Computer Communications
and Applications
Laboratory 3 (LCA3)

Artificial Intelligence &
Machine Learning Group

Master Thesis

Building Advanced Dialogue Managers for Goal-Oriented Dialogue Systems

Vladimir Ilievski

Supervisors

EPFL:	Prof. Patrick Thiran
Swisscom:	Dr. Claudiu Musat

March 16, 2018

Abstract

Goal-Oriented (GO) Dialogue Systems, colloquially known as goal oriented chatbots, help users achieve a predefined goal (e.g. book a movie ticket) within a closed domain. A first step is to understand the user's goal by using natural language understanding techniques. Once the goal is known, the bot must manage a dialogue to achieve that goal, which is conducted with respect to a learnt policy. The success of the dialogue system depends on the quality of the policy, which is in turn reliant on the availability of high-quality training data for the policy learning method, for instance Deep Reinforcement Learning.

Due to the domain specificity, the amount of available data is typically too low to allow the training of good dialogue policies. In this master thesis we introduce a transfer learning method to mitigate the effects of the low in-domain data availability. Our transfer learning based approach improves the bot's success rate by 20% in relative terms for distant domains and we more than double it for close domains, compared to the model without transfer learning. Moreover, the transfer learning chatbots learn the policy up to 5 to 10 times faster. Finally, as the transfer learning approach is complementary to additional processing such as warm-starting, we show that their joint application gives the best outcomes.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor at EPFL, prof. Patrick Thiran for sharing his valuable ideas and insights with me and the team over the course of this master thesis.

I would also like to thank my supervisor at Swisscom, Dr. Claudiu Musat for motivating, leading and supporting me during my work on this master thesis.

I would also like to express my sincere gratitude to my family - my parents and my sister - for supporting me throughout my studies and my life in general. I could not have imagined achieving this without their support.

Last but not least, I would like to thank prof. Joseph Sifakis and Dr. Simon Bliduze for giving me an opportunity to work part time as a Research Scholar at the Rigorous System Design Lab (RiSD), and support myself during the studies at EPFL.

Lausanne, 16 March 2018 Vladimir Ilievski

Contents

List of Figures

1 Introduction

Today, we live in the era of Artificial Intelligence (AI), which is penetrating in every aspect of our life. Part of this AI ecosystem are the spoken and text-based Dialogue Systems, which usage is constantly growing. These systems are quite popular because they have a potential to convey a conversation, just like a real human, acting in a truly intelligent manner.

To increase the hype, the Loebner prize [Epstein, 1992] is a competition for text-based Dialogue Systems inspired by the Turing's imitation game. The aim is to stimulate and motivate the creation of truly intelligent conversational machines. Despite this, the competition has not had a winner in all previous editions. According to [Kurzweil, 2010] it is only a matter of time when that will happen.

Moreover, with the increasing pervasiveness of smart phones and ubiquitous computer systems, the Dialogue Systems are getting even more attractive. Consequently, they started being used in a plethora of different applications, ranging from trivial chit-chatting to personal assistants. An example of such systems are the popular Apple's Siri, Google Now and Cortana from Microsoft [Strayer et al., 2017]. Therefore, it is of paramount importance to continue the development of these systems and push the boundaries even further.

The text-based Dialogue Systems colloquially known as Chatbots, are divided in two groups, depending on the nature of the conversation. In fact, there are: *i*) open-domain and *ii*) closed-domain Chatbots.

In the open-domain setting the conversation can go in any direction, which means the users can have an open conversation with the Chatbot about everything, usually in the format of chit-chatting, without any or minimal functionality. For instance, [Serban et al., 2016] used the Movie-DiC [Banchs, 2012] corpus of movie dialogues

to build a general-purpose Chatbot. Because of the general-coverage nature, the open-domain Chatbots require huge amount of annotated data, thus making it almost impossible to create one.

On the other hand, the closed-domain Dialogue Systems are more practical and easier to implement, because they focus only on few aspects and are designed to help users to achieve predetermined goals in a predefined domains. For example, it could be a travel planning task [Peng et al., 2017] or restaurant table booking dialogue system [Wen et al., 2016b], to help users book a flight or table in restaurant, in the most convenient way, i.e. by conversation. For this reason they are called Goal-Oriented (GO) Chatbots and can be grouped together in larger systems such as Amazon Alexa[1] to give an impression of a general coverage. Each individual component (which in Amazon Alexa can be viewed as skills of the overarching generalist bot) is closed-domain in nature.

1.1 Contributions and Thesis Outline

This thesis focuses only on a subset of the Goal-Oriented Chatbots, modeled as Partially Observable Markov Decision Processes (POMDP) [Young et al., 2013], where the rich set of Reinforcement Learning (RL) algorithms [Sutton and Barto, 1998] can be used to train them, for instance the Deep Q-Nets (DQN) [Mnih et al., 2015]. The lack of in-domain dialogue data is a key problem for training high quality RL-based Goal-Oriented Chatbots. We need in-domain labeled dialogues for two reasons: *i*) to warm-start the Chatbot, which is a standard widely used technique and *ii*) to train the chatbot by simulating a considerable number of different conversations.

In this thesis we argue that the domain similarity can be leveraged in a clever way to build efficient GO Chatbots with less data, using the so-called *Transfer Learning* technique. We use the similarity between a *source* and a *target* domain, as many domains, such as restaurant and movie booking, share to a large extent some common information. For example, in the restaurant booking scenario, the user might ask the question *"Which restaurant I can book a table for 3 people for tomorrow?"*, while in the movie booking domain the question could be *"Which theater I can book 3 tickets for tomorrow?"*. In both domains, the user includes information for number of people and time. We believe this information need not be learnt twice and that a transfer is possible.

[1] https://developer.amazon.com/alexa

We successfully combine *Transfer Learning* and RL-based Goal-Oriented Chatbots and to the best of our knowledge we are the first ones doing that. As a result of the research over the course of this thesis, we published a paper [Ilievski et al., 2018], which is submitted for a review at the 27th International Joint Conference on Artificial Intelligence (IJCAI). The contributions of this thesis are following:

- **Training GO Chatbots with less data**: In data constrained environments, models trained with *Transfer Learning* achieve better training and testing performances than ones trained independently.
- **Better GO Chatbot performance**: Using *Transfer Learning* has a significant positive effect on performance even when all the data from the target domain is available.
- **Intuitions on further improvements**: We show the gains obtained with *Transfer Learning* are complementary to the ones due to warm-starting and the two can be successfully combined.
- **New published datasets**: We publish new datasets for training Goal-Oriented Dialogue Systems, for restaurant booking and tourist info domains[2]. They are derived from the third Dialogue State Tracking Challenge [Henderson et al., 2013].

After the Related Work Chapter (Chapter 2), we first make a general overview of the Dialogue Systems in Chapter 3. Then, in Chapter 4, we describe the model of the RL-based GO Chatbots, which performance relies on a robust dialogue state tracking and an efficient learnt policy, as described in Chapter 5. We conduct our experiments and show the results in Chapter 6. Finally in Chapter 7 we conclude our work and present the possible future work.

[2]The datasets will be published in the camera-ready version

2 Related Work

2.1 Goal-Oriented (GO) Dialogue Systems

The Goal-Oriented (GO) Dialogue Systems have been under development in the past two decades, starting from the basic, handcrafted Dialogue Systems. For instance, [Larsson and Traum, 2000] introduced a framework for dialogue management development, based on hand-crafted rules. In the same direction, [Zue et al., 2000] built a weather information, hand-crafted Goal-Oriented Chatbot. The recent efforts to build such systems are generally divided in three lines of research.

2.1.1 Fully-Supervised Models

The first way is to treat them in an end-to-end, fully supervised sequence-to-sequence manner [Sutskever et al., 2014]. Thus, we can use the power of the deep neural networks based on the encoder-decoder principle to infer the latent representation of the dialogue state. However, it is worth noting that these models require considerable amount of data.

The authors in [Vinyals and Le, 2015] used standard Recurrent Neural Networks (RNNs) and trained a Goal-Oriented Chatbot in a straightforward sequence-to sequence fashion. They benchmarked their findings in IT helpdesk troubleshooting domain, where costumers face computer related issues, and a specialist help them by conversing and walking through a solution. Due to the incapability of the recurrent nets to compress very long dependencies, this chatbot is not having a strong reasoning power.

To overcome the RNN memory limitations, in their work [Bordes and Weston, 2016]

used LSTM cells in combination with explicit memory, known as Memory Networks [Sukhbaatar et al., 2015], to build a Goal-Oriented Chatbot, using the bAbI[1] tasks for restaurant reservation. The chatbot demonstrated better reasoning power, thus remembering and updating the past user preferences. For this reason, this work represents a testbed for testing the shortcomings and strengths of fully-supervised end-to-end Goal-Oriented Dialogue Systems.

2.1.2 Reinforcement Learning-based Models

Another branch of research had emerged, focusing on the Deep Reinforcement Learning because the fully-supervised approach is data-intensive. These models have a quite complex structure, since they include many submodules, such as Natural Language Understanding (NLU) [Hakkani-Tür et al., 2016] and Natural Language Generation (NLG) [Wen et al., 2015] units, as well as a Dialogue State Tracker (DST).

Aligned in this direction, [Cuayáhuitl, 2017] created a simple Dialogue System for a restaurant reservation domain. The system's actions solely depend on the RL-based agent, by performing action selection directly from raw text of the last user and system responses instead of manual feature engineering. Therefore, this system does not include any language understanding and state tracking units, which in turn is quite constraining. Another simple Question-Answering Chatbot (Q&A bot) is presented in [Dhingra et al., 2016]. It is an end-to-end Reinforcement Learning Chatbot, which helps users search Knowledge Bases (KBs) for movies, without composing complicated queries.

As an extension of all the previous work, [Li et al., 2017] went one step further and built a comprehensive Movie Booking Chatbot. It includes a User Simulator [Li et al., 2016], which simulates the user in the training process, Natural Language Understanding (NLU) and Natural Language Generation (NLG) units, as well as a basic Dialogue State Tracker and a Policy Learning module. It is trained in an end-to-end fashion, by leveraging the Deep Q-Nets (DQN) [Mnih et al., 2015] for the policy learning.

[1] https://research.fb.com/downloads/babi/

2.1.3 Hybrid Models

This line of research combines both, the Fully-Supervised and Reinforcement Learning-based approaches, in order to escape the limitations, characteristic for both of them.

In their work [Su et al., 2016] described a two-step approach to train a policy for Goal-Oriented Chatbot. In the first step, the algorithm is trained on a fixed corpus data in a supervised way. In the second step, the policy is fine-tuned using RL-based policy gradient [Williams, 1992] technique, in order to explore the dialogue space more efficiently.

Similarly, [Williams et al., 2017], proposed a solution to train a Goal-Oriented Dialogue System in two modes: off-line and on-line mode. In the off-line mode, the system is trained in a fully-supervise manner, combining an LSTM network with hand-crafted templates, to mitigate the data requirements. Afterwards, in the on-line mode, the system learns autonomously by incorporating RL-based policy gradient approach.

2.2 Data-Constrained Dialogue Systems

One desired property of the Goal-Oriented Chatbots is the ability to switch to new domains and at the same time not to lose any knowledge learned from training on the previous ones. This property is enforced due to the lack of in-domain data required to train high-quality Goal-Oriented Chatbots.

In this direction, the authors in [Gašić et al., 2015] proposed a Gaussian Process-based technique to learn generic dialogue polices, which are organized in a class hierarchy. These policies with a modest amount of data can be furthermore adjusted according to the use case of the dialogue system.

On the other hand, [Wang et al., 2015] learned domain-independent dialogue policies, such that they parametrized the ontology of the domains. In this way, they show that the policy optimized for a restaurant search domain can be successfully deployed to a lap-top sale domain.

Last but not the least, [Lee, 2017] utilized a continual learning technique, to smoothly add new knowledge in the neural networks that specialized a dialogue policy in an end-to-end fully-supervised manner.

Nevertheless, none of the previously mentioned papers tackles the problem of transferring the domain knowledge in case when the dialogue policy is optimized using a Deep Reinforcement Learning. In this thesis, we propose such method, based on the standard Transfer Learning technique [Pan and Yang, 2010]. Therefore, using this method we surpass the limitations to transfer the in-domain knowledge in Goal-Oriented Dialogue Systems based on Deep Reinforcement Learning.

3 Overview of Dialogue Systems

This chapter gives an overview of the Dialogue Systems in general. The research on the dialogue systems intents to create comprehensive systems that can hold a real conversation, successfully employing all aspects: reasoning power, giving well defined and reasonable responses, emotion detection etc. Depending on the nature of the input and output, there are two types of Dialogue Systems: *i*) Spoken Dialogue Systems (SDS) and *ii*) Text-Based Dialogue Systems colloquially known as Chatbots. Fundamentally, both types exhibit many similarities, only the pre- and post-processing techniques differ.

There is no consensus on the architecture of the Dialogue Systems, it is case-dependent. In general they are always composed of two parts: *i*) user, whether real or simulated and *ii*) the internal system. One good reference is given in [Pieraccini and Huerta, 2005]. In any case, both parts converse in alternating manner such that a *dialogue turn* is one cycle of consecutive utterances from the user and the system.

3.1 Spoken Dialogue Systems

The *Spoken Dialogue Systems* are specially designed for environments which does not include user interfaces such as big screens and keyboards, they rather use microphone and speakers. [Henderson, 2015c] shows a typical composition of the Spoken Dialogue Systems as well as the information flow. For this reason, the input and output is continuous speech signal, which requires special modules and techniques to handle (see Figure 3.1), which includes:

- The *Automatic Speech Recognition* (ASR) unit [Zhang et al., 2017] assigns

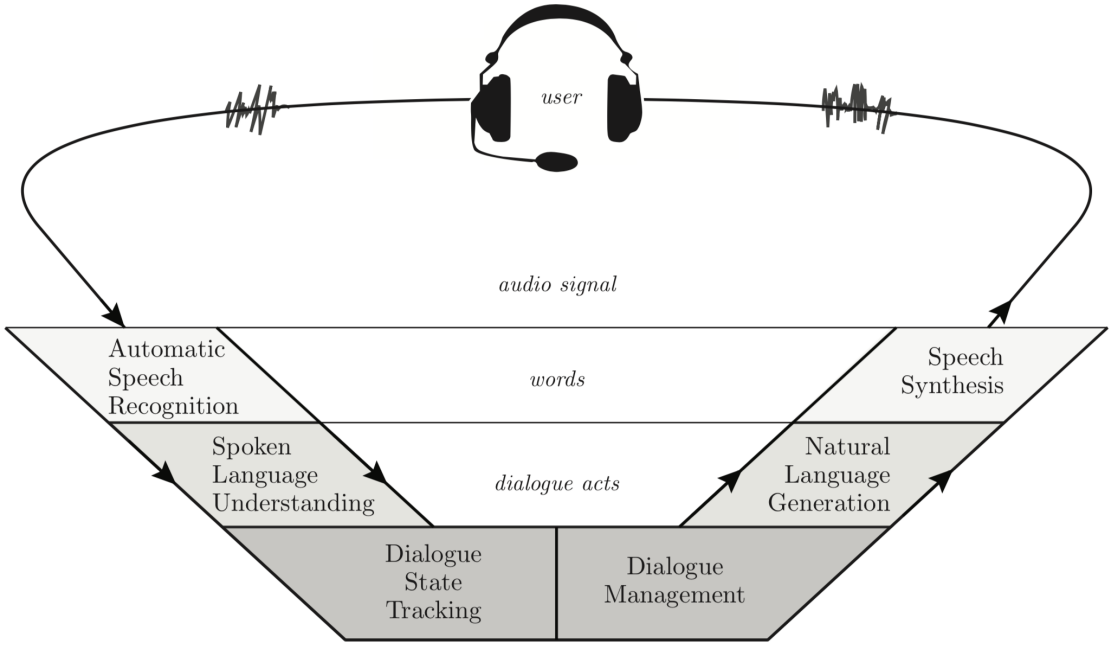


Figure 3.1 – The architecture of a Spoken Dialogue System from [Henderson, 2015c]

probabilities to the words in the user utterance.

- The *Spoken Language Understanding* (SLU) unit [Yao et al., 2014] infers the semantics of the user input.
- The *Speech Synthesis* (SS) unit [Zen et al., 2009], will convert the system's response into speech.

3.2 Text-Based Dialogue Systems (Chatbots)

Contrary to the Spoken Dialogue Systems, the Text-Based Systems (known as Chatbots) only focus on the interfaces which include screens and keyboards, meaning the input and output is text. Consequently, they include appropriate units and algorithms to handle text. Depending on the nature of the conversation, the Chatbots are divided in Open-Domain and Closed-Domain Chatbots known as Goal-Oriented (GO) Chatbots.

In the open-domain setting, the conversation can go in any direction, usually in the form of chit-chatting, without any purpose. Because of their nature to cover every possible case, it is almost impossible to create a perfect open-domain Chatbot, as shown in [Serban et al., 2016]. Thus, most of the chatbot research is on the

closed-domain Chatbots, which is a case in this thesis.

3.3 Goal-Oriented (GO) Chatbots

The Goal-Oriented (GO) Dialogue Systems are more useful and practical and are easier to implement. This is due to the fact that their domain of expertise is much narrower, focusing only on few key points of the dialogue. In general there are two dominant paradigms in GO Dialogue Systems implementations: *i*) Fully-Supervised and *ii*) Reinforcement Learning (RL) based.

3.3.1 Fully-Supervised GO Chatbots

In case of fully supervised implementation, we apply the recurrent neural networks (RNNs) encoder-decoder principles, mainly applied in the machine translation. An example of this kind of models is presented in [Bordes and Weston, 2016, Wen et al., 2016b]. These models are trained in a sequence-to-sequence fashion [Sutskever et al., 2014], that encode the user request and its context and decode the bot answer directly.

The fully supervised GO chatbots require a considerable amount of annotated human-human or human-machine dialogues since the system is trying to mimic the knowledge of the expert. Moreover, we don't have a control over the internal state, which means we can not model the dialogue as we wish.

3.3.2 Reinforcement Learning (RL) based GO Chatbots

On the other hand, we can model the GO Chatbots as a Partially Observable Markov Decision Process (POMDP)[Young et al., 2013]. The Reinforcement Learning (RL) [Sutton and Barto, 1998] algorithms are one rich subset of powerful and promising algorithms that can be applied in this case.

Figure 3.2 shows a typical composition of an RL-based GO Chatbot, as well as the information flow. Following the pipeline, there are three separate components, each having a specific role in the process:

1. First of all the *Natural Language Understanding* unit [Hakkani-Tür et al., 2016] will infer the semantics of the user input. This includes understanding the user intent and the slots (i.e. the relevant information).

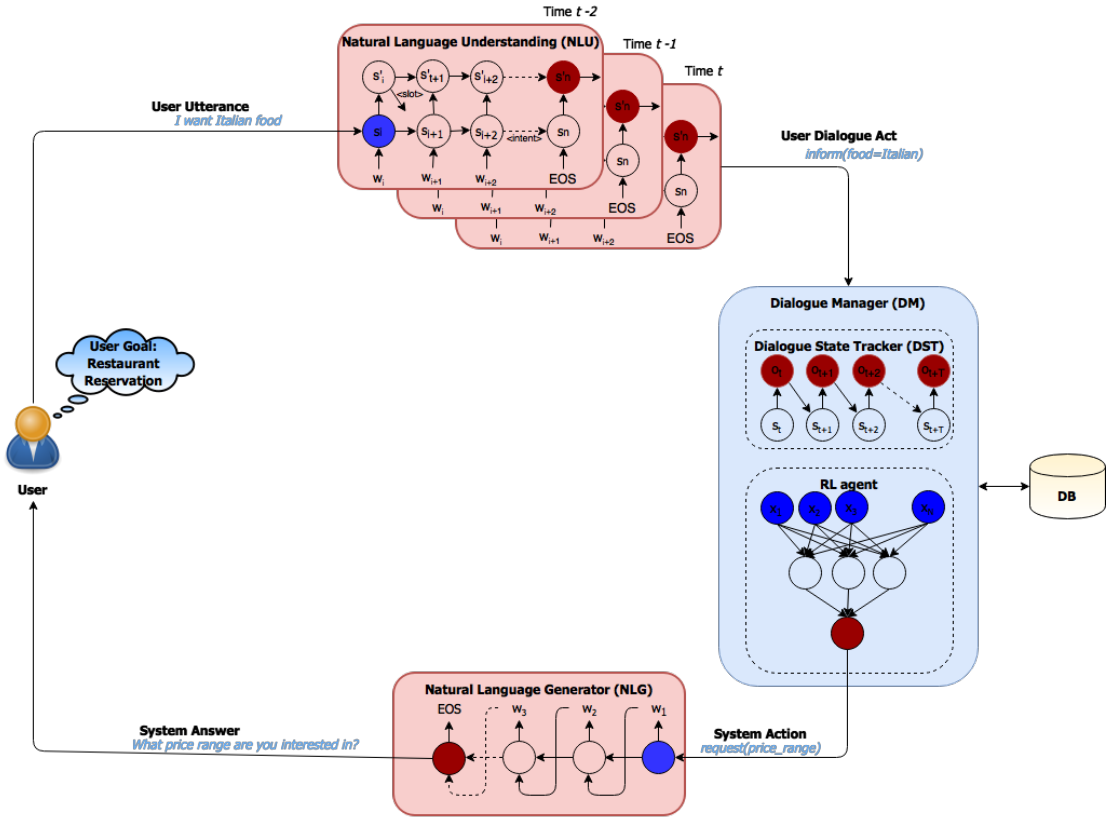


Figure 3.2 – Text-Based Dialogue System modeled as a Partially Observable Markov Decision Process. The user utterance is parsed by the NLU unit producing a dialogue act understandable for the system. In the Dialogue Manager, the state tracker is estimating the state such that the RL agent could take the ideal action. This action is further passed to the NLG unit and finally presented to the user in a human readable form.

2. In the second step, the *Dialogue Manager* (DM) will take care of the dialogue. Based on the context, previous user and system actions it will produce the next system action. Usually it includes two other subcomponents: *i*) *Dialogue State Tracker* (DST) [Henderson, 2015a], which purpose is to build reliable state of the dialogue and a *Policy Learning* module which reads the dialogue states and takes the next system action.

3. Finally, the *Natural Language Generation* unit [Wen et al., 2015], based on the DM output, will generate a natural sentence, understandable for the end user.

RL-based Chatbots require less annotated dialogues than their sequence-to-sequence counterparts, due to their ability to simulate the conversation, thus exploring the

unknown dialogue space more efficiently. The data requirements are however not trivial and obtaining the dialogue data is still the biggest obstacle their creators face. In the following chapters we dive deeper in the RL-based GO Chatbots and show how this obstacles can be surpassed using *Transfer Learning*.

4 Model of RL-based GO Chatbots

If we model the GO Dialogue Systems as POMDP and apply the Reinforcement Learning techniques as in [Zhao and Eskenazi, 2016, Li et al., 2017], then, several components are compromising that system as shown in Figure 3.2. It consists of two independent units: the *User Simulator* on the left side and the *Dialogue Manager* (DM) on the right side. In-between there are the Natural Language Understanding (NLU) and Natural Language Generator (NLG) units. Our work is based on the model from [Li et al., 2017], who proposed an end-to-end reinforcement learning approach to build a Goal-Oriented Chatbot in a movie booking domain.

Goal-Oriented bots contain an initial NLU component, that is tasked with determining the user's *intent* (e.g. *book a movie ticket*) and its parameters, also known as *slots* (e.g. date: *today*, count: *three people*, time: *7 pm*). The usual practice in the RL-based Goal-Oriented Chatbots is to define the user-bot interactions as *semantic frames*. At some point $\mathbf{t}$ in the time, given the user utterance $\mathbf{u_t}$, the system needs to perform an action $\mathbf{a_t}$. A bot action is, for instance, to request a value for an empty slot or to give the final result.

The entire dialogue can be reduced to a set of *slot-value* pairs, called *semantic frames*. Consequently, the conversation can be executed on two distinct levels:

1. **Semantic level:** the user sends and receives only semantic frames as messages.
2. **Natural language level:** the user sends and receives natural language sentences, which are reduced to, or derived from a semantic frame by using Natural Language Understanding (NLU) and Natural Language Generation (NLG) units respectively [Wen et al., 2015, Hakkani-Tür et al., 2016].

For instance, in the movie booking domain one semantic frame could be defined as:

{movie_name: "Titanic", number_of_people: "2", theater, intent="request"},

which in natural language could be written as:

"Which theater I can book 2 tickets for the movie Titanic?"

By exchanging this kind of a data structures, the user and the system can convey the entire conversation until reaching the goal.

4.1 User Simulator

The *User Simulator* creates a user - bot conversation, given the semantic frames. Because the model is based on Reinforcement Learning, a dialogue simulation is necessary to successfully train the model [Li et al., 2016].

From the dataset of available user goals the User Simulator randomly picks one, which is unknown for the Dialogue Manager. The user goal consists of two different sets of slots: *inform slots* and *request slots*.

- *Inform slots* are the slots for which the user knows the value, i.e. they represent the user constraints (e.g. {movie_name: "avengers", number_of_people: "3", date: "tomorrow"}).
- *Request slots* are ones for which the user is looking for an answer (e.g. { city, theater, start_time }).

Having the user goal as an anchor, the user simulator generates the *user utterances* $\mathbf{u_t}$. The initial user utterance, similar to the user goal, consists of the initial inform and request sets of slots. Additionally, it includes a user intent, like *open dialogue* or *request additional info*.

The user utterances generated over the course of the conversation follow an agenda-based model [Schatzmann and Young, 2009]. According to this model, the user is having an internal state $\mathbf{s_u}$, which consists a goal G and an agenda A. The goal furthermore is split in user constraints C and user requests R. In every consecutive time step $\mathbf{t}$, the user simulator creates the user utterance $\mathbf{u_t}$, using its current state $\mathbf{s_u}$ and the last system action $\mathbf{a_t}$. In the end, using the newly generated user utterance $\mathbf{u_t}$, it updates the internal state $\mathbf{s'_u}$.

4.2 Natural Language Understanding Unit

The *NLU* unit is responsible for transforming the user utterance to a predefined *semantic frame* according to the system's conventions, i.e. to a format understandable for the system. This includes a task of slot filling and intent detection.

For example, the intent, could be a *greeting*, like *Hello*, *Hi*, *Hey*, or it could have an *inform* nature, for example *I like Indian food*, where the user is giving some additional information. Depending on the interests, the slots could be very diverse, like the *actor name*, *price*, *start time*, *destination city* etc. As we can see, the intents and the slots are defining the closed-domain nature of the Chatbot.

The task of slot filling and intent detection is seen as a sequence tagging problem. For this reason, the NLU component is usually implemented as an LSTM-based recurrent neural network with a Conditional Random Field (CRF) layer on top of it. The model presented in [Hakkani-Tür et al., 2016], is a sequence-to-sequence model using bidirectional LSTM net, which fills the slots and predicts the intent in the same time. On the other hand, the model in [Liu and Lane, 2016] is doing the same using an attention-based RNN.

To achieve such a task, the dataset labels consist of: concatenated B–I–O (Begin, Inside, Outside) slot tags, the intent tag and an additional end-of-string (EOS) tag. As an example, in a restaurant reservation scenario, given the sentence *Are there any French restaurants in Toronto downtown?*, the task is to correctly output, or fill, the following slots: *{cuisine: French}* and *{location: Toronto downtown}*. The table below shows how we would correctly tag the previous example. One very effective technique to build better NLU units with less data (based on the *Active Learning* methodologies) is presented in [Dimovski et al., 2018].

Table 4.1 – An example of tagging a sentence in B–I–O (Begin, Inside, Outside) format

Are	there	any	French	restaurants	in	Toronto	downtown?
O	O	O	B-Cuisine	O	O	B-Location	I-Location

4.3 Natural Language Generator Unit

The NLG unit, on the other hand is the glue between the system and the user. Given the system response as a *semantic frame*, it maps back to a natural language sentence, understandable for the end user. The *NLG* component can be rule-based

or model-based. In some scenarios it can be a hybrid model, i.e. combination of both.

The rule-based NLG outputs some predefined template sentences for a given *semantic frame*, thus they are very limited without any generalization power. For this reason, they are only used in special occasions.

On the other hand, the model-based NLG units, are having learnable parameters and are usually trained in a sequence-to-sequence fashion. The models presented in [Wen et al., 2016a, Wen et al., 2015], use an LSTM-decoder with a given *semantic frame*, to generate template-like sentences with slot placeholders. Afterwards, a beam searched is applied, to replace the placeholders with actual values.

4.4 Dialogue Manager

At the core of the GO Dialogue Systems lies the *Dialogue Manager* (DM), supported by the NLU and NLG units. Additionally, the DM could be connected to some external Knowledge Base (KB) or Data Base (DB), such that it can produce more meaningful answers.

The Dialogue Manager consists the following two components: the *Dialogue State Tracker* (DST) and the *Policy Learning* which is the RL agent.

The *Dialogue State Tracker* (DST) is a complex and essential component that should correctly infer the belief about the state of the dialogue, given all the history up to that turn. The *Policy Learning* is responsible for selecting the best action, i.e. the system response to the user utterance, that should lead the user towards achieving the goal in a minimal number of dialogue turns.

4.4.1 Dialogue State Tracker

The *Dialogue State Tracker* (DST) is producing a meaningful state s_t of the dialogue up to time t in the time. The state s_t is a data structure, that should depict the state of the conversation to a level of detail that provides all necessary information in order an intelligent agent to easily and reliably select the next action.

The tracker, takes all possibly observable input up to time t which includes: all the user utterances and system actions taken so far, all the results from the NLU unit. Additionally, it might include all external knowledge provided in a knowledge

base or a data base. For example in a restaurant search scenario, the state might indicate the user price range and cuisine preferences, what information they are seeking, like a telephone number or address.

Therefore, given all of this information, a robust dialogue state tracker outputs a distribution $p(s)$ over all possible dialogue states. This is due to the fact that the true state is not fully observable from the raw input. Several factors contribute for that: ambiguous or not clearly specified user utterances, the noise and the error from the *NLU*, changes in user goal etc.

In order to tackle these challenges, in the literature there are three types of state trackers: *rule-based*, *generative models* and *discriminative models*. Very recently, the series of the Dialogue State Tracking Challenge (DSTC) have started [Williams et al., 2016, Henderson, 2015b], a competition aiming to boost the state trackers to the next level. Many state-of-the-art dialogue state trackers emerged from this competitions.

Hand-Crafted DST models

The *hand-crafted* dialogue state trackers are the most basic ones and they were used in the early dialogue systems. They infer the state of the dialogue by a manually designed and tuned parameters, such that the new state s' is derived from the last state s using the last user utterance. An example of such system is the weather information system developed at MIT, called JUPITER [Zue et al., 2000]. Moreover, in [Larsson and Traum, 2000] a hand-crafted rules are used to build a complex dialogue management system.

One strong advantage of this kind of state trackers is that they do not require any training data. However, due to the lack of flexibility and inability to adapt and generalize over many possible states, a data-driven approach is required and obvious.

Generative DST models

For this reason, the *generative models* emerged, modeling the dialogue as a dynamic Bayesian network considering the state s, and the user action u as an unobservable random variables. In general, given the input vector $x \in \mathbb{R}^N$ for some $N \in \mathbb{N}$, and the label y, the generative models are trying to estimate the joint distribution of x and y, i.e. $\Pr(y, x)$. Thus, the rest of the probabilities can be derived using the Bayesian rules. The Bayesian net for inferring the new state s' is shown in Figure

4.1. The new state s', depends on the previous state s, and the current machine action a', which on the other side depends on the noisy user action $\underline{\mathbf{u}}$.

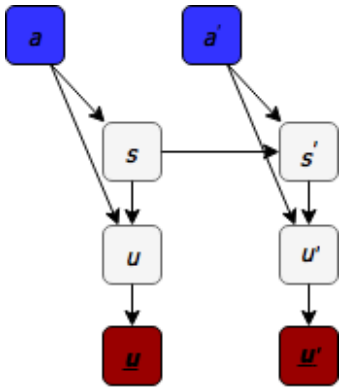


Figure 4.1 – Dynamic Bayesian Net for inferring the state of the dialogue. The state s' depends on the previous state s and the system action a'. This leads to a new user utterance u' (depending both on the state and system action), which becomes noisy afterwards.

Discriminative DST models

Finally, the *discriminative models* for state tracking are the most powerful ones. They are directly estimating the conditional probability Pr $(y|x)$, where y is the label and x is the underlying data. One example and very successful discriminative model is the model described in [Henderson et al., 2014a, Henderson et al., 2014b]. Using a word-based approach, the authors successfully scaled the model to work on unseen slots and values. This is done by using the *n-gram* technique on top of the slot-value pairs. Moreover, in order to make the system slot invariant, *delexicalized* features are used, which means introducing generic symbols. Afterwards, a Recurrent Neural Network is used to discriminate between the dialogue states.

4.4.2 Policy Learning

The *Policy Learning* module selects the next system actions to drive the user towards the goal in the smallest number of steps. It does that by using the Reinforcement Learning theory.

The theory of Reinforcement Learning is motivated by the neuroscientific perspectives of the animal and human behavior, deeply and well rooted in the nature. Therefore, by mathematically modeling the nature, we have an intelligent agent acting in an environment and perceiving a state s. Upon taking an action a, based

on the policy learned from the past experiences, it is receiving a reward r and it changes the state of the environment.

Therefore, the role of the dialogue agent is to learn an optimal policy for conducting efficient and successful dialogue with the user. This is done by following the reinforcement way of learning, defining final, and immediate reward, such that the agent should maximize the cumulative future reward. One way to do so, is by applying on off-policy method, such as Q-learning [Watkins and Dayan, 1992]. The Q-function is the utility of taking an action a, when the agent is perceiving the state s, by following a policy $\pi = P(a|s)$. The utility measure is defined as the problem of maximizing the cumulative future reward that the agent will receive.

The optimal action a, in a given state s, in a given time point t, according to the Q-learning is defined as:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s_t = s, a_t = a, \pi\right], \tag{4.1}$$

where $r_t, r_{t+1}, \ldots$ are the rewards at each time step, $\gamma \in [0,1]$ is the discount factor, i.e. the relevance of the future rewards, and $\pi = P(a|s)$ is the agent's policy. The optimal action-value (or Q) function obeys an important identity known as the Bellman equation, which states:

$$Q^*(s,a) = r + \gamma \max_{a'} Q^*(s',a'). \tag{4.2}$$

Obviously, this agent would follow a greedy strategy, and will always exploit the same set of actions in order to reach the goal. In practice, we want the agent to generalize well over the state space. For this reason, the agents incorporate different exploitation-exploration strategies. The most popular and quite effective one is the ϵ - greedy strategy, for $\epsilon \in [0,1]$. That means with a probability of ϵ the agent will select a random action, while with a probability of $1-\epsilon$ it will follow a greedy approach.

However, by following the Bellman Equation 4.2, the objective function for learning

the Q-function would be:

$$\mathbb{E}_{s,a,r,s'}\left[\left(\overbrace{r+\gamma\max_{a'}Q\left(s',a'\right)}^{\text{target value}}-\overbrace{Q\left(s,a\right)}^{\text{old value}}\right)^2\right]. \tag{4.3}$$

Therefore, in order to find a function approximation of the Q-function, we have to minimize the equation 4.3.

The recently developed method, by a group of researches at Deep Mind, successfully applies a deep feed-forward neural network as a function approximator of the Q-function. Detailed information about the Deep Q-learning (DQN) technique and how to build more efficient policies using them will be provided in Chapter 5.

5 Efficient Policy Learning via Transfer Learning

In Chapter 4, we introduced the Q-learning and the very recent, popular and effective technique for approximating the Q-function using a deep feed forward neural network, know as Deep Q-Learning (DQN). In this chapter, we will dive in depth, explaining the advantages of the DQN, since we use it extensively in our dialogue systems.

5.1 Deep Q-Learning (DQN)

There are two types of DQN algorithms: *i*) standard DQN and *ii*) Double DQN (DDQN) which is an extension and more robust version of the standard DQN algorithm.

5.1.1 Standard DQN

The agents should be able to generalize well, over a high-dimensional, partially observable and complex input. Exactly this was the difficulty that most of the RL algorithms were facing, so their applicability was mainly in domains of fully observable, finite and low-dimensional states. However, all of this changed after introducing the Deep Q-Network, by a group of researchers at Deep Mind [Mnih et al., 2015]. It is a standard deep feed-forward neural network, which approximately calculates $Q(s, a|\,\theta)$, where θ are the parameters, (i.e. weights) of the Q-Network.

As we already explained, the goal in the reinforcement learning is to minimize the equation 4.3. However, in the RL community it is widely known that a nonlinear approximator of the Q-function, such as a neural network, causes instability and

divergence. This is due to the following two reasons:

1. The correlation between the sequence of observations, i.e. every next state depends on the previous states and actions, and
2. The targets (labels), depend on the network weights. More precisely, to calculate the target $r + \gamma \max_{a'} Q(s', a')$, used as a correction, we used the same weights which are changing through the time. This is in total contrast with the supervised learning, where the targets are fixed before the learning starts.

The first problem is solved by using a biologically inspired mechanism, called *experience replay*, i.e. to learn from experiences from an arbitrary point in the past. In order to perform such a mechanism, we store the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ in a dataset $D_t = \{e_1, \cdots, e_t\}$ of tuples. Therefore, at training time, we randomly sample experiences from the dataset D_t, following some probability distribution, which in the simplest case is a uniform distribution. Using this technique, we overcome the first problem, but we are still not able to perform learning in the network, due to the second issue, the targets still depend on the network weights.

For this reason, we introduce another Q-Network, called *target network*, with fixed parameters θ'. As its name suggests, this network is only used to calculate the targets $Q^{\star}(s', a'|\theta')$, independently from the primary Q-Network. The target network parameters are only updated with the primary Q-Network parameters every C steps and are held fixed between individual updates.

Thus, with the randomly drawn mini-batch of experiences and the target Q-Network we perform learning using the following loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} \overbrace{Q(s', a'|\theta')}^{\text{calc. by target net}} - \overbrace{Q(s, a|\theta)}^{\text{calc. by Q-net}}\right)^2\right] \tag{5.1}$$

The full algorithm for the Deep Q-Learning with experience reply is described in more details in Appendix A.1.

To evaluate the performances of the DQN algorithm, the researchers at Deep Mind took advantage of the Atari 2600 platform, offering 49 challenging games. The

DQN algorithm outperformed the best existing reinforcement learning methods on 43 games without incorporating any prior knowledge about the Atari 2600 games. These outstanding results confirmed the superiority of the DQN algorithm and established it as a state-of-the-art technique in the Reinforcement Learning community.

5.1.2 Double DQN (DDQN)

The standard DQN algorithm is based on the Bellman equation, thus it includes maximization step as shown in Equation 5.1. For this reason, it learns unrealistically high action values, which tends to prefer overestimated to underestimated values.

In their paper [Van Hasselt et al., 2016], theoretically prove that the overestimations occur non-uniformly and negatively affect the performance of the DQN algorithm. Therefore, they proposed an extension of the standard DQN algorithm, called Double DQN (DDQN), in order to overcome these issues.

In the standard DQN algorithm, the decision for the next action is taken according to the following identity: $r + \gamma \max_{a'} Q(s', a')$. For this reason, the same neural network is used to evaluate the Q-function and then to select the best action.

In Double DQN, this process is decoupled. One neural network is used to evaluate the Q-function and a second neural network is used to select the best action. In mathematical notation, the next action is taken according to the following identity: $r + \gamma Q(s', \text{argmax}_{a'} Q(s', a'))$.

5.2 DQN for GO Chatbots

The applications of the DQN algorithm are not only limited to the Atari 2600 games. Very recently the researchers started applying the Deep Q-Learning to various tasks, including the Goal-Oriented Dialogue Systems.

In the case of Goal-Oriented Chatbots, the agent is getting the new state s_t from the Dialogue State Tracker (DST) and then it takes a new action a_t, based on the ϵ-greedy policy. It means, with a probability $\epsilon \in [0, 1]$ it will take a random action, while with a probability $1 - \epsilon$ it will take the action resulting with a maximal Q-value. We thus trade between the exploration and exploitation of the dialogue space. For each slot that might appear in the dialogue, the agent can take two actions: either to ask the user for a constraining value or to suggest to the user a

value for that slot. Additionally, there is a fixed size of slot-independent actions, to open and close the conversation.

The agent receives positive and negative rewards accordingly, in order to force the agent to successfully conduct the dialogue. It is *successful* if the number of totally required dialogue turns to reach the goal is less than a predefined maximal threshold n_{max_turns}. For every additional dialogue turn, the agent receives a predefined negative reward $r_{ongoing}$. In the end, if the dialogue fails, it will receive a negative reward $r_{negative}$ equal to the negative of the predefined maximal allowed dialogue turns. If the dialogue is successful, it will receive a positive reward $r_{positive}$, two times the maximal allowed dialogue turns.

An important addition is the *warm-starting* technique that fills the experience replay buffer with experiences coming from a successfully finished dialogues i.e. with positive experiences. This dramatically boosts the agent's performances before the actual training starts, as will be shown in Section 6.2.3. The training process continues with running a fixed number of independent training epochs n_{epochs}. In each epoch we simulate a predefined number of dialogues $n_{dialogues}$, thus filling the experience memory buffer. The result consists of mini-batches to train the underlying Deep Q-Net.

During the training process, when the agent reaches for the first time a success rate greater or equal to the success rate of a rule-based agent s_{rule_based}, we flush the experience replay buffer, as described in detail in [Li et al., 2017]. We do this because the DQN-based agent cannot produce valuable experiences in the beginning, which are all stored in the experience buffer.

5.3 Transfer Learning

The main goal of this work is to study the impact of a widely used technique - *Transfer Learning* on goal oriented bots. As the name suggests, transfer learning transfers knowledge from one neural network to another. The former is known as the source, while the latter is the target [Pan and Yang, 2010]. The goal of the transfer is to achieve better performance on the target domain with limited amount of training data, while benefiting from additional information from the source domain. In the case of dialogue systems, the input space for both source and target nets are their respective dialogue spaces.

The training process without transfer learning, shown in Figure 5.1a, processes the two dialogue domains independently, starting from randomly initialized weights.

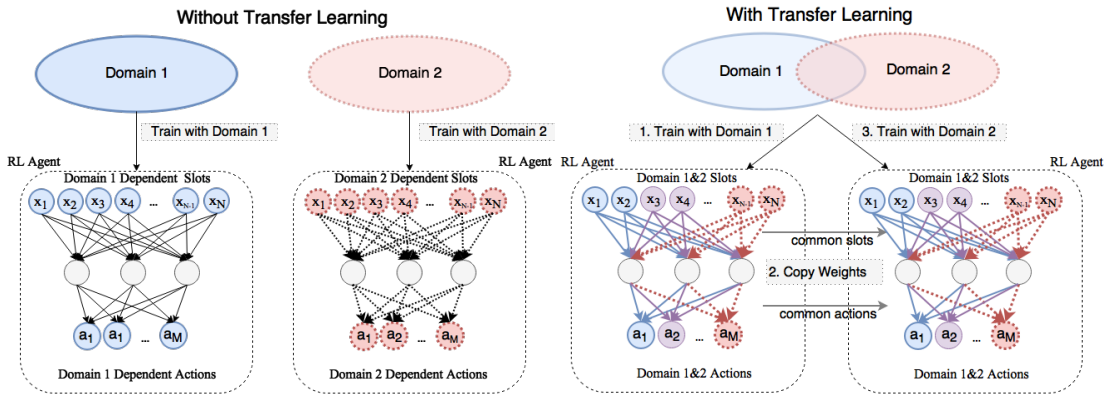


Figure 5.1 – Comparison of the Goal-Oriented Dialogue System training process, without transfer learning (left side) and with transfer learning (right side).

The results are dialogue states from separate distributions. Additionally, the sets of actions the agents might take in each domain are also independent.

On the other hand, as depicted in Figure 5.1b if we want to benefit from transfer learning, we must model the dialogue state in both domains, as if they were coming from the same distribution. The sets of actions have to be shared too. The bots specialized in the source domain must be aware of the actions in the second domain, even if these actions are never used, and vice versa. This requirement stems from the impossibility of reusing the neural weights if the input and output spaces differ. Consequently, when we train the model on the source domain, the state of the dialogue depends not only on the slots that are specific to the source, but also on those that only appear in the target one. This insight can be generalized to a plurality of source and target domains. The same holds for the set of actions.

Algorithm 1 Transfer Learning Pseudocode

```
1: procedure InitializeWeights(sourceWeights, commonSlotIndices, commonActionIndices)
2:     targetWeigths ← RandInit()
3:     for i in commonSlotIndices do
4:         targetWeigths[i] ← sourceWeights[i]
5:     for i in commonActionIndices do
6:         targetWeigths[i] ← sourceWeights[i]
7:     return targetWeigths
```

When training the target domain model, we no longer randomly initializing all weights. The weights related to the source domain - both for slots and actions -

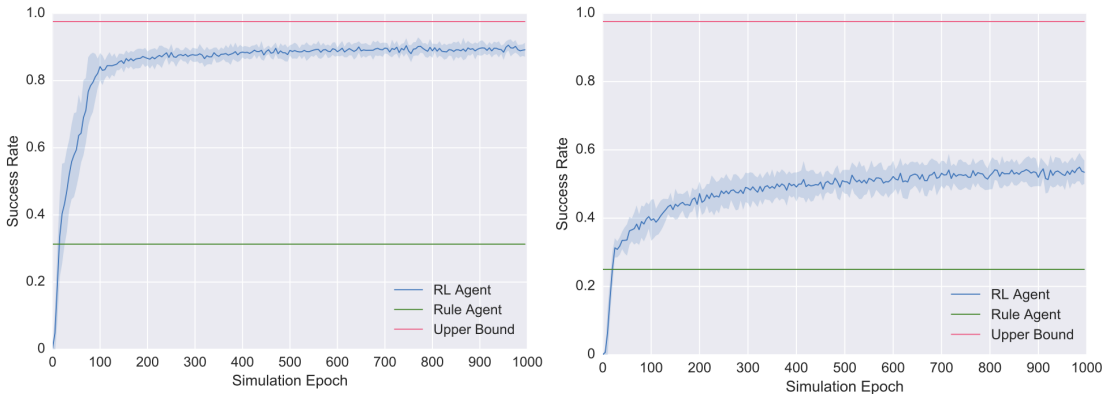
are copied from the source model. The pseudocode for this weight initialization is portrayed in the Algorithm 1.

6 Experiments and Results

In this chapter we present all valuable experiments and the obtained results. Our work is based on the work from [Li et al., 2017]. For this reason, in the first part we briefly present the baseline experiments and results. In the second part we focus on the transfer learning experiments.

6.1 Baseline Experiments

In all baseline experiments, the Chatbot is trained on the *Movie Booking* domain. The type of slots for this domain are given in Figure 6.2. The size of the training set is 128 user goals. The maximal number of allowed dialogue turns is set to $n_{max\_turns} = 20$, thus the negative reward for a failed dialogue is $r_{negative} = -20$,



(a) Learning curve on a semantic level (b) Learning curve on a natural language level

Figure 6.1 – Baseline experimental results for 100 runs with a 95% confidence interval

while the positive reward for a successful dialogue is $r_{positive} = 40$. In all experiments we use a warm-starting and train for $n_{epochs} = 1000$ epochs, each simulating $n_{dialogues} = 100$ dialogues. This is a bit huge number of epochs to run and we believe that the chatbot is overfitting. However, our intention was to reproduce the results from the baseline paper.

We present the results for both levels: semantic level and natural-language level. In Figure 6.1a the learning curve for the training on semantic level is shown, while in Figure 6.1b the learning curve for the training on natural language level is shown. The same experiment is repeated 100 times, thus the results are reported with a 95% confidence interval. Due to the noise introduced by the NLU and NLG unit, the Chatbot performance on natural language level is considerably lower than the performance on the semantic level.

6.2 Transfer Learning Experiments

In this set of experiments, we operate on the semantic level, removing the noise introduced by the NLU and NLG units. We want to focus exclusively on the impact of transfer learning techniques on dialog management. The details of the system implementation[1] are presented in Appendix A.2.

6.2.1 Setup of Experiments

All experiments are executed using a setup template. Firstly, we train a model on the *source domain* and reuse the common knowledge to boost the training and testing performance of the model trained on a different, but similar *target domain*. Secondly, we train a model exclusively on the target domain, without any prior knowledge. This serves as a baseline. Finally, we compare the results of these two models. We thus have two different cases:

1. *Domain Overlap* - the source *MovieBooking* and target *RestaurantBooking* domains are different, but share a fraction of the slots.

2. *Domain Extension* - the source domain, now *RestaurantBooking*, is extended to *Tourist Information*, that contains all the slots from the source domain along with some additional ones.

[1]Link to the GitHub repository: https://github.com/IlievskiV/Master_Thesis_GO_Chatbots

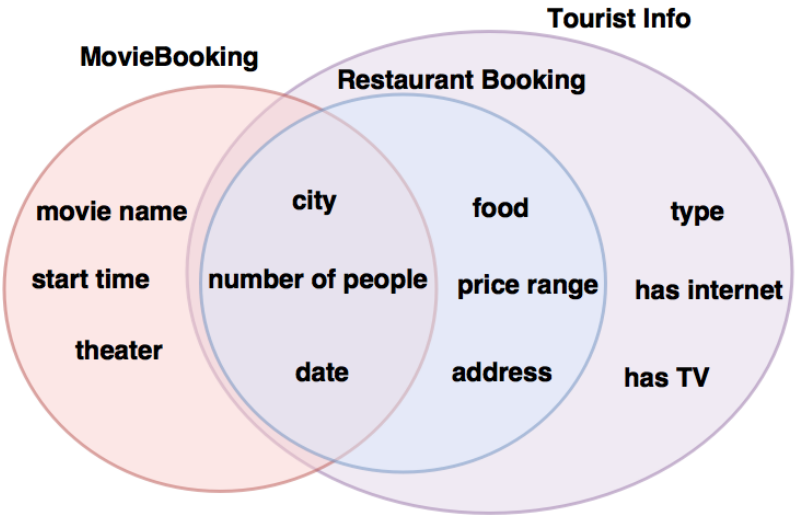


Figure 6.2 – Slot types in the three different domains

The reason for the choice of source domain in the domain overlap case is designed to enable a direct comparison to the results of [Li et al., 2017], who built a GO bot for movie booking. For the domain extension case, the only combination available was $Restaurant - Tourism$. The type of slots in each domain are given in Figure 6.2. For each domain, we have a training set of 120 user goals, and a testing set of 32 user goals.

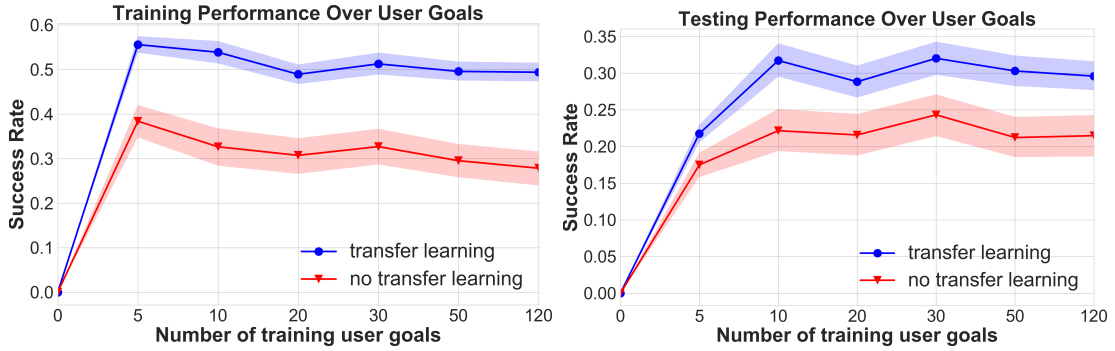
Following the above mentioned setup template, we conduct two sets of experiments for each of the two cases. The first set shows the overall performance of the models leveraging the transfer learning approach. The second set shows the effects of the warm-starting jointly used with the transfer learning technique.

In all experiments, when we use a warm-starting, the criterion is to fill agent's buffer, such that 30 percent of the buffer is filled with positive experiences (coming from a successful dialogue). After that, we train for $n_{epochs} = 50$ epochs, each simulating $n_{dialogues} = 100$ dialogues. We flush the agent's buffer when the agent reaches, for a first time, a success rate of $s_{rule\_based} = 0.3$. We set the maximal number of allowed dialogue turns n_{max_turns} to 20, thus the negative reward $r_{negative}$ for a failed dialogue is -20, while the positive reward $r_{positive}$ for a successful dialogue is 40. In the consecutive dialogue turns over the course of the conversation, the agent receives a negative reward of $r_{ongoing} = -1$. In all cases we set $\epsilon = 0.05$ to leave a space for exploration. By using this hyperparameters, we prevent the system to overfit and to generalize very well over the dialogue space. Finally, we report the success rate as a performance measure.

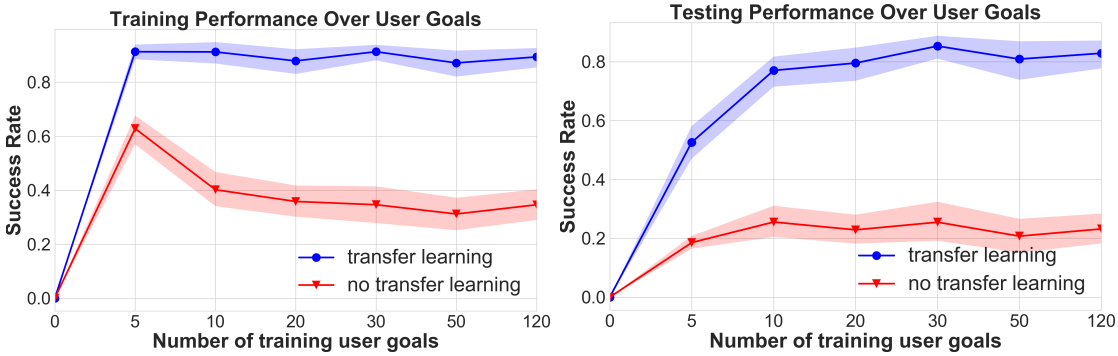
6.2.2 Training with Less Data

Due to labeling costs, the availability of in-domain data is the bottleneck for training successful and high performing Goal-Oriented chatbots. We thus study the effect of transfer learning on training bots in data-constrained environments.

From the available 120 user goals for each domain's training set, we randomly select subsets of 5, 10, 20, 30, 50 and all 120. We then warm-start and train both the independent and transfer learning models on these sets. We test the performance on both the training set (*training performance*) and the full set of 32 test user goals (*testing performance*). We repeat the same experiment 100 times, in order to reduce the uncertainty introduced by the random selection. Finally, we report the success rate over the user goal portions with a 95% confidence interval.



(a) Restaurant Booking with pre-training on Movie Booking domain



(b) Tourist Info with pre-training on Restaurant Booking domain

Figure 6.3 – Average training and testing success rates with 95% confidence, for 100 runs over a randomly selected user goal portions of size 5, 10, 20, 30, 50 and 120, for both models: with and without transfer learning.

The training and testing results, in the first case of domain overlapping, are shown in Figure 6.3a. The success rate of the model obtained with transfer learning is 65% higher than that of the model trained without any external prior knowledge.

In absolute terms the success rate climbs on average from 30% to 50%. For the test dataset, transfer learning improves the success rate from 25% to 30%, for a still noteworthy 20% relative improvement.

In the case of domain extension, the difference between the success rates of the two models is even larger (Figure 6.3b). This was expected, as the extended target domain contains all slots from the source domain, therefore not losing any source domain information. The overall relative success rate boost for all user goal portions is on average 112%, i.e. a move from 40% to 85% in absolute terms. For the test set, this difference is even larger, from 22 to 80% absolute success rate, resulting in 263% relative boost.

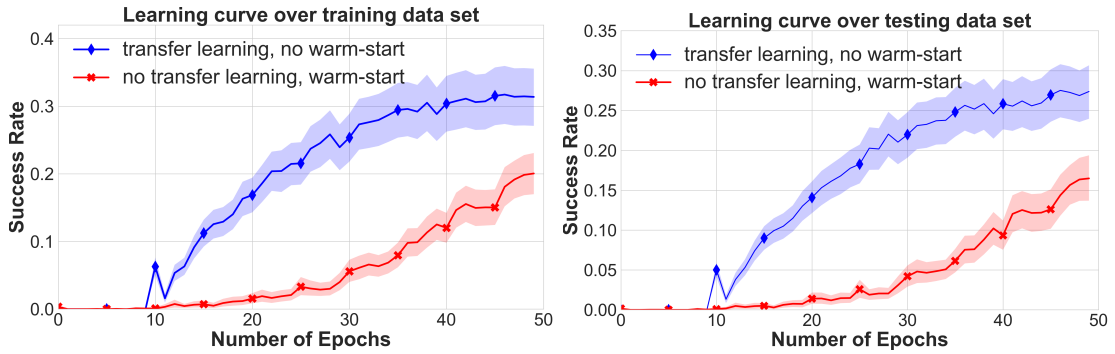
These results show that transferring the knowledge from the source domain, we boost the target domain performance in data constrained regimes.

6.2.3 Faster Learning

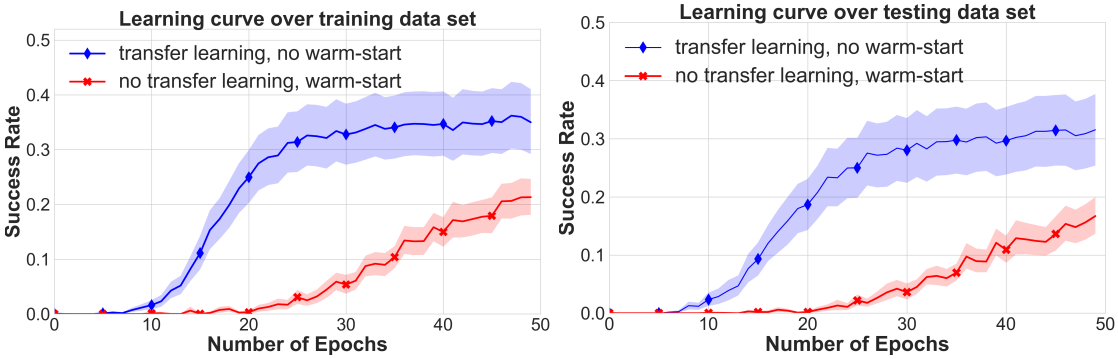
In a second round of experiments, we study the effects of the transfer learning in the absence and in combination with the warm-starting phase. As warm starting requires additional labeled data, removing it further reduces the amount of labeled data needed. We also show that the two methods are compatible, leading to very good joint results.

We report the training and testing learning curves (success rate over the number of training epochs), such that we use the full dataset of 120 training user goals and the test set of 32 user goals. We repeat the same process 100 times and report the results with a 95% confidence interval. The performances in the first case of domain overlapping are shown in Figure 6.4a, while for the other case of domain extension, in Figure 6.4b. The bot using transfer learning, but no warm-starting, shows better learning performances than the warm-started model without transfer learning. Transfer learning is thus a viable alternative to warm starting.

However, models based on transfer learning have a significant variance, as the learning is progressing. This happens because in many experiment runs the success rate over all epochs is 0. In those cases, the agent does not find an optimal way to learn the policy in the early stages of the training process. This results with filling its experience replay buffer mostly with negative experiences. Consequently, in the later stages, the agent is not able to recover. This makes a combination with warm starting desirable.



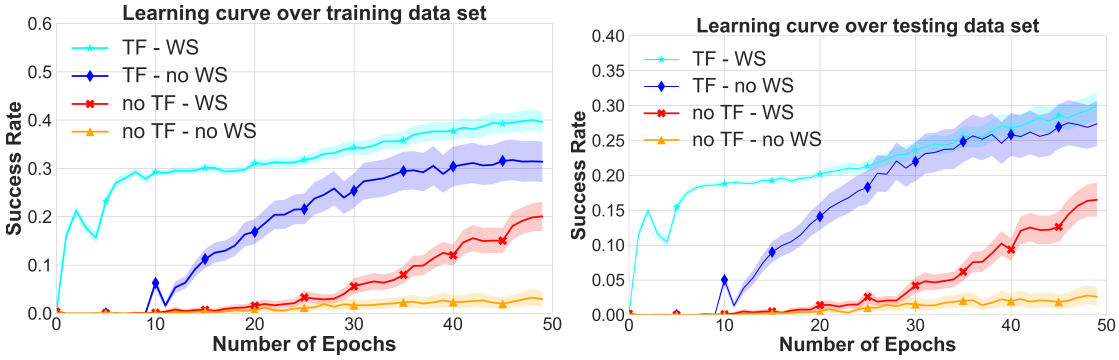
(a) Restaurant Booking with pre-training on Movie Booking domain



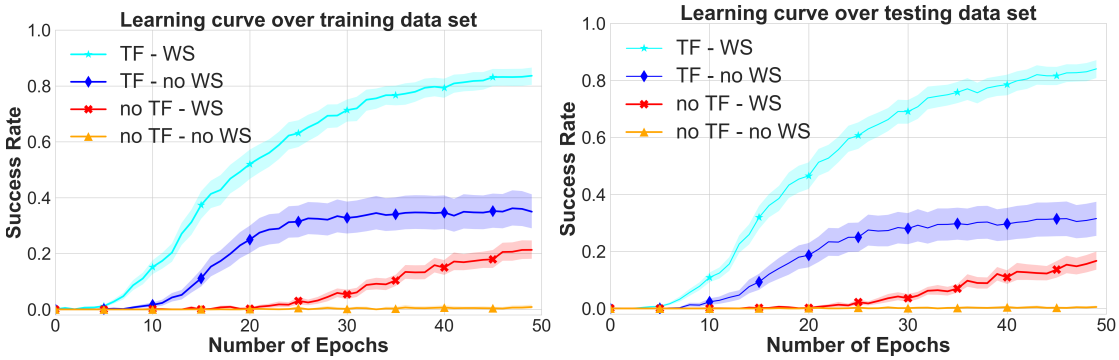
(b) Tourist Info with pre-training on Restaurant Booking domain

Figure 6.4 – Average training and testing success rates with 95% confidence, for 100 runs over the number of epochs, for both models: with and without transfer learning. The model with transfer learning is not warm-started.

For convenience reasons, in Figure 6.5 we show all possible cases of using and combining the transfer learning and warm-starting techniques. We can see that the model combines the two techniques performs the best by a wide margin. This leads to a conclusion that the transfer learning is complimentary to the warm-starting, such that their joint application brings the best outcomes.



(a) Restaurant Booking with pre-training on Movie Booking domain



(b) Tourist Info with pre-training on Restaurant Booking domain

Figure 6.5 – Success rates for all model combinations - with and without Transfer Learning (TF), with and without Warm Starting (WS).

7 Conclusion and Future Work

In this thesis, we examined in depth the Goal-Oriented Chatbots, especially the Reinforcement Learning-based. We show that the *Transfer Learning* technique can be successfully applied to boost the performances of the RL-based Goal-Oriented Chatbots. We do this for two different use cases: *i*) when the source and the target domain overlap, and *ii*) when the target domain is an extension of the source domain.

We show the advantages of the transfer learning in a low data regime for both cases. When a low number of user goals is available for training in the target domain, transfer learning makes up for the missing data. Even when the whole target domain training data is available, the transfer learning benefits are maintained, with the success rate increasing threefold.

We also demonstrate that the transfer knowledge can be a replacement of the warm-starting period in the agents or can be combined with it for best results.

Last but not the least, we create and share two datasets for training Goal-Oriented Dialogue Systems in the domains of Restaurant Booking and Tourist Information.

With the promising results we achieved during the work on this thesis, the following possibilities are worth trying to investigate into:

- Study the effects of the *Transfer Learning* when the Chatbot is working on a natural language level.
- Build better *User Simulators*, since the simulator in this work is hand-crafted and too limited.
- To build better and more robust Dialogue State Trackers.

A Appendix A

A.1 Standard DQN algorithm

Algorithm 2 Deep Q-Learning with Experience Replay [Mnih et al., 2015]

procedure DQN(N, M, T, ϵ, γ)
  Initialize replay memory $\mathcal{D}$ to capacity N
  Initialize action-value function Q with random weights θ
  for *episode* in $1, M$ **do**
    Initialize sequence $s_1 = x_1$ and preprocessed sequence $\phi_1 = \phi(s_1)$
    for t in $1, T$ **do**
      With probability ϵ take a random action a_t,
      otherwise select $a_t = \max_a Q^\star(\phi(s_t), a; \theta)$
      Execute action a_t and observe a reward r_t and a new state x_{t+1}
      Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(x_{t+1})$
      Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
      Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
      Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q^\star(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
      Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ (Eq. 5.1)

A.2 System Implementation

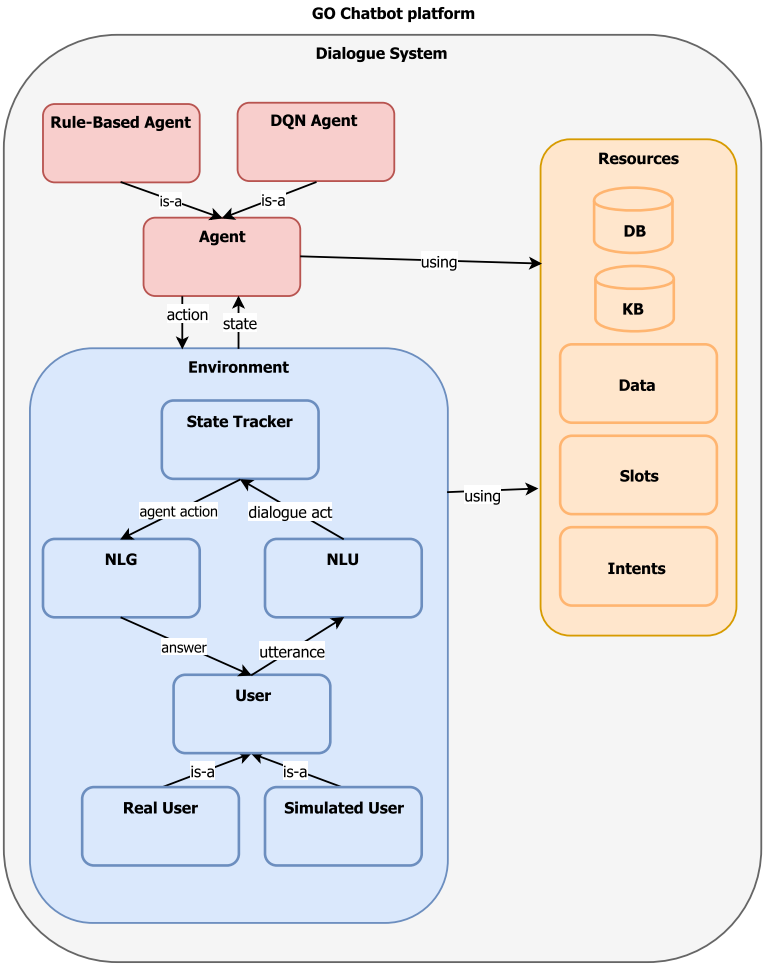


Figure A.1 – High-level overview of the system for training Goal-Oriented Chatbots

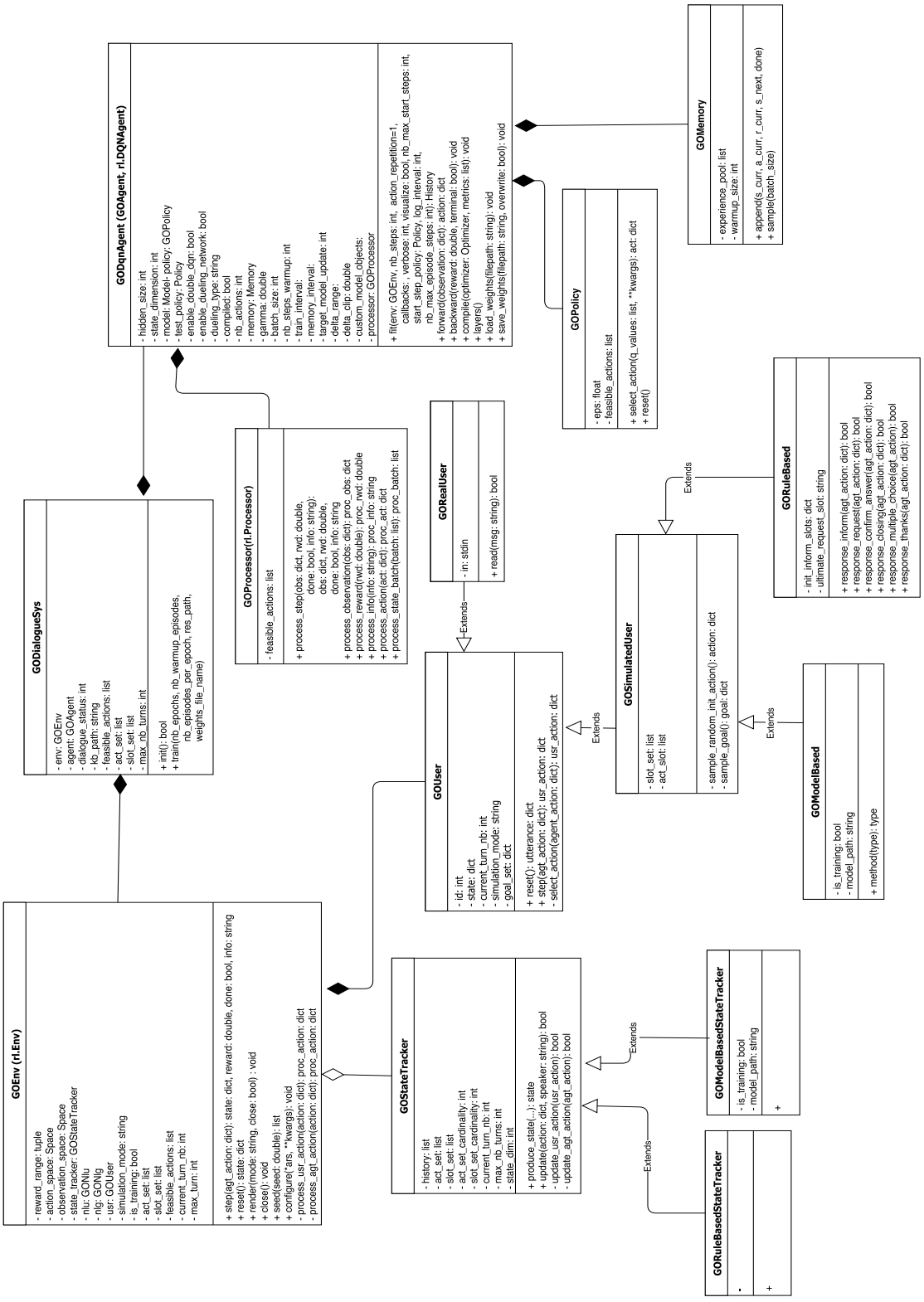


Figure A.2 – UML Diagram of the system for training Goal-Oriented Chatbots

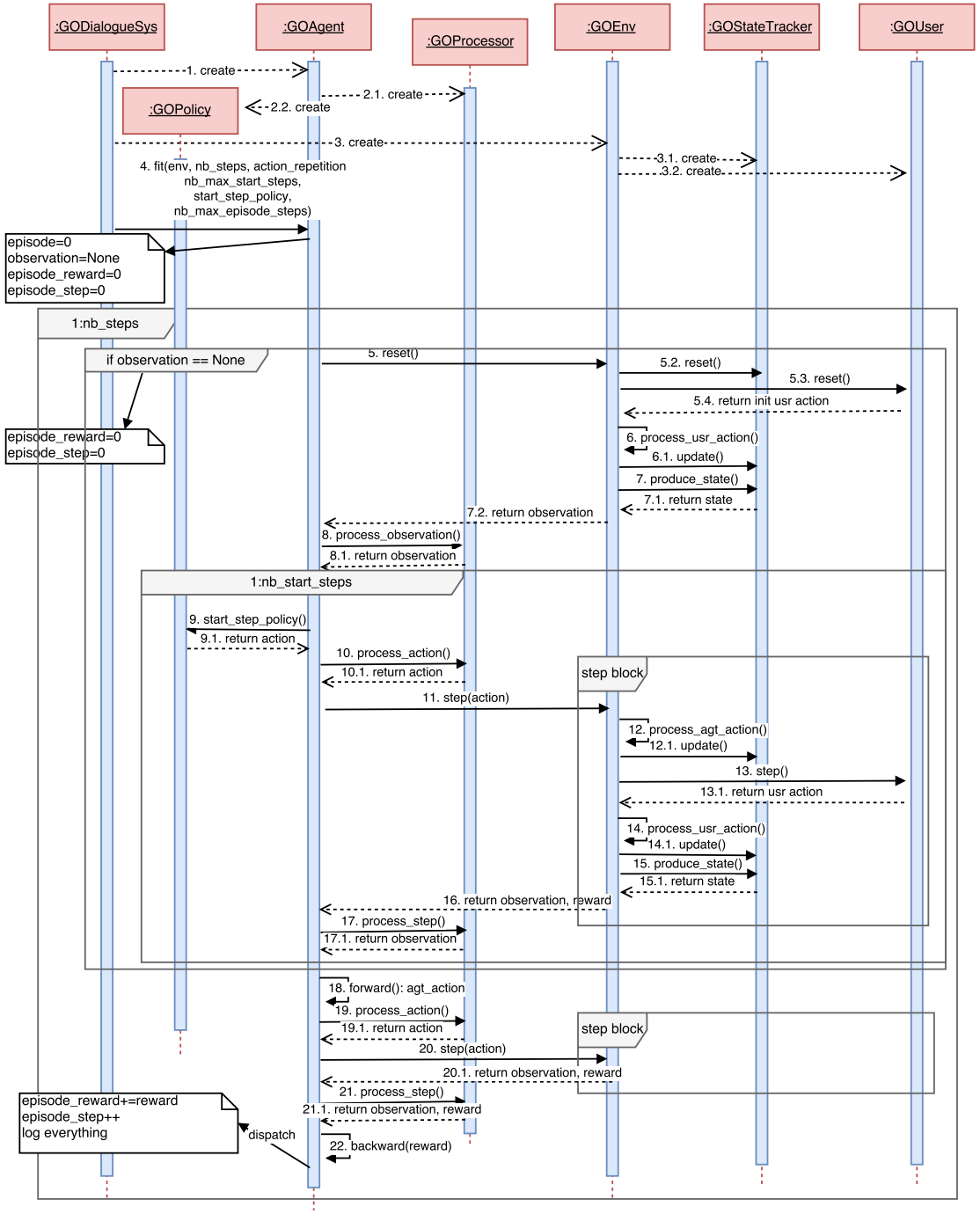


Figure A.3 – Sequence Diagram of the system for training Goal-Oriented Chatbots

Bibliography

[Banchs, 2012] Banchs, R. E. (2012). Movie-dic: a movie dialogue corpus for research and development. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 203–207. Association for Computational Linguistics.

[Bordes and Weston, 2016] Bordes, A. and Weston, J. (2016). Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

[Cuayáhuitl, 2017] Cuayáhuitl, H. (2017). Simpleds: A simple deep reinforcement learning dialogue system. In *Dialogues with Social Robots*, pages 109–118. Springer.

[Dhingra et al., 2016] Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y.-N., Ahmed, F., and Deng, L. (2016). End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.

[Dimovski et al., 2018] Dimovski, M., Ilievski, V., Musat, C., Hossmann, A., and Baeriswyl, M. (2018). Submodularity-inspired data selection for goal-oriented chatbot training based on sentence embeddings. *arXiv preprint arXiv:1802.00757*.

[Epstein, 1992] Epstein, R. (1992). The quest for the thinking computer. *AI magazine*, 13(2):81.

[Gašić et al., 2015] Gašić, M., Kim, D., Tsiakoulis, P., and Young, S. (2015). Distributed dialogue policies for multi-domain statistical dialogue management. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5371–5375. IEEE.

[Hakkani-Tür et al., 2016] Hakkani-Tür, D., Tür, G., Celikyilmaz, A., Chen, Y.-N., Gao, J., Deng, L., and Wang, Y.-Y. (2016). Multi-domain joint semantic frame parsing using bi directional rnn-lstm. In *INTERSPEECH*, pages 715–719.

[Henderson, 2015a] Henderson, M. (2015a). Machine learning for dialog state tracking: A review. In *Proc. of The First International Workshop on Machine Learning in Spoken Language Processing*.

[Henderson, 2015b] Henderson, M. (2015b). Machine learning for dialog state tracking: A review. In *Proceedings of The First International Workshop on Machine Learning in Spoken Language Processing*.

[Henderson et al., 2013] Henderson, M., Thomson, B., and Williams, J. (2013). Dialog state tracking challenge 2 & 3.

[Henderson et al., 2014a] Henderson, M., Thomson, B., and Young, S. (2014a). Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299.

[Henderson et al., 2014b] Henderson, M., Thomson, B., and Young, S. J. (2014b). Robust Dialog State Tracking Using Delexicalised Recurrent Neural Networks and Unsupervised Adaptation. In *Proceedings of IEEE Spoken Language Technology*.

[Henderson, 2015c] Henderson, M. S. (2015c). *Discriminative methods for statistical spoken dialogue systems*. PhD thesis, University of Cambridge.

[Ilievski et al., 2018] Ilievski, V., Musat, C., Hossmann, A., and Baeriswyl, M. (2018). Goal-oriented chatbot dialog management bootstrapping with transfer learning. *arXiv preprint arXiv:1802.00500*.

[Kurzweil, 2010] Kurzweil, R. (2010). *The singularity is near*. Gerald Duckworth & Co.

[Larsson and Traum, 2000] Larsson, S. and Traum, D. R. (2000). Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural language engineering*, 6(3-4):323–340.

[Lee, 2017] Lee, S. (2017). Toward continual learning for conversational agents. *arXiv preprint arXiv:1712.09943*.

[Li et al., 2017] Li, X., Chen, Y.-N., Li, L., and Gao, J. (2017). End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

[Li et al., 2016] Li, X., Lipton, Z. C., Dhingra, B., Li, L., Gao, J., and Chen, Y.-N. (2016). A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

[Liu and Lane, 2016] Liu, B. and Lane, I. (2016). Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[Pan and Yang, 2010] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[Peng et al., 2017] Peng, B., Li, X., Li, L., Gao, J., Celikyilmaz, A., Lee, S., and Wong, K.-F. (2017). Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2221–2230.

[Pieraccini and Huerta, 2005] Pieraccini, R. and Huerta, J. (2005). Where do we go from here? research and commercial spoken dialog systems. In *6th SIGdial Workshop on Discourse and Dialogue*.

[Schatzmann and Young, 2009] Schatzmann, J. and Young, S. (2009). The hidden agenda user simulation model. *IEEE transactions on audio, speech, and language processing*, 17(4):733–747.

[Serban et al., 2016] Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

[Strayer et al., 2017] Strayer, D. L., Cooper, J. M., Turrill, J., Coleman, J. R., and Hopman, R. J. (2017). The smartphone and the driver's cognitive workload: A comparison of apple, google, and microsoft's intelligent personal assistants. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 71(2):93.

[Su et al., 2016] Su, P.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. (2016). Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*.

[Sukhbaatar et al., 2015] Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

[Van Hasselt et al., 2016] Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pages 2094–2100.

[Vinyals and Le, 2015] Vinyals, O. and Le, Q. (2015). A neural conversational model. *arXiv preprint arXiv:1506.05869*.

[Wang et al., 2015] Wang, Z., Wen, T.-H., Su, P.-H., and Stylianou, Y. (2015). Learning domain-independent dialogue policies via ontology parameterisation.

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

[Wen et al., 2016a] Wen, T.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L. M., Su, P.-H., Ultes, S., Vandyke, D., and Young, S. (2016a). Conditional generation and snapshot learning in neural dialogue systems. *arXiv preprint arXiv:1606.03352*.

[Wen et al., 2015] Wen, T.-H., Gasic, M., Mrksic, N., Su, P.-H., Vandyke, D., and Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

[Wen et al., 2016b] Wen, T.-H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L. M., Su, P.-H., Ultes, S., and Young, S. (2016b). A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

[Williams et al., 2016] Williams, J., Raux, A., and Henderson, M. (2016). The dialog state tracking challenge series: A review. *Dialogue & Discourse*.

[Williams et al., 2017] Williams, J. D., Asadi, K., and Zweig, G. (2017). Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

[Yao et al., 2014] Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., and Shi, Y. (2014). Spoken language understanding using long short-term memory neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 189–194. IEEE.

[Young et al., 2013] Young, S., Gašić, M., Thomson, B., and Williams, J. D. (2013). Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

[Zen et al., 2009] Zen, H., Tokuda, K., and Black, A. W. (2009). Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064.

[Zhang et al., 2017] Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Bengio, C. L. Y., and Courville, A. (2017). Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*.

[Zhao and Eskenazi, 2016] Zhao, T. and Eskenazi, M. (2016). Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*.

[Zue et al., 2000] Zue, V., Seneff, S., Glass, J. R., Polifroni, J., Pao, C., Hazen, T. J., and Hetherington, L. (2000). Juplter: a telephone-based conversational interface for weather information. *IEEE Transactions on speech and audio processing*, 8(1):85–96.