
- **Task 6**: The teacher gives positive reward only 50% of the time.
- **Task 7**: Rewards are missing and the teacher only gives natural language feedback.
- **Task 8**: Combines Tasks 1 and 2 to see whether a learner can learn successfully from both forms of supervision at once.
- **Task 9**: The bot asks questions of the teacher about what it has done wrong.
- **Task 10**: The bot will receive a hint rather than the correct answer after asking for help.

We refer the readers to (Weston, 2016) for more detailed descriptions and the motivation behind these tasks. The difference in our system is that the model can be trained on-the-fly via the simulator: after receiving feedback and/or rewards, the model can update itself and apply its learning to the next episode. We present results on Tasks 2, 3 and 4 in this appendix

B Instructions given to Turkers

These are the instructions given for the textual feedback mechanical turk task (we also constructed a separate task to collect the initial questions, not described here):

Title: Write brief responses to given dialogue exchanges (about 15 min)

Description: Write a brief response to a student's answer to a teacher's question, providing feedback to the student on their answer.

Instructions:

Each task consists of the following triplets:

1. a question by the teacher
2. the correct answer(s) to the question (separated by "OR")
3. a proposed answer in reply to the question from the student

Consider the scenario where you are the teacher and have already asked the question, and received the reply from the student. Please compose a brief response giving feedback to the student about their answer. The correct answers are provided so that you know whether the student was correct or not.

For example, given 1) question: "what is a color in the united states flag?"; 2) correct answer: "white, blue, red"; 3) student reply: "red", your response could be something like "that's right!"; for 3) reply: "green", you might say "no that's not right" or "nope, a correct answer is actually white".

Please vary responses and try to minimize spelling mistakes. If the same responses are copied/pasted or overused, we'll reject the HIT.

Avoid naming the student or addressing "the class" directly.

We will consider bonuses for higher quality responses during review.

T : Which movie did Tom Hanks star in ? S : Forrest Gump	T : Which movie did Tom Hanks star in ? S : Brad Pitt.
Task 1: Imitating an Expert Student S: Forrest Gump T: (no response)	Task 1: Imitating an Expert Student S: Forrest Gump T: (no response)
Task 2: Positive and Negative Feedback T: Yes, that's right! (+)	Task 2: Positive and Negative Feedback T: No, that's incorrect!
Task 3: Answers Supplied by Teacher T: Yes, that is correct. (+)	Task 3: Answers Supplied by Teacher T: No, the answer is Forrest Gump !
Task 4: Hints Supplied by Teacher T: Correct! (+)	Task 4: Hints Supplied by Teacher T: No, it's a movie !
Task 5: Supporting Facts Supplied by Teacher T: That's right. (+)	Task 5: Supporting Facts Supplied by Teacher T: No, because Forrest Gump starred actors Tom Hanks, Robin Wright, Gary Sinise !

Task 6: Partial Feedback
if random(0,1)<0.5 then
    T: That's correct. (+)
else T: That's correct.
	Task 6: Partial Feedback
	T: Sorry, wrong.

Task 7: No Feedback T: Yes.	Task 7: No Feedback T: No.

Task 8: Imitation and Feedback Mixture
if random(0,1)<0.5 then
    T: Yes, that's right! (+)
else T: (no response)
	Task 8: Imitation and Feedback Mixture
	if random(0,1)<0.5 then
	    T: Wrong.
	else S: Forrest Gump

Task 9: Asking For Corrections T: Correct! (+)	Task 9: Asking For Corrections T: No, that's wrong. S: Can you help me? T: Forrest Gump !
Task 10: Asking For Supporting Facts T: Yes, that's right! (+)	Task 10: Asking For Supporting Facts T: Sorry, that's not it. S: Can you help me? T: A relevant fact is that Forrest Gump starred actors Tom Hanks, Robin Wright, Gary Sinise !

Figure 5: The ten tasks our simulator implements, which evaluate different forms of teacher response and binary feedback. In each case the same example from WikiMovies is given for simplicity, where the student answered correctly for all tasks (left) or incorrectly (right). Red text denotes responses by the bot with S denoting the bot. Blue text is spoken by the teacher with T denoting the teacher's response. For imitation learning the teacher provides the response the student should say denoted with S in Tasks 1 and 8. A (+) denotes a positive reward.

C Additional Experiments

Iteration	1	2	3	4	5	6
Imitation Learning	0.24	0.23	0.23	0.23	0.25	0.25
Reward Based Imitation (RBI)	**0.95**	**0.99**	**0.99**	**0.99**	**1.00**	**1.00**
Forward Pred. (FP)	**1.00**	0.19	0.86	0.30	**99**	0.22
RBI+FP	**0.99**	**0.99**	**0.99**	**0.99**	**99**	**0.99**
FP (balanced)	**0.99**	**0.97**	**0.98**	**0.98**	**0.96**	**0.97**
FP (rand. exploration $\epsilon = 0.25$)	**0.99**	0.91	0.93	0.88	0.94	0.94
FP (rand. exploration $\epsilon = 0.5$)	**0.98**	0.93	**0.97**	**0.96**	**0.95**	**0.97**

Table 3: Test accuracy of various models in the dataset batch size case (using batch size equal to the size of the full training set) for bAbI, task 3. Results > 0.95 are in bold.

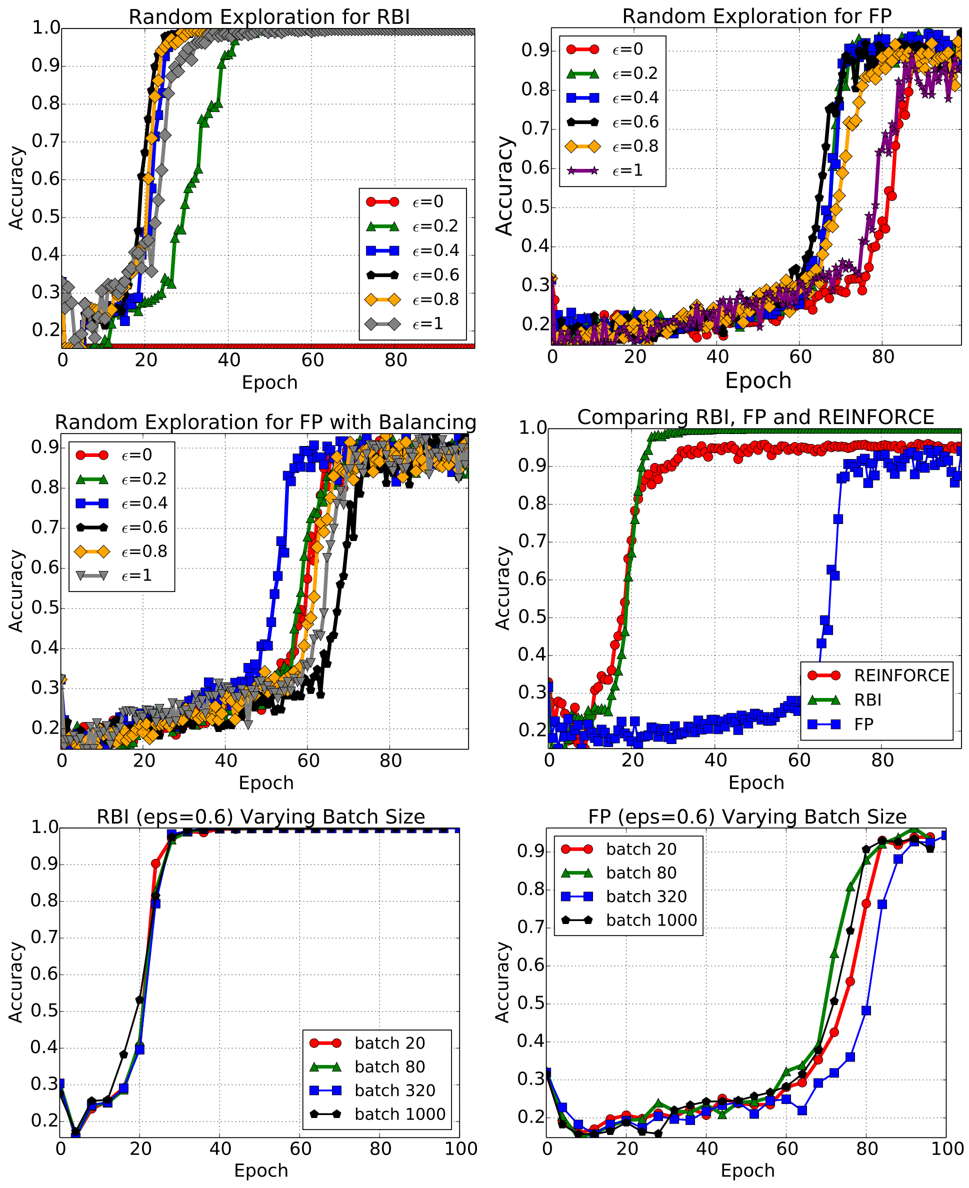


Figure 6: **Training epoch vs. test accuracy for bAbI (Task 2) varying exploration ϵ and batch size.**

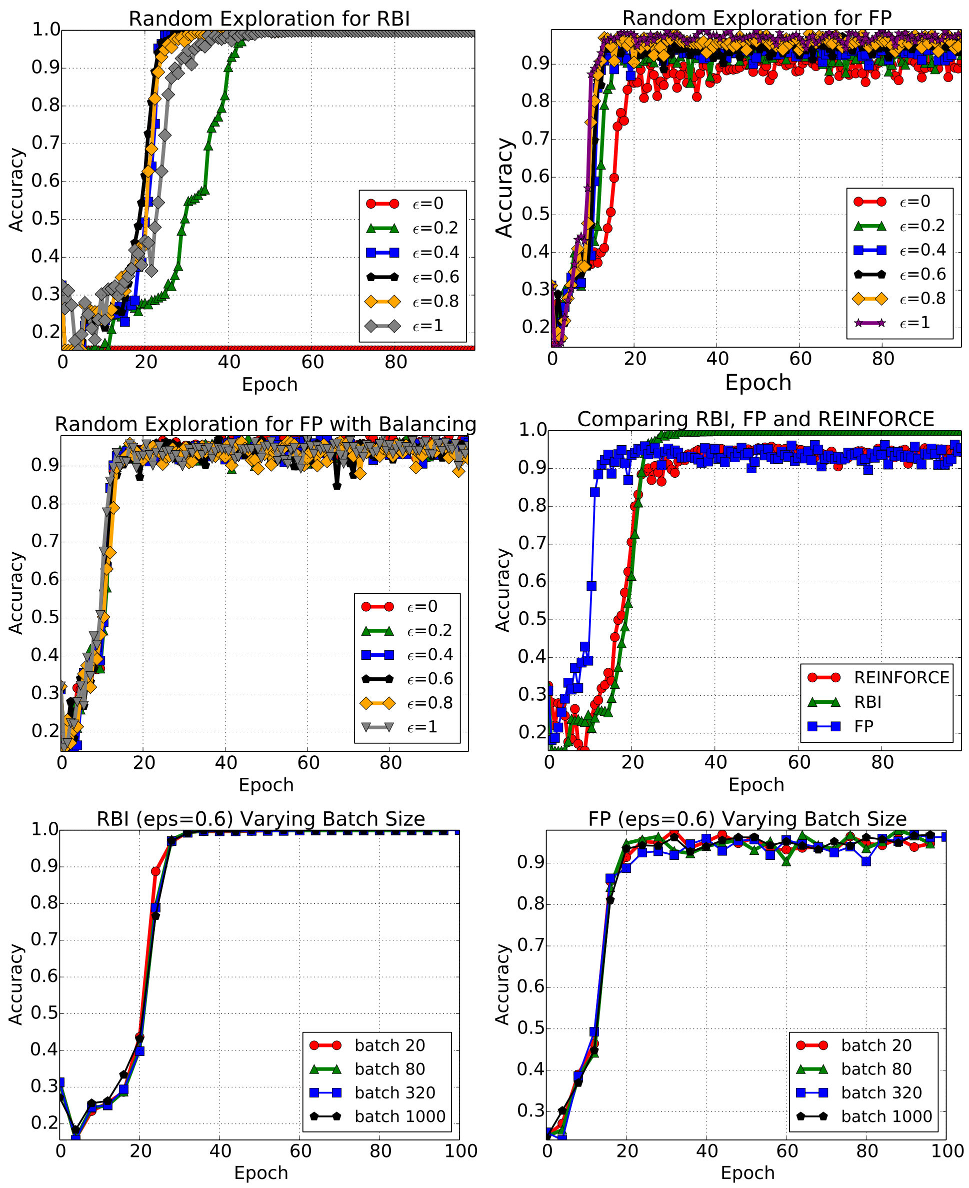


Figure 7: **Training epoch vs. test accuracy for bAbI (Task 3) varying exploration ϵ and batch size.** Random exploration is important for both reward-based (RBI) and forward prediction (FP).

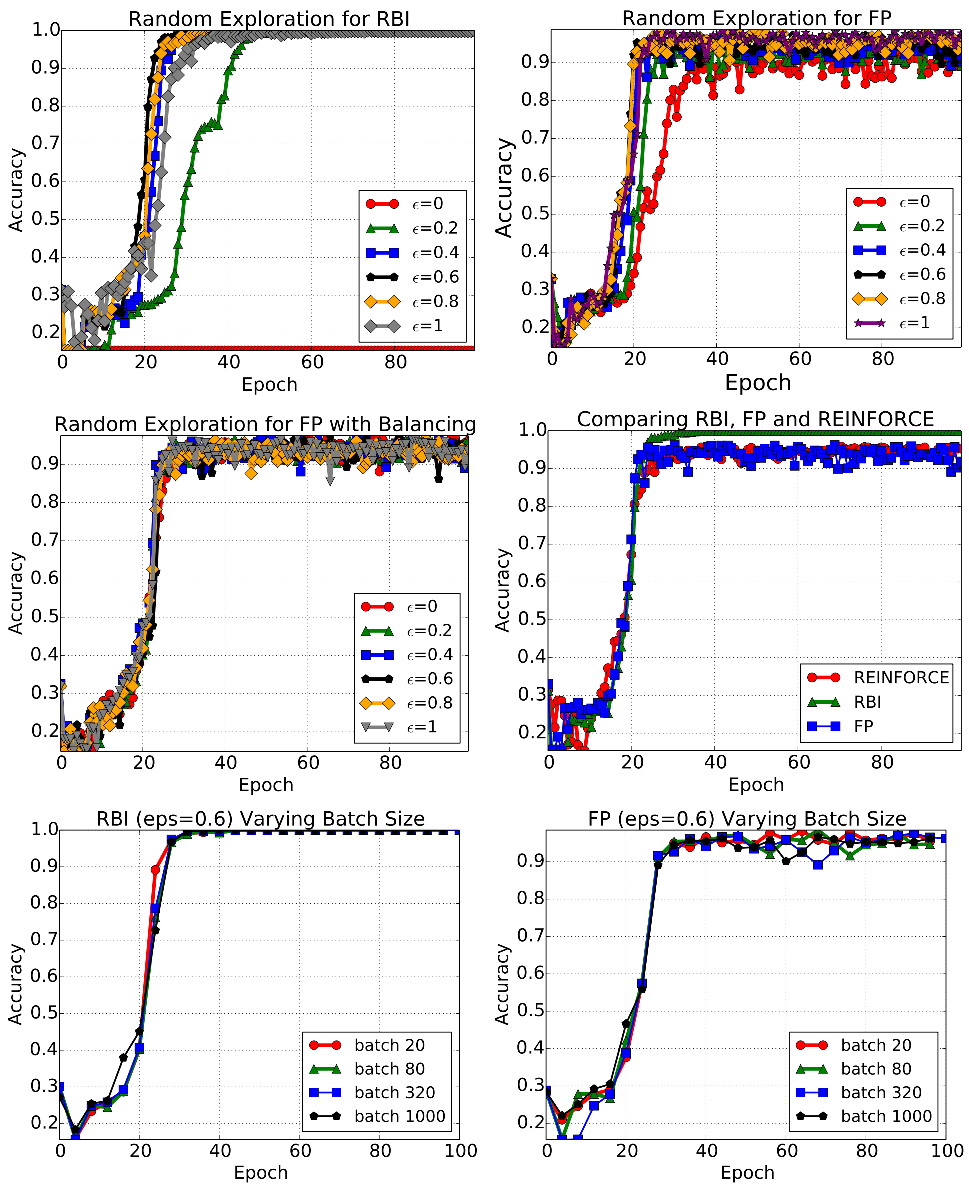
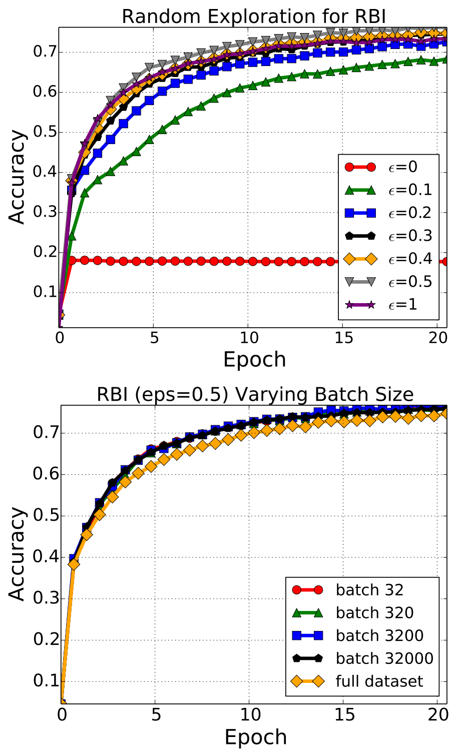


Figure 8: **Training epoch vs. test accuracy for bAbI (Task 4) varying exploration ϵ and batch size.** Random exploration is important for both reward-based (RBI) and forward prediction (FP).



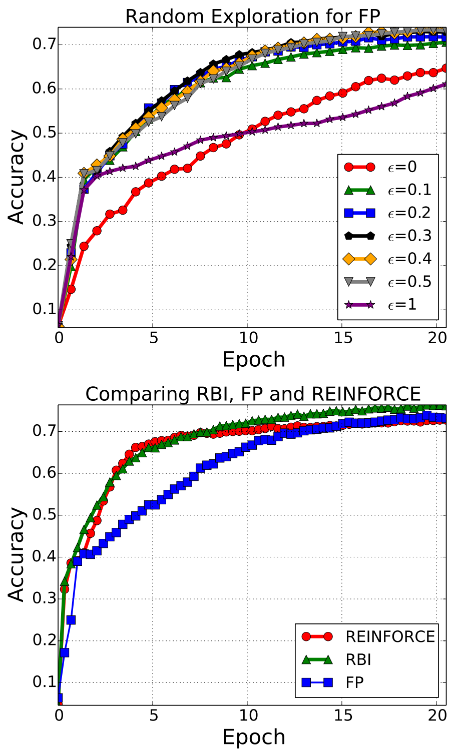


Figure 9: **WikiMovies**: Training epoch vs. test accuracy on Task 2 varying (top left panel) exploration rate ϵ while setting batch size to 32 for RBI, (top right panel) for FP, (bottom left) batch size for RBI, and (bottom right) comparing RBI, REINFORCE and FP setting $\epsilon = 0.5$. The model is robust to the choice of batch size. RBI and REINFORCE perform comparably.

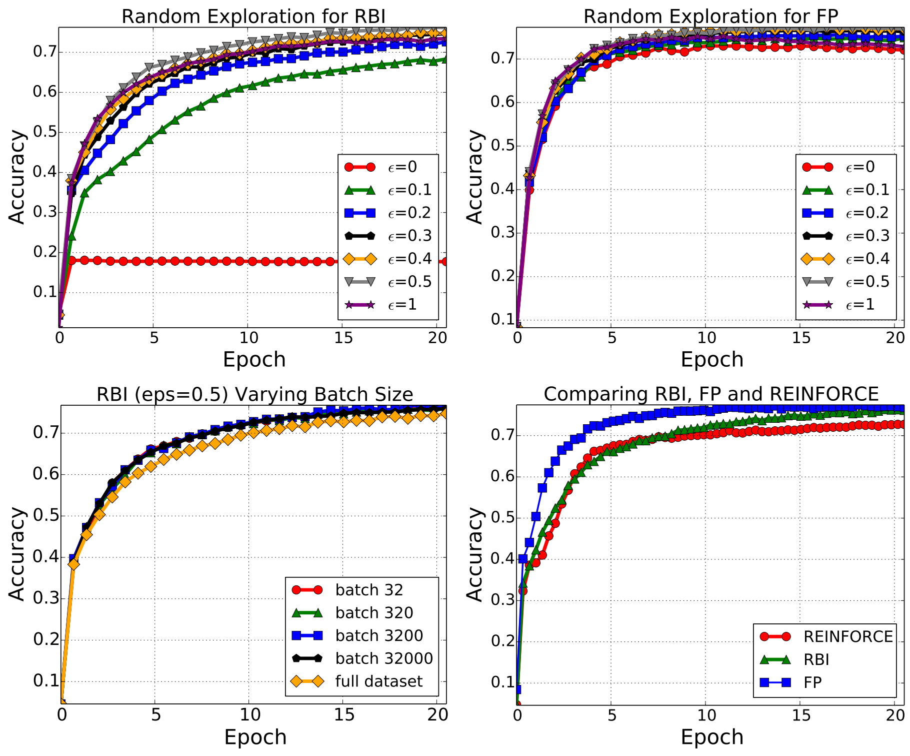


Figure 10: **WikiMovies**: Training epoch vs. test accuracy on Task 3 varying (top left panel) exploration rate ϵ while setting batch size to 32 for RBI, (top right panel) for FP, (bottom left) batch size for RBI, and (bottom right) comparing RBI, REINFORCE and FP setting $\epsilon = 0.5$. The model is robust to the choice of batch size. RBI and REINFORCE perform comparably.

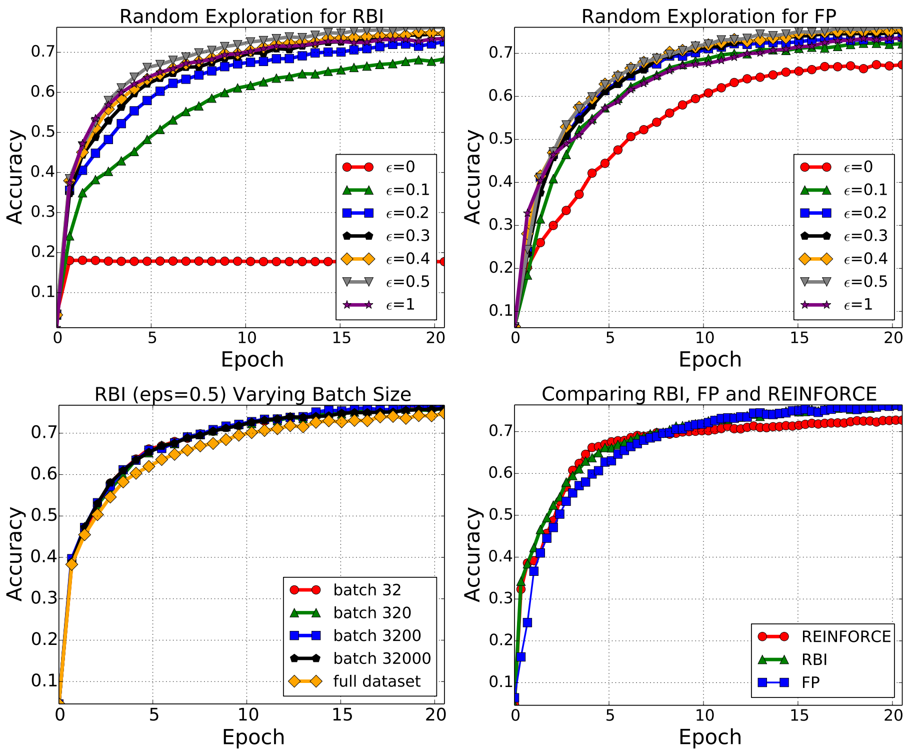


Figure 11: **WikiMovies**: Training epoch vs. test accuracy on Task 4 varying (top left panel) exploration rate ϵ while setting batch size to 32 for RBI, (top right panel) for FP, (bottom left) batch size for RBI, and (bottom right) comparing RBI, REINFORCE and FP setting $\epsilon = 0.5$. The model is robust to the choice of batch size. RBI and REINFORCE perform comparably.

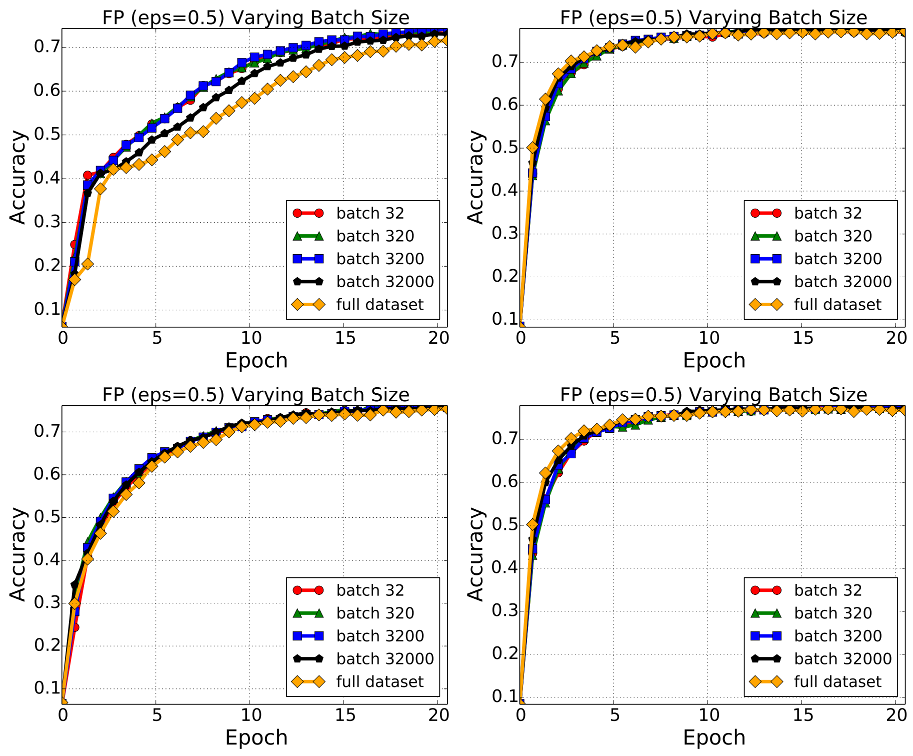


Figure 12: **WikiMovies**: Training epoch vs. test accuracy with varying batch size for FP on Task 2 (top left panel), 3 (top right panel), 4 (bottom left panel) and 6 (top right panel) setting $\epsilon = 0.5$. The model is robust to the choice of batch size.

C.1 Additional Experiments For Mechanical Turk Setup

In the experiment in Section 5.2 we conducted experiments with real human feedback. Here, we compare this to a form of synthetic feedback, mostly as a sanity check, but also to see how much improvement we can get if the signal is simpler and cleaner (as it is synthetic). We hence constructed synthetic feedback for the 10,000 responses, using either Task 2 (positive or negative feedback), Task 3 (answers provided by teacher) or a mix (Task 2+3) where we use one or the other for each example (50% chance of each). The latter makes the synthetic data have a mixed setup of responses, which more closely mimics the real data case. The results are given in Table 4. The RBI+FP combination is better using the synthetic data than the real data with Task 2+3 or Task 3, which is to be expected, but the real data is competitive, despite the difficulty of dealing with its lexical and semantic variability. The real data is better than using Task 2 synthetic data.

For comparison purposes, we also ran a supervised (imitation learning) MemN2N on different sized training sets of turker authored questions with gold annotated labels (so, there are no numerical rewards or textual feedback, this is a pure supervised setting). The results are given in Table 5. They indicate that RBI+FP and even FP alone get close to the performance of fully supervised learning.

Model	$r = 0$	$r = 0.1$	$r = 0.5$	$r = 1$
Reward Based Imitation (RBI)	0.333	0.340	0.365	0.375
Forward Prediction (FP) [real]	0.358	0.358	0.358	0.358
RBI+FP [real]	0.431	0.438	0.443	0.441
Forward Prediction (FP) [synthetic Task 2]	0.188	0.188	0.188	0.188
Forward Prediction (FP) [synthetic Task 2+3]	0.328	0.328	0.328	0.328
Forward Prediction (FP) [synthetic Task 3]	0.361	0.361	0.361	0.361
RBI+FP [synthetic Task 2]	0.382	0.383	0.407	0.408
RBI+FP [synthetic Task 2+3]	0.459	0.465	0.464	0.478
RBI+FP [synthetic Task 3]	0.473	0.486	0.490	0.494

Table 4: **Incorporating Feedback From Humans via Mechanical Turk: comparing real human feedback to synthetic feedback.** Textual feedback is provided for 10,000 model predictions (from a model trained with 1k labeled training examples), and additional sparse binary rewards (fraction r of examples have rewards). We compare real feedback (rows 2 and 3) to synthetic feedback when using FP or RBI+FP (rows 4 and 5).

Train data size	1k	5k	10k	20k	60k
Supervised MemN2N	0.333	0.429	0.476	0.526	0.599

Table 5: **Fully Supervised (Imitation Learning) Results on Human Questions**

	$r = 0$	$r = 0.1$	$r = 0.5$	$r = 1$
$\epsilon = 0$	0.499	0.502	0.501	0.502
$\epsilon = 0.1$	0.494	0.496	0.501	0.502
$\epsilon = 0.25$	0.493	0.495	0.496	0.499
$\epsilon = 0.5$	0.501	0.499	0.501	0.504
$\epsilon = 1$	0.497	0.497	0.498	0.497

Table 6: **Second Iteration of Feedback** Using synthetic textual feedback of synthetic Task2+3 with the RBI+FP method, an additional iteration of data collection of 10k examples, varying sparse binary reward fraction r and exploration ϵ. The performance of the first iteration model was 0.478.

C.2 Second Iteration of Feedback

We conducted experiments with an additional iteration of data collection for the case of binary rewards and textual feedback using the synthetic Task 2+3 mix. We selected the best model from the previous training, using RBI+FP with $r = 1$ which previously gave a test accuracy of 0.478 (see Table 4). Using that model as a predictor, we collected an additional 10,000 training examples.

We then continue to train our model using the original 1k+10k training set, plus the additional 10k. As before, we report the test accuracy varying r on the additional collected set. We also report the performance from varying ϵ, the proportion of random exploration of predictions on the new set. The results are reported in Table 6. Overall, performance is improved in the second iteration, with slightly better performance for large r and $\epsilon = 0.5$. However, the improvement is mostly invariant to those parameters, likely because FP takes advantage of feedback from incorrect predictions in any case.