Transfer Learning for Reinforcement Learning in Dialogue Systems

Dan Iter [1] **Allan Jiang** [1] **Li Deng** [1]

Abstract

With the proliferation of speech based assistants and chat bots, there is a growing demand for natural language interfaces to a wide variety of applications. Reinforcement learning has shown to be a powerful method of modeling a task-based dialogue. However, there are two major challenges to widely deploying reinforcement learning agents in these settings: training an agent can require a large amount of data and a model for one task does not generalize well to other tasks. We experimented with a system for transferring models between tasks using transfer learning, and presented our exploration in retrain and fine-tuning these models using user simulation to generate artificial dialogues. We show that we were able to improve the original system in (Li et al., 2016b) by implementing a larger and faster *DQN* model in *Tensorflow*, with a 0.82 success rate slightly higher than original 0.80 success rate. Finally the problem we encountered in transfer learning is discussed, and our efforts in approaching the problem of unifying state representations are presented.

1. Introduction

1.1. Task Based Dialogues

With the growing popularity of speech based assistants such as *Siri*, Google *Home* and Amazon *Echo* and the growing trend of chat bots being integrated into our every day applications like Slack and Facebook M, there is a growing demand for natural language interfaces to a wide variety of applications.

Reinforcement learning has shown to be a powerful method of modeling a dialogue (M. Gaic & Young., 2013; Milica Gasic & Young, 2017), in particular for specific tasks, such as booking flights or IT help desk support. In this setting a user interacts with a natural language interface to an agent that must take the correct actions to complete the task. The agent must also track states in the conversation to inform its next action. If the task is completed such the correct data was used and the task executed, such as booking the correct flight for the user, then we say the task and dialogue were successful and we give the agent a positive reward. Alternatively, if the agent is unable to understand what the user wants, the user feels that the agent is not understanding her correctly or the final action executed is not correct, we say that the dialogue failed and we give the agent a negative reward.

[1]Stanford University, Stanford, USA.

1.2. Slot Filling Systems

One of the most fundamental tasks for task-based dialogue systems is to act as an interface for slot-filling systems. In a slot-filling system, the agent must collect specific data from a user and fill the relevant slots to be able to complete a task. For example, when buying movie tickets, the agent must find out details such as movie title, theater, time, number of tickets, etc. Hence, any slot-filling system can be viewed as a system that fills out a set of required and optional fields, much like filling out a form.

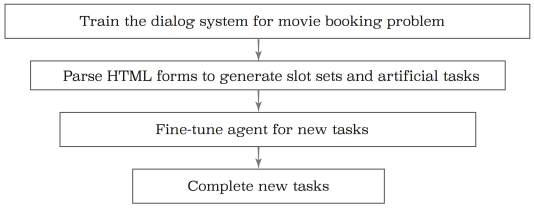

Figure 1. Transfer Learning: movie to flight

We use HTML forms to represent the underlying tasks that the agent must complete and submit. We hope to show that given an HTML form we can quickly build a natural language interface to that form using a reinforcement learning agent. We accomplish this by initializing the RL model with weights from another pre-trained model for a different task. We then retrain and fine-tune the model using generated dialogues using our user simulation which generates user goals and actions based on fabricated training data for the HTML form.

1.3. Reinforcement Learning for Task Completion

Reinforcement learning is a natural model for learning an optimal strategy for task-based dialogue systems.

- **Modeling**: The process to be optimized is a set of dialogue actions taken by the agent (and responses by the user), given a state of the conversation history that's occurred so far.
- **Reward**: The reward is based on if the task completed based on the dialogue is successful or not. Usually there is a negative reward for each turn in the dialogue to encourage shorter, more efficient dialogues.
- **State**: The state captures various features about the dialogue, such as whose turn it is, how many turns have been taken and what were the previous actions.

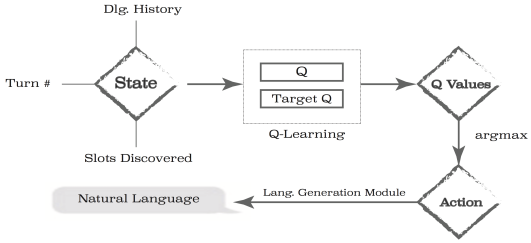

Figure 2. Q-learning for Task Completion Agent

- **Action**: Finally, the actions of the *RL* agent are mapped to dialogue actions. This means that given a state, the *RL* agent must choose what is the next action to take in the dialogue. Actions can be either continuous or discrete. For continuous actions, the agent would output some "action" that is fed into an *LSTM* which outputs natural language. To limit the scope, we focus on discrete dialogue actions, where each action is based on requesting or informing a slot. Specifically, one dialogue action may be asking for the number of travelers for a flight or providing the available flight times tomorrow. We convert the numerical action into natural language by mapping the actions to sentence templates.
- **Success**: success of a task is defined to be the completion of the agent in filling all slots required in the template predefined.

In task-based dialogue systems we focus on filling a number of slots with information from the user. The most important features in the state are which slots have already been filled and which ones are being requested or provided by the user. For example, a user may either ask for when a movie is playing (a *request* slot) or may state that she wants to see a movie tomorrow at 9pm (an *inform* slot).

1.4. Challenges - Data and Generalization

There are two challenges with regards to data collection and model generalization:

- **Data Collection**:Traditionally, dialogue systems train reinforcement learning agents with huge amounts of real human dialogues. Collecting a dataset can be prohibitively expensive because it requires real human participation and requires manual labeling. The problem is further exacerbated when training a model for multiple tasks, since you need to collect a large dataset for each task. Furthermore, real data may have degenerate patterns and biases that may negatively affect the training. For example, in many open dialogue systems, statements such as "I don't know" are very common, and thus push models to fall into pathological states where they only emit this trivial output in an attempt to mimic human speakers.

  We address the challenge of training on large datasets but generating artificial dialogues with a user simulation, as done in (Li et al., 2016b). The user simulation leverages a Natural Language Generation (*NLG*) model to answer questions from an agent to simulate a real dialogue. While we still need to engineer the *NLG* by writing dialogue templates, it makes collection of usable training data tractable. Ultimately, we are able to train an RL agent on a new task with no additional real training data.
- **Model Generalization**: The second main challenge when training dialogue systems is that you must train a separate model for each task. Traditionally, most dialogue systems are based on templates and rules that can be impossible to translate to a new task without rebuilding it from scratch. Since our RL agent is based on a neural network, we hypothesis that the lower layers of the neural network are learning fundamental properties of the dialogue. Furthermore, since we are focusing on a simple class of task based dialogues, we expect there to be large overlap in terms of the core structure of the dialogues. Therefore, we use transfer learning to apply the structure learned from one task to another. This significantly decreases training time and makes the agent applicable to most slot filling applications.

1.5. Summary

Many consumer and corporate applications can benefit from improved dialogue systems. A popular approach for building and training dialogue systems is to use reinforcement agents. The need for large datasets makes training such systems expensive and generalization between tasks has not been extensively studied in the literature. We aim

to address both of these challenges by using a user simulator to generate artificial training data; and explore transfer learning to simplify the training of agents for new tasks with no high quality data. We demonstrate our exploration in building such a dialogue system to transfers the knowledge from a movie booking task to a flight booking task using transfer learning and to generated training data from a user simulation.

2. Related Work

Abundant research has been conducted on task-oriented conversation agents. Related work may be categorized into the following types:

Dialogue Generation A difficult aspect of neural dialogue generation is generating dialogs that are not "short-sighted". Short-sighted dialogs are chosen without regard to influences on future outcomes in the conversation. In a recent work titled *Deep Reinforcement Learning for Dialogue Generation* Li et al. represent a state by the previous two turns (Li et al., 2016a). The reward is defined as a weighted sum of ease of answering, variety of information flow and semantic coherence, promoting the influence of future outcomes of the conversation. The system is task-free, and is evaluated by the length of dialogue, diversity of content, as well as with human evaluation.

Task-oriented Conversation A recent work on task-oriented conversations proposed an end-to-end deep reinforcement learning framework backed by a structured database to enable agents to run simulated conversations for booking movie tickets (Li et al., 2017). They propose three components for their framework:

- **Natural Language Generation** (NLG) module: generate text corresponding to a user's dialogue actions,
- **Natural Language Understanding** (NLU) module: parse the text inputs into semantic frames,
- **Dialogue Management** (DM) module: (including a state tracker and a policy learner) to accumulate semantics, track the dialogue states, and produce the next dialogue action.

Deep RL for Task-completion Dialogue System Combined with task-completion and Deep Reinforcement Learning, Tiancheng at CMU (Zhao & Eskenazi, 2016) built a model for *20Q* game, where the agent ask binary questions to the user, get answers *yes/no/unknown* and then guesses a famous person in knowledge base. A correct guess results in a positive reward, while incorrect guess results in a negative reward. An LSTM is used to track conversation states and capture essential information in latent dialogue states.

Based on the previous work above, we propose a more general task-based conversational agent trained with deep *RL*, which can be applied to multiple domains with natural language generation and understanding.

2.1. Transfer learning

Many users of deep neural networks have noticed that the first layers of neural networks tend to learn similar structures for a various related tasks (Jason Yosinski, 2014). This means that the neural network is effectively a feature generation tool, which converts the raw state into a dense feature vector that can be classified with logistic regression. Transfer learning is done by first training a model from scratch. Then the weights of the model are copied to another network used for a different task. The new task may have a different number of outputs, in this case, the set of dialogue actions. Therefore, we freeze all the weights of the network and only train the final fully connected layer from scratch. Recent works has showed this approach provides good results for a number of tasks, specifically in computer vision (using ImageNet features on Flowers102). Mo et al (Kaixiang Mo, 2017) applied transfer learning to dialogue systems, but focused on personalization rather than transferring between tasks.

Essentially, the NLG module maps structured state and action data into natural language, and the NLU module maps it back. Then, the state and action pairs are used for reinforcement learning on the rewards of whether or not the task was successfully and efficiently completed.

3. Approach

3.1. Retrieving User Goal

We begin with a collection of websites containing HTML forms to represent each user goal. For example, when signing up for a new account online, websites will require user details such as first name, last name, email, job title, company size, etc. Then we take each website and scrape its HTML contents, and parse each of its forms to generate a goal description in JSON, which will be used to drive user simulator.

To more naturally emulate a user, we need example values for each field in the form that the user simulator can include in its dialog. Hence, we generate synthetic data for each data field parsed from the form using open source tools such as `faker.js` (Squires, 2017). These generated values for the fields represent constraints that the user agent abides by during dialogue generation.

3.2. User Simulation

Now, given a suitable goal description, we train a user simulation with the extracted JSON goal (see Figure 3). The user goal contains `inform slots`, which are our generated values that the user simulator tells the agent, and `request slots`, which are slots that the user does not know the values of and wants to learn from the agent throughout the conversation.

Throughout the simulation, the user maintains a stack-like representation of constraints C and the current request R. At each time step t, the user simulator will generate the next action given its current status and agent action in order to update its current state. We use the open source implementation in `TC-Bot` (Li, 2017) for the user simulation, along with its NLU and *NLG* subsystems, and train this implementation given the user goals generated from the HTML forms in the previous subsection.

3.3. RL Dialogue Agent

The reinforcement learning dialogue agent has conversations with the user simulator to try to help the user simulator complete its goal. The goal is considered completed when the agent answers all the request slots of the user. We hope to see that the RL dialogue agent will help us generalize the conversation model to new actions and fields for arbitrary tasks generated from HTML forms. To do so, we will use Word2Vec embeddings on the dialogue examples to help generalize training across different user goals and request domains. To best represent the vast state and action space, we will train a more complex model with a DNN using Tensorflow.

3.4. State Representation

The original state representation as used by (Li et al., 2016b) uses a vector to represents the state of the conversation that includes the **user action**, **user inform slots**, **user request slots**, **agent action**, **agent inform slot**, **agent inform slot**, **slots already filled**, **turn count**, and some information about the knowledge base being used. The slot information is stored as a one hot encoding. This proved to be problematic because the slots are different for different tasks and therefore the state shapes would be different and more importantly, not semantically equivalent. To unify the state representation so that it can be generalized across different tasks, we use word vectors to represent the slots that are referenced in the state space. This way the dimensions are constant across different tasks. More importantly, the word vectors should capture the semantic meaning of each slot and hopefully be able to translate that between tasks. We use word vectors from pretrained *Glove* presented by (Jeffrey Pennington, 2014).

To help the reader get a better idea of how state is represented for a dialogue, a sample representation in JSON format is presented below.

```
{
  "request_slots": {
    "ticket": "UNK"
    "theater": "UNK"
    "starttime": "UNK"
  },
  "diaact": "request",
  "inform_slots": {
    "ticket_count": "3",
    "date":
    "2017-05-23T05:13:10.339-00:00",
    "movie": "the incredibles"
  }
}
```

Figure 3. Example User Goal Description

3.5. Transfer learning

We train a dialogue agent that uses a DQN model to learn a good mapping from the state described above to the next dialogue action. We attempt to transfer knowledge learned in the model for one task to another. We do this by first training a model from scratch on the task for which we have more and higher quality data. We then load the model for this task into the DQN model for a new task. Note that we must omit the final layer, because this layer is different for each task, because the number of actions for a task can vary. Finally we retrain the model that is initialized with the weights from the other model.

There are two ways to train the model. We can either only train the final layer, which will effectively learn the mapping from a dense representation of the state learned in the first model and the dialogue actions for the new task. The second approach is to apply gradients to all the parameters in the model but use a smaller learning rate for the pretrained weights so keep them from moving to far from the pretrained values. We show the comparative results of applying both types of transfer learning and transfer knowledge in both directions between the two tasks that we evaluate.

4. Evaluation

The reward function for each conversation is determined by whether or not all of the user's request slots have been filled. Thus, our first evaluation result will measure our model's performance on a single task, say, for booking flights using the HTML forms across a variety of major

airlines.

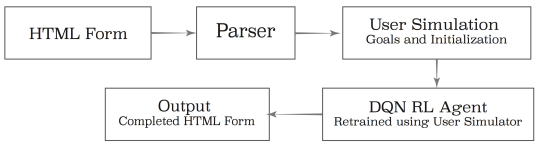

Figure 4. The end to end system.

In addition, we are interested in the transfer learning properties of our model.

Here, we will evaluate how well the RL agent does for a different task when fine tuned by a couple new user goals (by transforming HTML forms from a different domain, say a customer support portal) using user simulation.

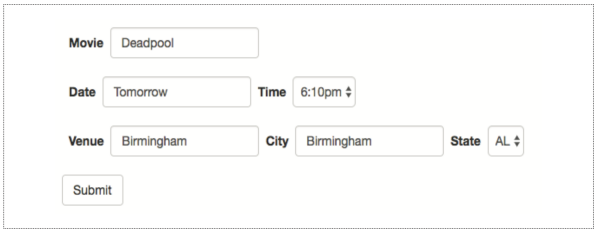

Figure 5. A sample HTML form corresponding to dialogue in Figure 8

Then, we will perform simple human evaluation to see if the trained *RL* agent sufficiently generalized the ability to transfer relevant information to a user via natural language and collect the user information to inform its own state.

5. Experiments and Results

5.1. Reinforcement Learning on Task Completion

For the task of movie ticket booking as presented in the paper (Li, 2017), we re-implemented the Deep Q-learning model in *TensorFlow* with deeper network. Original implementation in *numpy* includes one hidden layer of size 80 followed by a *ReLu*, and a fully connected layer. After fine-tuning the hyper-parameters, we were able to achieve a slightly higher success rate after 500 episodes of training, **0.82** (492 out of 600 episodes) compared to original **0.80** as seen in Figure 7 and Figure 6.

Example dialogue generated for movie booking task is shown here in Figure 8.

5.2. Transfer Learning

Now, we aim to use our trained model for movie ticket bookings on a different task – flight ticket booking .

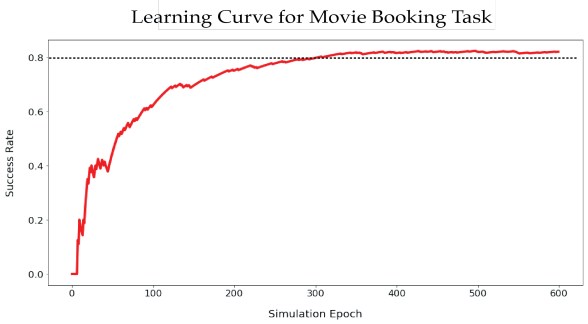

Figure 6. Learning curve on the movie booking task using our deeper *Tensorflow* Model

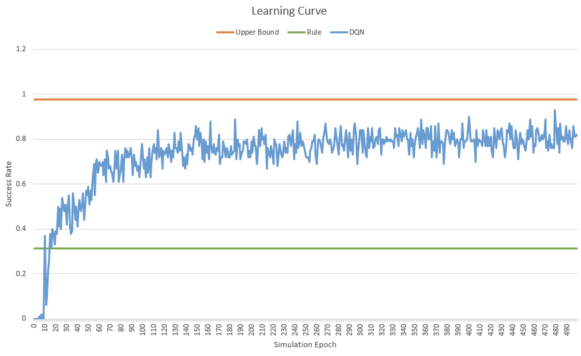

Figure 7. Benchmark: learning curve on the movie booking task using the initial *numpy* model

5.2.1. The Flight Booking Task

To create a new conversational agent for a new task, we need

- A **new knowledge base** for the user simulator and agent to work from,
- A **new set of user goals**,
- A **new set user and agent dialogue pairs** to enable simulated conversations between user simulator and the agent.

We focus on the new task of booking flights from `https://www.studentuniverse.com/`. The HTML form for flight booking has the following fields: *origin, destination, departure date, arrival date, number of travelers*.

We randomly generate a new set of 1000 entries in the knowledge base using a random airport code as both the origin and the destination, a random int between 0 and 2 for the departure date, and a random int between 3 and 5 for the arrival date. The number of travelers is randomly chosen between 1 and 4.

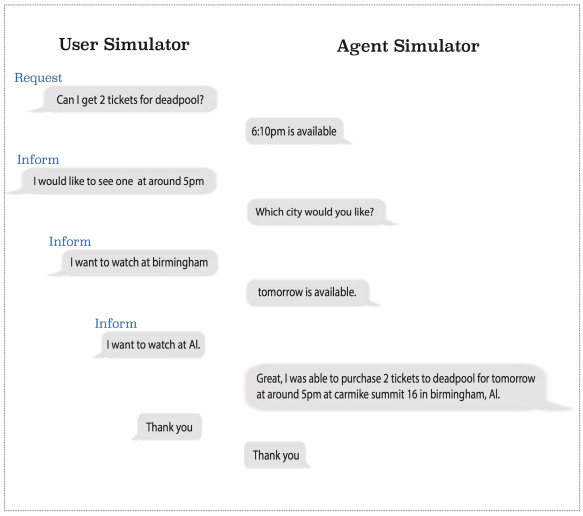

Figure 8. Example Dialogue Generated for Moving Booking Task

Then, we randomly sample 128 entries from the knowledge base and generate a goal corresponding to that entry. As shown in Figure 3, each goal has 3 fields, the `dialogue act`, the `request slots`, and the `inform slots`. For the purposes of our user simulator for flight bookings, we only create request dialogue acts. Then, we randomly choose some slots to be request slots, while the rest are inform slots matching the values from the original knowledge base entry.

Finally, we hand-write a set of dialogue act pairs. For each combination of inform slots and request slots, we write a corresponding natural language template describing the interaction. Figure 10 shows some example dialogue templates.

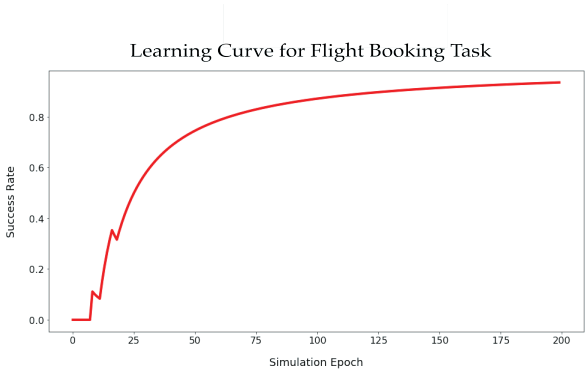

Figure 9. Learning curve for the flight task (without transfer learning).

```
{
  "dia_acts": {
    "inform": [
      {
        "request_slots": [],
        "nl": {
          "agt":
          "$origin1$ is available.",
          "usr":
          "I want to go to $origin1$."
        },
        "inform_slots": [
          "origin1"
        ]
      }
    ],
    "request": [
      {
        "nl": {
          "usr":
          "What's your departure date?",
          "agt": "When will you depart?"
        },
        "request_slots": [
          "flightDate1"
        ],
        "inform_slots": [
        ]
      }
    ]
  }
}
```

Figure 10. A single example dialogue template of an `inform` dialogue pair and a `request` dialogue pair.

5.2.2. Training on New Task: Flight Booking

Once we've created a new knowledge base, goal set, and dialogue pairs for the flight task, we train it without transfer learning to get a sense of how the model trains. We use the same Deep Q-learning architecture from the movie booking task and to produce the graph seen in Figure 9. We see that the flights task is substantially easier than the movies task, as the agent starts to consistently ace the task after the 20th epoch.

5.2.3. Transfer Learning for Flight Booking

After we have a well-trained model for task completion on movie bookings, we experimented with transfer learning by applying our model on another task: booking flights. We retained all the previous layers in the *TensorFlow* model and only re-trained the final fully connected layer. The basic assumption here is that even though the task is different—i.e. final output is different—the agent should maintain a similar interpretation of state representation across different tasks.

However, one unexpected problem occurred: our flight task has a different state representation from the original one due to different slot settings and natural language pairs. The original state representation for movie booking is a 218 dimensional vector, while the state representation for our flight task is a 118 dimensional vector, thus making the trained model not compatible for the new task since we retain the first layer. This is because the input state is formed by concatenating one-hot vectors of various model inputs, including the slot terms, and dialogue acts, which vary in number from task to task.

We experimented with two ideas to approach the problem:

- **Extending State Representation to Same Dimension**: after we extract the original state representation, we extend it to a 300 dimensional vector by padding with 0s. The idea is that by using the same one hot encodings across tasks, meaningful state representations will be preserved and our model may can simply ignore the paddings, which will always be 0 during the learning phase.

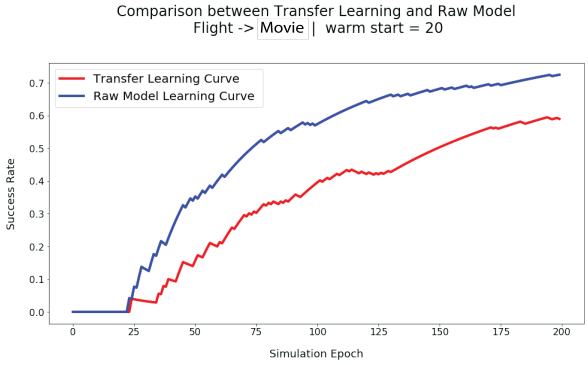

Figure 11. Transfer Learning: flight task to movie task

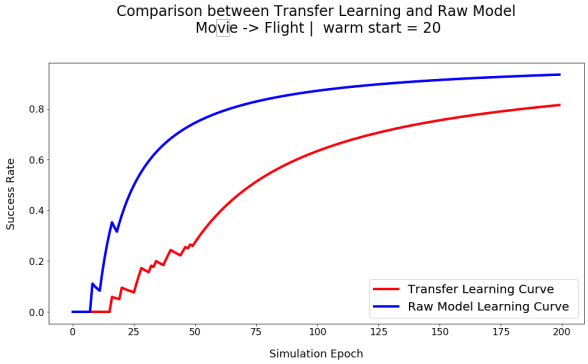

Figure 12. Transfer Learning: movie task to flight task

Unfortunately, even though the model with transplanted weights continued to learn in the extended state representation, transfer learning did not perform as well as expected. The agent with pre-trained model for transfer learning was consistently outperformed by the model without usage of transfer learning.

We tried transfer learning in both directions: transferring the trained model from the movie task to the flight task, and transferring the trained model from the flight task to the movie task, as shown in Figure 11 and Figure 12.

We think our transfer learning models perform worse than the original models because the flight task and movie task are fundamentally different in complexity, and also have different slots to fill and thus different representations for the slots in the dialogue state.Hence, the weights in the model were tuned for a problem of a different level of complexity and with significantly different state representations. As our transfer learning model only retrained the final layer, this made the power of the final layer to model our Q values much more limited, and hence our success rate suffers in comparison to the case where we train *all* the model weights directly as in Figure 6 and Figure 9.

- **Representing Dialogue State using Word Embeddings**: The agent's dialogue state is based on a variety of parameters, including the set of all possible slots and dialogue acts to be filled during the task. In the original model, these terms are all represented as one-hot vectors. As this caused problems in the dimensionality of our dialogue state from the flight task to the movie task, we also tried representing those input words in their word embedding space, by averaging the *GloVe* vectors each of the dialogue slot terms (Pennington et al.). In this representation, the state representations for any task maps to the same dimension, while the semantic meaning of the dialogue state is still maintained.

  Unfortunately, this approach did not work: the agent on the trained model started collecting the minimum possible reward throughout training and failed to learn after 100 epochs. Interestingly, when running the model without transfer learning while using this state representation, the model did train. One potential reason it failed in the transfer learning setting was that the agent trained on one language setting had weights too specific to that set of word vectors, and failed to adjust to another set of word vectors.

6. Conclusion

We demonstrated improvements to the model proposed in (Li et al., 2016b) by using a deeper *DQN*, improving its task

completion accuracy from a 0.80 success rate to a 0.82 success rate. However, we were unsuccessful in our efforts to demonstrate further enhancements to this model's robustness through transfer learning in the experiments we ran so far.

One of the bottlenecks was the comprehensiveness of the knowledge base, goal, and dialog data we needed to generate. In particular, the dialogue templates are excruciatingly long, as a dialogue must be created for each combinatorial configuration of input slots and request slots. As a result, our dialogue template file has over 2000 lines of dialogue data.

While creating the dialogue data is labor intensive, we believe the most difficult part about applying transfer learning to this task is in how to unify the input state representations across different tasks to enable transfer learning, as this will train the model weights to more reasonable settings for task transfer. The subject of our future research directions chiefly involves devising new methods of input state representation that can be unified across tasks. The main challenge is that different tasks have different input dimensions, so our state representation strategy needs to find a smart way to retain the dialogue state regardless of how many slots the system needs to fill. Two potential methods we can think of are to map similar dialog slots to each other in the embedding space to increase the transferability of the model's trained parameters, either by hand, or using a second model to automatically make that mapping during training of the second model. We think that these efforts may yield much stronger and exciting results for transfer learning of our task-completion dialogue agent.

7. Acknowledgements

We would like to thank *Prof.* Emma Brunskill for instructing this course and giving us the opportunity to have hands-on experience with Reinforcement Learning.

8. Code Repository

Our code is hosted at https://github.com/dengl11/CS234-Project, with constant and equal contributions from our three team members.

References

Jason Yosinski, Jeff Clune, Yoshua Bengio Hod Lipson. How transferable are features in deep neural networks? *NIPS; arXiv:1411.1792*, 2014.

Jeffrey Pennington, Richard Socher, Christopher D. Manning. Glove: Global vectors for word representation. *EMNLP*, 2014.

Kaixiang Mo, Shuangyin Li, Yu Zhang Jiajun Li Qiang Yang. Personalizing a dialogue system with transfer reinforcement learning. *arXiv:1610.02891*, 2017.

Li, Jiwei, Monroe, Will, Ritter, Alan, Galley, Michel, Gao, Jianfeng, and Jurafsky, Dan. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016a.

Li, Xiujun. Tc-bot. https://github.com/MiuLab/TC-Bot, 2017.

Li, Xiujun, Lipton, Zachary C, Dhingra, Bhuwan, Li, Lihong, Gao, Jianfeng, and Chen, Yun-Nung. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*, 2016b.

Li, Xuijun, Chen, Yun-Nung, Li, Lihong, and Gao, Jianfeng. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*, 2017.

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

Milica Gasic, Nikola Mrksic, Lina M. Rojas-Barahona Pei-Hao Su Stefan Ultes David Vandyke Tsung-Hsien Wen and Young, Steve. Dialogue manager domain adaptation using gaussian process reinforcement learning. *Computer Speech and Language preprint*, 2017.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. Glove: Global vectors for word representation.

Squires, Marak. faker.js. https://github.com/Marak/faker.js, 2017.

Zhao, Tiancheng and Eskenazi, Maxine. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*, 2016.