Task-oriented Dialog Policy Learning via Deep Reinforcement Learning and Automatic Graph Neural Network Curriculum Learning

Master Thesis

29 May 2023

Author: **Koen Hanneman (5743931)**
k.a.hanneman@students.uu.nl

1st Supervisor and Examiner: **Shihan Wang**
s.wang2@uu.nl

Daily Supervisor: **Yanyang Zhao**
y.zhao7@uu.nl

2st Examiner: **Mehdi M. Dastani**
m.m.dastani@uu.nl

Utrecht University

MSc Artificial Intelligence
Graduate School of Natural Sciences
Utrecht University

Abstract

In a task-oriented dialog system, a core component is the dialog policy, which determines the response action and guides the conversation system to complete the task. Optimizing such a dialog policy is often formulated as a reinforcement learning (RL) problem. But given the subjectivity and open-ended nature of human conversations, the complexity of dialogs varies greatly and negatively impacts the training efficiency of the RL-based method. A proven method to solve this problem is curriculum learning (CL) which breaks down complex problems and improves learning efficiency by providing a sequence of learning steps of increasing difficulty, similar to human learning. However, existing models implement this sequence by ordering tasks just based on complexity, without taking into account task similarity. In this thesis, we propose a method that reduces the distance between similar tasks in a curriculum, which is hypothesised to lead to increased training efficiency. Therefore, we introduce a curriculum learning model by offline generating a sequence of similar tasks via a graph neural network (GNN), and where the low-level dialog policy is transferred in each iteration of the curriculum. After this, the curriculum learning model performance is compared, on the MultiWOZ dataset, against the performance of dialog policy learning without a curriculum and was found to outperform the baseline model in specific scenarios.

Contents

1 Introduction **3**

2 Background **5**
2.1 Task-Oriented Dialog Systems 5
2.2 Reinforcement Learning 7
2.3 Curriculum Learning 9
2.4 Graph Neural Network 10

3 Related Work **10**
3.1 Deep Reinforcement Learning for Dialog Policy Learning 11
3.2 Conversation Graphs 12
3.3 Curriculum Learning for RL-based Dialog Policy Learning 12
3.3.1 Experience level CL for RL-based dialog policy learning 12
3.3.2 Task level CL for RL-based dialog policy learning 13

4 Research Question **14**

5 Method **14**
5.1 Conversation graph generation 15
5.1.1 Conversation graph definition 15
5.1.2 Node labels 16
5.1.3 Conversation generation 17
5.2 Conversation graph similarity 19
5.2.1 Graph neural network 19
5.3 Curriculum generation 20
5.3.1 Complexity metrics 20
5.3.2 Curriculum generation 21
5.4 Reinforcement Learning with a Curriculum 22

6 Results **25**
6.1 SimGNN 25
6.2 Curriculum Generation 26
6.3 Dialog Policy Learning with Curriculum Learning 27
6.3.1 Experiment 1 28
6.3.2 Experiment 2 28
6.3.3 Experiment 3 29
6.3.4 Experiment 4 30
6.3.5 Against the Basline DQN 30

7 Discussion **31**
7.1 Research Question and Conclusion 31
7.2 Future work 33

8 References **35**

1 Introduction

In the past decade, there has been a significant rise in the accessibility and quality of virtual personal assistants, which allow users to complete most daily tasks by speaking naturally. This goes for elementary tasks like informing about the weather or setting timers as seen, to even more abstract tasks like having a conversation. And recently the breadth of tasks has grown even more arising from the most recent upsurge in mainstream attention around ChatGPT from OpenAI, which was based on their InstructGPT model (Ouyang et al., 2022). This model's ability to follow very diverse instructions accurately, caused many to use the chatbot as a search engine, summary generator, text-writer, or even coder. That ChatGPT excelled in many varied areas is ultimately what propelled ChatGPT into the mainstream. It is also what separated it from the previously most advanced mainstream chatbots, the Intelligent Virtual Assistants (IVA) like Apple Siri, Google Assistant, and Amazon Alexa (Tulshan & Dhage, 2019) that are exceedingly voice activated (Hoy, 2018). In common parlance, these IVA chatbots have seemingly fallen behind in their perceived performance. However, the design goals of IVAs and Generative Artificial Intelligence (GenAI) like ChatGPT are very different (Huang et al., 2020). IVAs are designed to support users with accessing software operations within their personal devices, like managing a calendar or setting a timer. While GenAIs are designed to engage with the user by creating something outside their personal space, like written text or code. Even some functions that seemingly overlap between IVA and GenAI chatbots operate very differently. For example, when tasked with searching for the phone number of a restaurant, the IVA chatbot can directly look up the up-to-date phone number on the internet, while the GenAI chatbot is limited by only being able to reproduce the phone number from the archived data used to create the GenAI model. Therefore, the design goals of chatbots largely influence what a chatbot is able to do, and how it operates. This also means that improvements in the technology to create ChatGPT can often not be directly transferred to other chatbot types.

Given the wide variety of tasks that can be accomplished by chatbots, it is important to place them into distinct categories, so that improving a chatbot system can be accurately targeted. Generally, chatbots can be classified according to four aspects (Nimavat & Champaneria, 2017). The first aspect is a chatbot's knowledge domain which dictates what information the model is trained on and has access to. In an open knowledge domain, a chatbot can respond to general topics, as is possible with ChatGPT. In a closed knowledge domain, a chatbot only works within an intentional domain and will fail to respond to queries outside that domain. The second aspect is the proxemics-like distance of a chatbot's provided service. An interpersonal service is when the chatbot operates outside the user's personal space, while an intrapersonal service is when the chatbot behaves more like a companion. Service can also be inter-agent when the chatbot interacts with other chatbots. The third aspect is the chatbot's goal, which can be informative, conversational, or task-based. And the fourth aspect is the chatbot response generation method, which influences the conversation quality.

The specific chatbot category relevant to this thesis is the Task-Oriented Dialog System (Chen et al., 2017; Z. Zhang, Takanobu, et al., 2020). In essence, this type of chatbot is characterised by its expertise to complete specific tasks, similar to an IVA but can be more limited in the number of tasks it can perform. Task-oriented dialog systems have a closed knowledge domain, meaning the functionality of the system is focused on specialised proficiency. For example, ordering a taxi to a desired location. A closed knowledge domain is not limited to one topic or one task, and can include multiple topics or tasks. It does mean, however, that its limits are more explicitly defined as opposed to other chatbots. For the provided service aspect, task-oriented dialog systems are not limited to operating in either the user's personal domain or outside the user's personal domain. Dialog systems that help users within websites like a web shop are strictly interpersonal. But dialog systems that can book a restaurant are more intrapersonal when that reservation can be stored in the user's calendar. As indicated by the task-oriented dialog systems name, its goal is task-based. For a successful system, the aim is to complete assignments as given by the user. The response generation aspect is not relevant to describe the task-oriented dialog system conceptually but is of course important for the actual system implementation quality.

For task-oriented dialog systems to work optimally, it is necessary that the system can effectively guide a conversation to the user's desired goal. Because the design goal of task-oriented dialog systems is to complete tasks, while the pleasantness of a conversation that achieves a user goal is important, it is for the most part about the content of the conversation because that directly impacts if a user's goal is even achieved (Walker et al., 1997). For example, if a user wants to order a taxi and prefers one company, but the system keeps suggesting rides for anything but that company in a polite manner, it will be impossible to complete that user's goal. Absurd scenarios like this are not common in any competent dialog system, but it is intuitive that impeding a user can also occur more subtly. Particularly because each action of the dialog system affects the response of the user, and a user getting lost in the conversation is also sub-optimal. A core factor of the task-oriented dialog system is, therefore, its behaviour. The component that selects actions is called the dialog policy, and as explained it is crucial to obtain a quality dialog policy.

Depending on how varied conversations can be within the dialog systems' knowledge domain, obtaining a well-performing dialog policy that accomplishes tasks for this domain can become challenging. If a dialog policy were to be manually made, the challenge becomes avoiding human error when hand-crafting all the code that covers every possible interaction and overcoming the time cost to create all that code. Often then, for large knowledge domains, it is more beneficial to acquire the dialog policy through some form of machine learning. In Dialog Policy Learning (DPL), obtaining an optimal dialog policy is solved by utilizing machine learning. DPL is often formulated as a Reinforcement Learning (RL) problem, where the dialog systems behaviour is learned in an action-reward structure that models a user/user-simulator as an interactive environment (X. Li et al., 2017). Essentially, in this method, the user simulator rewards the quality/productive responses of the system. These rewards promote the intended system behaviour.

However, the problem with the RL-based method is that when training data varies the learning time increases. And given the subjectivity and open-ended nature of human conversations, the complexity of dialogs between different users varies greatly. A proven method to solve this problem is Curriculum Learning (CL) (Peng et al., 2018) which aims to mimic the human procedure of progressively learning from simple concepts to difficult ones. It breaks down complex problems and improves human learning efficiency by providing a sequence of learning steps of increasing difficulty. Most recently, curriculum learning has also seen some use in dialog policy learning (Liu et al., 2021; Zhao et al., 2021), although its use is still limited. Here, the curriculum is automatically generated according to task complexity. However, this method does not consider the similarity between tasks. Reducing the distance between similar tasks in a curriculum could increase learning efficiency if fewer tasks are necessary to learn complex tasks. This thesis distinguishes itself because it aims to expand on curriculum learning for dialog policy learning by accounting for task similarity with conversation graphs.

The structure of this thesis will be the following. First, in the Background section important terms mentioned in this proposal will be explained to the degree sufficient to understand the research. Then, in Related Work section we will examine work related to the field of task-oriented dialog systems. Starting with the general field of task-oriented dialog systems, and then narrowing down to this research's scope. In the Research Question section the research question and related sub-questions will be formulated. Following, in Method section the outline for the method of this research will be given. Subsequent in Results section the experiment results will be analysed. And lastly, in the Discussion section the meaning of the results will be discussed and related to answering the research question. This will also include possible directions for future work to take after this research.

2 Background

To adequately understand this thesis the following section will in detail describe how each part of the theory behind this research is interlinked. All the important concepts will first be broadly described and will be succeeded by a more formal and technical definition. This will start with an explanation of Task-Oriented Dialog Systems and how these are constructed, with a focus on Dialog Policy Learning (DPL). Next will follow an explanation of Reinforcement Learning (RL) and relevant sub-fields Q-learning and Deep Reinforcement Learning. After comes an explanation of Curriculum Learning (CL) and how this interacts learning with RL. Lastly, Graph Neural Networks will be explained and how this method can be used in CL.

2.1 Task-Oriented Dialog Systems

Task-Oriented Dialog Systems (Z. Zhang, Takanobu, et al., 2020) are dialog systems that are designed to assist users with completing domain-specific tasks through conversation. For example, one domain would be booking a spot at a restaurant. In this domain, tasks that the system can achieve would be booking a restaurant according to a user's specific preferences such as the preferred food, date, and the number of seats, etc. This collection of preferences is also called a user goal.

The main method of interaction with the user is through conversation in natural language. So the system needs to be able to interpret natural language and form a response in natural language. To this end, a task-oriented dialog system can be generally structured according to two different methods.

The first method is called end-to-end. Here the system is a single system that processes user input directly to natural language output. The process is natural language gets encoded into latent variables necessary to look into a knowledge base containing all the information of the domain. The latent variables then get decoded directly into the natural language output. The black-box nature of end-to-end models makes this method less stable and less controllable than the other method for constructing dialog systems (Gao et al., 2018).

In contrast to the first method, the second method is called the pipeline method. Here the system is modular, where each of its four components feeds into one another to form one complete dialog system. This is the method that is the focus of this research and consequently will be explained in further detail. The four components in a pipeline structure are Natural Language Understanding, Dialog State Tracking, Dialog Policy Learning, and Natural Language Generation. How these components are interlinked can be observed in figure 1, and in order their function will be described below.

First is the Natural Language Understanding (NLU) component that interprets user input. User input comes, in the context of a dialog system, in the form of written text. The natural language text is decoded into a semantic frame that consists of three essential aspects: a domain, intent and a number of semantic slots. Aspects of a semantic frame are commonly decoded simultaneously in a joint NLU model (Weld et al., 2022). The domain aspect covers the topic information of the text. For example, in the sentence "*Book a restaurant table for Tuesday.*" the extracted domain might be `restaurant`. The intent aspect encompasses what the speaker of the text aims to achieve. In the previous example sentence, the intent might be represented by `book-restaurant`, as in the goal of the speaker is to book a spot at a restaurant. The last aspect of the semantic frame is a collection of different semantic slots, which essentially translates select words or sentences into the

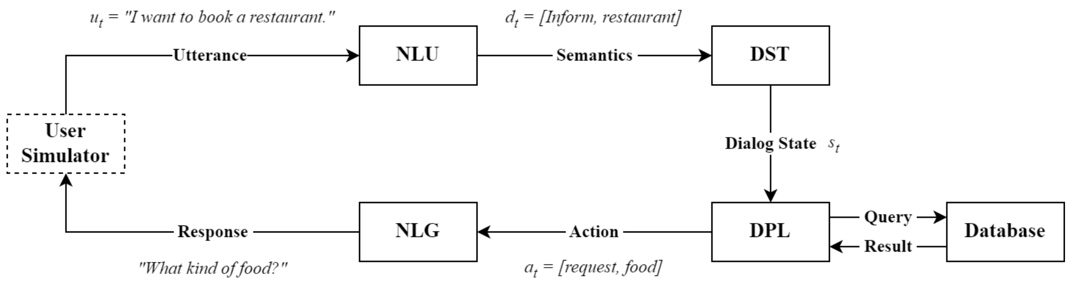


Figure 1: Overview of the components in a task-oriented dialog system with a pipeline structure.

semantic information that gives context to the extracted intent. In the previous example, this could be semantic slots such as the `time`, `date`, `location`, or `food-type`. In practice, the number of slots should be sufficient to interpret the relevant information for all possible domains and intentions. Historically, the decoding of dialog in the NLU component was realized with statistical models, however now it is most commonly achieved with machine learning models.

Second, the Dialog State Tracking (DST) component keeps track of the conversation history and estimates the dialog state (Williams et al., 2016), which makes the dialog state the representation of the dialog session till the present conversation step. The structure of a dialog state could consist of user preferences, an estimate of the user goal, and other information. As an example, in a restaurant domain, the user preferences could be restaurant rating and pricing. And an estimate of the user goal could be a list of restaurants that are suitable given the preferences. Seemingly simple, this task nevertheless poses a challenge given that NLU results can have inconsistent accuracy and that user miscommunication is always a factor. To tackle DST there generally exist three types of algorithms. The first makes use of hand-crafted rules. These rules are used to map one or multiple previous dialog states and NLU results to the current dialog state. Hand-crafted rules have the benefit of requiring no pre-existing real conversation data to operate. However, this also makes it so thoughtful formulation is necessary for the algorithm to produce accurate results. Other DST algorithms are generative models, which use Bayesian networks to predict the dialog state. In broad terms, the network uses as its input features like the system action, predicted user action, and NLU results in a probability calculation called Bayesian inference as an estimation of the current dialog state. The different probabilities can be learned from large amounts of conversation data. So, generative models more accurately represent dialog states in real conversations than rule-based models. Although, in Bayesian networks, the features are explicitly modelled, which is impractical for a large number of features. The last class of DST algorithms are discriminative models, which employ a trained classifier like logistic regression to score multiple dialog states and after pick the best-fitting current dialog state. Making such a model requires labelled conversation data for training, but when available can integrate a considerable number of features.

Third, the Dialog Policy Learning (DPL) component, which depending on the DST-produced current state decides the next system action. In the DPL, the process that produces the next system action is referred to as the dialog policy. In a task-oriented dialog system, the aim of these actions is mainly to further progress towards achieving the current user goal as estimated by the DST component. So, as an example, in a conversation with a user, the current determined user goal is `book-restaurant`, and in the dialog state, the required `time` slot for that restaurant reservation is unknown. The dialog policy will then determine that the best course of action is to ask for a preferred time, and therefore produce the action `inform-time`. The challenge for the dialog policy is then to correctly converse with a user based on the available information to the system. Depending on the size of the dialog state and the number of available actions it can be feasible to create the dialog policy as a hand-crafted rule-based system, similar to the DST component. However, like any hand-crafted rule-based system, it has the same drawbacks of being time-consuming to construct and inflexible. Because every chosen action leads to a new state, in essence, within one conversation the dialog policy is activated for every state in the sequence of states that is a conversation. Often then, DPL is formulated as a Reinforcement Learning (RL) problem because of a sequence of actions synergy with feedback. How this is defined will be further explored in the 2.2 subsection.

Finally, the Natural Language Generation (NLG) component translates the system action into a natural language response. For example, the system action `inform-time` in the domain `restaurant` could be translated by the NLG component into "*At what time would you like to reserve a table?*". Creating natural language with NLG is a challenge to fulfil the criteria of variation, adequacy, fluency, and readability (Stent et al., 2005). One prominent framework for natural language generation is sentence planning (Reiter & Dale, 1997). In this framework, it is first decided what information is important to communicate, and the order to do it in. The relevant information is then expressed in phrases. And lastly, the sentences are adjusted to improve readability and fit grammatically with each other. Naturally, each part of this framework can be approached with multiple methods (Gatt & Krahmer, 2018). NLG can also be approached as a global framework with neural networks. Here the model is trained on dialogs with LSTM models to work.

The interaction between these components can be seen in figure 1. Most industry-deployed models are pipeline models because of their modular structure and interpretability (Z. Zhang, Takanobu, et al., 2020). As mentioned earlier, the focus of this research project is on improving the learning efficiency of the DPL component of the dialog system. And because the modular structure of pipeline models allows for direct interfacing with this component, this project focuses on pipeline models.

Conversations with the dialog system take place in a variable amount of speaking turns t (Sacks et al., 1978). Each turn starts with a natural language user utterance $u_t = [x_1, ..., x_n]$ of n tokens x_i. In tasks such as movie-ticket booking, information about the preferences of the user is required for the dialog system to consult the domain-specific knowledge base for suitable suggestions. Acquired information values are stored in slots i when the given values are valid for the said slot. Slot-filling is thus the exercise of collecting these values. Slot-value recognition in an u_t is achieved by predicting the user intent class d with $p_{intent}(d|u_t)$ and predicting labels y_i for each token with $p_{s-v}(y_1, ..., y_n|u_t)$. And to keep track of the current dialog state can be estimated by $p_i(d_{i,t}|u_1, ..., u_t)$, which gives the class for each slot i on the t-th turn. Given the dialog state, the next required action is given by a policy π learnt through reinforcement learning. Finally, the system response is generated by mapping actions to natural language with a machine learning model.

2.2 Reinforcement Learning

Reinforcement learning (RL) Kaelbling et al., 1996 is a form of machine learning which is characterized by an intelligent agent that learns via trial and error within an environment. Much like how biological organisms learn to distinguish beneficial from harmful interactions within the real world based on their real-world consequences.

Formally, in RL the environment comprises a set of environment states S, and as for interaction within the environment, the agent has access to a set of possible actions A. Within this dynamic environment, the agent is asked to complete a certain task and reach some goal. This task is episodic, meaning that it occurs for a certain amount of time-steps t, and ends when a terminal state (i.e. the goal) or some time-step limit is reached. In the episodic loop, for each time-step t the agent chooses an action a_t based on the current state s_t. This episodic loop is illustrated in figure 2. The

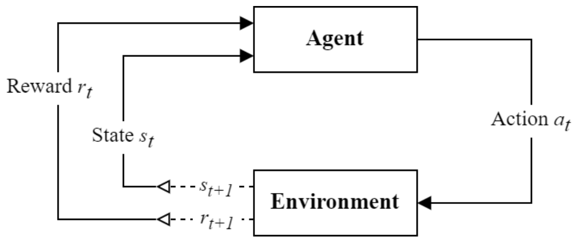


Figure 2: The episodic interaction loop of agent and environment of RL.

episodic interaction loop of RL can be modelled as a Markov Decision Process (MDP) because it contains the Markov property: the next state depends only on the current state and the available actions within. The MDP for RL can be defined as the 5-tuple (S, A, P, R, γ). The content of the set of states S and actions A was already established. Next in this definition is the state-transition function $P(s_{t+1}|s_t, a_t)$, which produces the next state s_{t+1} as a response to the current state s_t and action a_t. This state transition is not necessarily deterministic and can be a probability function. Then there is the reward function $R(s, a)$ which given some state s and action a synthesizes the corresponding reward r. Here, typically higher output indicates that an action led to getting closer to the current task goal. And lastly, there is the discount factor γ where $0 \leq \gamma < 1$. In the case of accumulative rewards, the discount factor reduces the value of future rewards. This prevents an infinite sum in infinite-horizon models where an agent can look to infinite future states.

In the intelligent agent, the function that decides which action a_t at the current state s_t is called the policy $\pi(a_t|s_t)$ function. A policy is therefore a map from states to actions which dictates the agents' behaviour, meaning that the performance of the agent is determined by the quality of its policy π. So, finding the optimal policy π^* is essentially the goal for an RL agent. One common approach is with model-free methods, where the reward function R and state transition function P are unknown to the agent.

In Temporal Difference (TD) learning (Sutton, 1988), the optimal policy is found by learning the value function $V^\pi(s)$. The value function expresses the predicted accumulative, discounted, future reward given a state s. In the TD algorithm the optimal value function $V^*(s)$ is learned by iteratively updating it according to the update rule $V(s) \leftarrow \alpha[r + \gamma V(s') - V(s)]$ where α is the learning rate. Updating the value function by estimating the value of a state through sampling the value of future states is called bootstrapping.

However, this is not the only practice for finding the optimal policy π^*. Other popular methods are called the state-action-reward-state-action (SARSA) algorithm and Q-learning (Watkins & Dayan, 1992). In both SARSA and Q-learning, the optimal policy is learned by learning the action-value function $Q^\pi(s, a)$, instead of the value function, of which the main distinction is that the action a is included.

To learn the optimal action-value function $Q^*(s, a)$, the SARSA algorithm has the update rule $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$, and similarly, the Q-learning algorithm has the update rule $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$. Within these update rules the fundamental difference between the two algorithms can be observed. The SARSA algorithm is on-policy by using $Q(s', a')$. In contrast, the Q-learning algorithm approximates Q^* directly with $\max_{a'} Q(s', a')$ as a greedy approach.

So far the above RL methods were examined in a general machine-learning context. However, reinforcement learning can be made specifically as deep reinforcement learning (Y. Li, 2017). In deep learning methods, models consist of multiple hidden layers between the input and output layers, where the output of each layer feeds into the next (LeCun et al., 2015). These layers allow for approximating different non-linear functions.

In the past deep learning for RL was quite unstable or divergent when using non-linear functions to approximate the action value function Q. This was until the introduction of the Deep Q-learning Network (DQN) (Mnih et al., 2015), which as the same suggests is a deep learning implementation of Q-learning. At the time the DQN implemented several novel techniques to solve the long-standing deep reinforcement learning instability.

The first cause of instability occurs when training RL on a sequence of environmental states where there are correlations present between those states. For example in video games, each frame of the video influences the next frame. Training on many similar states with gradient descent can lead to overfitting on those specific states. This was solved by the DQN with experience replay, which essentially records a cache of previously explored states, called the replay buffer. At each training step, from the replay buffer training samples are selected at random. Another cause of instability was the fact that updates to the Q function can significantly change the policy because bootstrapping changes the outcomes of both $Q(s, a)$ and $Q(s', a')$. To solve this the DQN only updates the Q function periodically. The last cause of instability was the correlations between the action-values and the target values.

2.3 Curriculum Learning

Curriculum Learning (CL) (Bengio et al., 2009; Narvekar et al., 2020; X. Wang et al., 2021) can generally be described as sequencing tasks in a way that benefits learning the overall task. In concept, this learning method can be compared to curricula present in all human learning; start with small and simple tasks and gradually build upon these to eventually arrive at more complex tasks. Most obvious is this in education, where it is structured to start with easier concepts to complex ones. For example, learning mathematics starts with basic operations such as addition and subtraction. And when we grasp these concepts, only then do we attempt to learn to solve algebraic formulas. However, this also extends to athletics. We don't start with the highest jump possible but work towards it. Not only humans benefit from curricula but animals too, where this training technique is called shaping (Skinner, 1958).

In the early 1990s, the first applications of curricula to the training of artificial intelligence can be found in grammar learning (Elman, 1993), and robotics control (Sanger, 1994). Conclusions of these early works confirmed that the task order of least to most difficult also applies to artificial intelligence.

Generally, a curriculum is implemented by giving certain tasks to the machine learning model by an external task selector. A predefined task order is the simplest form of task selection, which is an offline method. Online methods dynamically create a curriculum via task selectors. For example in co-learning methods like asymmetric self-play (Sukhbaatar et al., 2017), a teacher and student agents optimise a policy simultaneously. In this method, the teacher agent provides tasks to the student agent, which the student aims to solve in the least amount of actions, while the teacher tries to increase the number of actions it takes for the student to solve the task. Essentially, this cycle generates a curriculum with task difficulty. This method is similar to General Adversarial Networks (GAN).

Curriculum learning has applications in supervised learning (Guo et al., 2018; Tay et al., 2019), reinforcement learning (), and graph learning (Gong et al., 2019). In the context of this research, CL will be applied to RL, so the curriculum will be formally defined with that in mind.

Formally, a curriculum C can be defined as the directed acyclic graph (V, E, g, T), with the set of vertices V and set of directed edges E. It also contains the set of tasks T, wherein task m is defined as the four-tuple (S, A, p, r). Here, the task is structured like RL and comprises the set of states S, set of actions A, state transition function p, and reward function r. The function g that from vertices gives subsets of transitions in D^T, which is the set of all possible transitions (s, a, r, s') from the tasks in T. In this definition, tasks m and transitions contain the reward function r, which are the same definitions used in reinforcement learning.

Defining the curriculum as a graph is required for conceivable non-linear curricula, like graph curricula (Svetlik et al., 2017). In graph curricula, non-linearity can be achieved through dynamically selecting tasks or even by training on different tasks in parallel. However, in its most simple form, a curriculum can be expressed as a linear sequence $[m_1, m_2, ...m_n]$ of length n, where every task simply follows one another. The linear sequence curriculum is what will be utilized in this thesis, and the implemented approach to construct such a curriculum will be further explained in the Method section.

2.4 Graph Neural Network

Graph Neural Networks (GNN) (Z. Wu et al., 2020) are a category of end-to-end deep learning models that solve tasks involving graphs. This category is significantly different because unlike other neural networks such as Recurrent Neural Networks (RNN), instead of Euclidean data (e.g. images, text), as their input, GNNs embed graphs. In Euclidian space, the distance between two points can be calculated based on their position. In graphs, however, the distance between two nodes is not based on position but based on the connection to other nodes. The start of research into GNNs started with the first attempts of classifying graphs (Sperduti & Starita, 1997) using neural networks. Later this research was expanded upon and the first GNN (Gori et al., 2005) was established. This first GNN was an extension of the RNN architecture, which uses the previous output of the network layers to influence the present layers.

In general, a graph $G = (V, E)$ is defined as the set of nodes V, and the set of edges E. Typically graphs in the GNN are represented by an $n \times n$ adjacency matrix A where $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \in E$. Here $e_{ij} = (v_i, v_j) \in E$ denotes an edge between nodes $v_i, v_j \in V$. Then like any other deep learning model, the graph would process through possibly several hidden layers L, each with hidden nodes $h_v^{(l)}$ and a set of weights W. The output of a GNN can vary by focusing on the node-level, edge-level, or graph-level.

Any further details of the GNN architecture come down to the specific type of GNN. One commonly used type is the Graph Convolutional Network (GCN) (Kipf & Welling, 2016) distinguishes itself from other GNNs by the use of convolution, which refers to the technique of multiplying the inputs of a layer with a set of weights that also encompasses neighbouring inputs. In graphs, the neighbourhood of a node v is defined as $N(v) = \{u \in V | (v, u) \in E\}$.

3 Related Work

In this section, fields of research adjacent to this thesis will be traversed and examined to give context to this thesis. First will follow the methods of deep reinforcement learning for dialog policy learning. Second, various research which concerns itself with mapping conversations to graphs will be explored. Then will follow curriculum learning when it is applied to reinforcement learning methods for dialog policy learning in task-oriented dialog systems.

3.1 Deep Reinforcement Learning for Dialog Policy Learning

When it comes to dialog policy learning in task-oriented dialog systems, as mentioned in the Background section, various different methods have been designed in prior research. One of the prevalent approaches for dialog policy learning is reinforcement learning (Levin et al., 2000; Scheffler & Young, 2000). Within this category of approaches, there are again various methods (Kwan et al., 2023) of which the most relevant selection of these uses the DQN method, just like the model of this thesis. To get an overview of what this sub-field of DQN-based dialog policy learning contains, a brief overview of prominent works will be outlined.

One recent DPL model that uses a DQN is the Deep Dyna-Q (DDQ) (Peng et al., 2018), which traces back to dynamic programming (Sutton, 1990). In DDQ, conversations from a (hypothetically) real user with the dialog system simultaneously train the dialog policy and also construct a world model. In the planning phase, the world model generates a number of dialog states based on the recorded real user data. These generated dialog states are then used to evaluate the dialog system response action to that state and further train the dialog policy. Essentially, this world model and planning phase can be seen as the act of repeating different scenarios and learning which response yields the best result. This enables the DDQ to train more effectively on real user conversations without requiring more, and in practice expensive, real user data. The framework of DDQ is extended in the Discriminative DDQ (D3Q) (Su et al., 2018). The addition made in D3Q is the RNN-based discriminator component between the world model and the planning phase. The discriminator is to evaluate how close the generated dialog states of the world model look to the real user conversation and only allows the highest-quality dialog states to be used in the planning phase. And DDQ is also extended in the Switch-based Active DDQ (Switch-DDQ) (Y. Wu et al., 2019). The Switch-DDQ makes use of a switcher component that decides while learning the dialog policy whether it is beneficial to use the real user data or the generated dialog states of the world model. Similarly to DDQ, the Opposite Agent Awareness (OPPA) (Z. Zhang, Liao, et al., 2020) model uses a user estimator, trained from conversation data, to approximate the reaction of a user. This makes learning from real user conversation data more efficient.

An alternative DQN for dialog policy learning is the Bayes-by-Backprop Q-network (BBQN) (Lipton et al., 2018). In Bayes-by-Backprop (Blundell et al., 2015) the weights in a neural network are instead learned probability distributions. This can therefore quantify the uncertainty of a learned dialog policy, which can be used to guide the exploration of other dialog policies. The BBQN was shown to outperform the basic DQN in terms of success rate.

One of these is the HER-DQN (Lu et al., 2019) which is a DQN based on hindsight experience replay (HER) (Andrychowicz et al., 2017). Hindsight experience replay is a variation of the experience replay found in the DQN algorithm. In HER, the stored experience in the replay buffer includes the achieved states in that experience as opposed to strictly the agent's initial goal state. This allows for the stored state transitions of these experiences to give different rewards from just negative and therefore gives training with experiences sampled from the replay buffer more data to learn from, which is advantageous when learning a policy for large state spaces. The HER-DQN adapts this technique for dialog policy learning with two methods. One by trimming failed dialogs to create successful ones, and another by stitching dialog segments together into a successful dialog. Another variation combining dialog segments to construct HER for a DQN is the Learning with Hindsight, User modelling, and Adaptation (LHUA) method (Cao et al., 2020). Here, an additional adaptive coordinator is introduced to dynamically switch between real user dialog and simulated user dialog that is modelled from the real and simulated dialogs.

Other mentionable approaches to policy learning with a DQN are the Deep Q-learning from Demonstrations (DQfD) (Gordon-Hall, Gorinski, Lampouras, et al., 2020) and its extension RoFL

(Gordon-Hall, Gorinski, & Cohen, 2020). The Deep Advantage Actor-Critic (DA2C) (Fatemi et al., 2016) leverages actor-critic architecture to utilize training data efficiently. To tackle multi-domain dialog the Deep Transferable Q-network (DTQN) (Xu et al., 2020) has additional transfer learning on dialog acts or slots between different domains

In Kwan et al., 2022 the problem of large state-action space in dialog policy learning is mentioned. It states that, regardless of the method by which to facilitate reinforcement learning, because of its reward-based process and many explorable dialog states, it still takes time for a model to converge. Some recent works solve this problem by breaking up dialogs into sub-tasks, sub-domains, or sub-goals with hierarchical reinforcement learning (Budzianowski et al., 2017; Kristianto et al., 2018; Peng et al., 2017; Tang et al., 2018).

3.2 Conversation Graphs

Conversation graphs have seen use in prior research, especially when it comes to the analysis of conversations on various social network platforms. In Brambilla et al., 2022, conversation graphs are created based on Twitter messages. Here the messages are subjectively categorised to simplify the intent of a message. Another research (Zayats & Ostendorf, 2018) creates conversation graphs based on threaded replies in Reddit posts. The conversation graphs in combination with temporal data are used in a model to predict popular and controversial comments in a thread. Other research on threaded forum posts (Aumayr et al., 2011) does reconstruction of threads by generating conversation graphs. However, these conversation graphs differ from the graphs in this thesis because the structure of a thread is tree-like, meaning its branches don't connect to each other once they split off. This differs from the graphs described in the 5 section because these graphs all have one start and end node.

When it comes to the generation of conversation graphs, a mathematical model of conversation graph generation is developed in Kumar et al., 2010. This research falls under the field of random graph generation (Durrett, 2007). This is a different approach for graph generation compared to the use of a user simulator in this thesis. Given that the graphs of this thesis have one start and end node, it can also be seen as a graph of solutions to a problem than a conversation, with each node between the start and end node solving a problem one at a time.

3.3 Curriculum Learning for RL-based Dialog Policy Learning

In a similar method to dividing the learning process, in the reinforcement learning field advancements have been made by using transfer learning in a curriculum. With curriculum learning, the learning process is divided by changing when samples of the training set are encountered.

3.3.1 Experience level CL for RL-based dialog policy learning

One way to create such a curriculum is by utilizing the acquired experience from training an RL agent, and by using past training samples to support current learning. These methods derive a curriculum at the experience level with experience replay.

Experience replay sequences tasks by storing previous state-action-reward experience, and replaying the stored samples in the learning process. One way to produce these experience samples is by approximating the value function from a convolutional neural network (Mnih et al., 2015). In Prioritized Experience Replay (PER) all recorded experiences are ordered by some measure of importance and then replayed in that order, essentially creating a curriculum. For example in Schaul et al., 2015 samples were ordered with a temporal difference (TD) error measure, which indicates the unexpectedness of a transition. In Ren et al., 2018 a Complexity Index (CI) measure was used instead. An alternative approach to experience replay is Hindsight Experience Replay (HER). Here samples are structured based on any goal an agent reaches, not strictly from the agent's initial goal (Andrychowicz et al., 2017). However, in the standard version of HER, experience samples are not prioritized in terms of usefulness to the learning process, which can negatively impact learning efficiency. This was improved upon by Fang et al., 2019 through the dynamic selection of samples based on diverse goal exploration, and proximity to the initial goal. Similarly, in Nguyen et al., 2019, the rank of goals in experience samples is adjusted with a threshold. A goal's rank is decreased if the distance of a goal to the initial goal falls below this threshold.

Sequencing can also be done through co-learning, which applies multi-agent interaction. In Sukhbaatar et al., 2017 this method is adopted in so-called self-play, where one agent sets a goal for another equal agent that then tries to achieve this goal. These two agents can also be described as teacher and student. The sequencing of tasks occurs via different individual internal rewards. Here, the teacher is rewarded by increasing the time of the student's successful task completion, and the student is rewarded by efficiently solving a task. And this creates a feedback loop of increasing task complexity. These outlined PER, HER, and co-learning methods are all forms of an implicit curriculum. In these methods, the task sequence is embedded within the model via changing task selection, or multiple improving agents, without any explicit task categorization based on task complexity.

3.3.2 Task level CL for RL-based dialog policy learning

It is also possible to set up an explicit curriculum by sequencing tasks before training, in a different method called curriculum learning (Bengio et al., 2009; Narvekar et al., 2020). The idea of a curriculum is inspired by human learning; to start with small and simple tasks and gradually build upon these to eventually arrive at more complex tasks. For example, in education, learning mathematics starts with basic operations such as addition and subtraction. And when we grasp these concepts, only then do we attempt to learn to solve algebraic formulas. In the early 1990s, the first applications of curriculum learning to machine learning can be found in grammar learning (Elman, 1993), and robotics control (Sanger, 1994). The conclusions of these early works confirmed that the task order of least to most difficult also applies to artificial agents.

Because building a sufficiently large curriculum for machine learning can be time-consuming, creating a curriculum is often done with curriculum generation. Algorithm for curriculum generation order tasks automatically in a linear sequence (Narvekar et al., 2017), or even into curriculum graphs (Svetlik et al., 2017). In Narvekar et al., 2017 the automatic sequencing is achieved through a recursive algorithm that, for a target task, tries to solve the task and update its policy within a certain learning budget and threshold. After a successful solution, the task is put into the curriculum and the policy is updated. If the task is not solvable, this step gets recursively repeated with a simpler task. And because the learning budget is increased each time, eventually the algorithm terminates when the target task is solved within that budget. While other methods for curriculum do exist, the advantage of sequencing methods is that they do not alter the environment of the learning agent, and have therefore been successful across many fields.

Recently curriculum learning was used with RL-based dialog policy learning in Zhao et al., 2021. Here the focus is more on stability instead of efficiency. In this approach, additional reinforcement learning is performed between the dialog agent and teacher model, which in turn decides the curriculum schedule, limited by an over-repetition discriminator. The teacher model primarily constructs the schedule, with only light automatic dialog complexity evaluation. Therefore it differs from curriculum learning which is based on just complexity evaluation. Additionally, the automatic dialog complexity that is there is based on the number of interactions within a user goal, and not done with dialog graphs.

Another approach to curriculum learning in RL-based dialog policy learning is Liu et al., 2021. Here, only automatic dialog complexity evaluation generates the curriculum. Calculation of the dialog complexity is based on dialog state differential space, where differential space indicates the error between the actual next dialog state and the predicted state by a neural network. The automatic curriculum generation aspect of this method for dialog policy learning is what we aim to improve.

4 Research Question

To improve upon and build on previous work, in this thesis, curriculum generation will be approached by representing user goals in conversation graphs. In a conversation graph, dialogs that share the same user goal are grouped together. This is possible because every dialog starts from nothing and so all share the same start dialog state, therefore the start of each conversation graph can also be the same. And the end dialog state is also the same for multiple dialogs, because it represents fulfilling the relevant user goal. An example dialog graph can be seen in figure 5, of which the particulars are defined in the Method section. Such a collection of dialogs is called a topic. The reasoning for using a graph arrangement is that the branching structure of graphs should better represent human dialogs, as some human conversations achieve similar goals while taking different and deviating paths to get there. Often conversations on the same topic also have common elements, which can be used in generating conversation graphs by compression of identical dialog states. In curriculum learning, conversation graphs could aid the learning process by grouping similar graphs together and therefore decreasing the time between similar user goals occurring in dialog policy learning, which can lead to quicker learning on topics that have a large variation in complexity. Conversation graphs could make it possible to reduce redundant repetition when training a model by eliminating training samples that are too similar, although this is not explored in this thesis.

In summary, in this thesis, the assumption is that with conversation graphs, dialog policy learning with a curriculum could be made more efficient because it should lead to better sequencing of user goals. This project expands curriculum learning methods for dialog policy learning by changing the user goal representation to graphs. Therefore, the investigation of how the idea of a conversation graph-based curriculum can help RL-based dialog policy models to learn complicated tasks faster is a core objective. And so, the research question and related sub-questions addressed in this proposal and answered by the following work will be formulated as follows:

- *How can the efficiency of reinforcement learning in task-oriented dialog policy learning be increased using conversation graph-based curriculum learning?*

    1. *How can an effective conversation graph for dialogs be created?*
    2. *How do conversation graphs impact curriculum generation?*
    3. *How does graph neural network-generated curriculum learning influence learning efficiency?*

5 Method

In this section the methodology, as illustrated in figure 3, will be explained and how this relates to answering the research question. First, as groundwork, the task-oriented dialog system code-base that

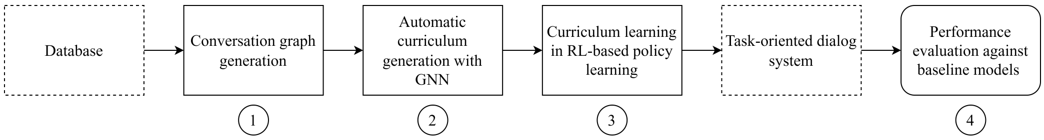


Figure 3: Breakdown of the necessary steps to achieve the CL-based dialog policy learning method of this thesis.

is used will be based on the open-source ConvLab-2 framework (Zhu et al., 2020)[1], which includes code for the user simulator and every pipeline model component of a task-oriented dialog system can be independently configured. In the method figure 3, the dashed box labelled "database" represents the dataset for the task-oriented dialog system. For the dataset, the widely used MultiWOZ 2.1 dataset (Eric et al., 2019)[2] is employed, which is used to evaluate task-oriented dialog systems and is also used in curriculum learning-related work that this research aims to improve upon (Liu et al., 2021). The MultiWOZ 2.1 dataset includes over 10.000 dialogs that span 7 different domains. The rest of the steps in the figure 3 follow. Step 1 starts with generating conversation graphs that form the basis for creating a curriculum. What this entails is explained further in section 5.1 Afterwards, step 2, in section 5.2 and section 5.3, explains how a GNN is utilized to generate the static linear sequence of tasks that is the curriculum. And last, for step 3 the section 5.4 explains how the curriculum learning is integrated with the reinforcement learning algorithm.

We expect that following this methodology, the resulting answers to the sub-questions will lead to an answer to the research question. The answer to sub-question 1, finding an effective conversation graph, will be produced by the method that follows in the current method section and investigating the quality of the resulting conversation graphs. The solution for sub-question 2, regarding the impact of graph representations on curriculum learning generation, will be established by evaluating the quality of the curriculum in terms of sorting the user goals from least to most complex. Another aspect of evaluating the generation method is how practical this method is to use. And for the solution to sub-question 3 comes from the comparison between the dialog policy learning rate and that of a base DQN model that does not utilize curriculum learning.

5.1 Conversation graph generation

This section will describe the method for generating dialog graphs. First, the dialog graph will be informally defined. Second, the method used to establish the nodes in this graph by specifying their node labels. And third, how the graph comes together by utilizing the task-oriented dialog system for the generation of conversations.

5.1.1 Conversation graph definition

Task-oriented dialog system conversations can be mapped to a conversation graph with the dialog state p_i and actions a for all speaking turns t in a conversation. To represent this definition of conversation in a graph structure, an intuitive idea might be to create a node for every state and label an edge with the turn between the states. However, the way graphs are encoded in the GNN edge labels is not included, which makes this representation incompatible. Instead, each turn and state in a conversation is defined in the graph structure as a state-action node. Edges then represent the order of each state-action. This description of a conversation graph will now be formally defined.

[1] `https://github.com/thu-coai/ConvLab-2`
[2] `https://github.com/budzianowski/multiwoz`

Definition. A conversation graph $G = (V, E)$ with the set of nodes V and set of edges E. Let $v_i \in V$ represent a node and $e_{ij} \in E$ represent an edge between nodes v_i and v_j. In the $n \times n$ adjacency matrix A, $A_{ij} = 1$ when $e_{ij} \in E$ and $A_{ij} = 0$ when $e_{ij} \notin E$. For every turn t in a conversation of length T, the function $\phi : S \to R$ maps each state-action $s_t \in S^T$ (formed of the dialog state p_t and system action a_t) to a node label $x \in R^n$. All nodes $v_i \in V$ have unique node labels $x_v \in R^n$, which can be formulated in the expression $\forall v_i \forall v_j((v_i \neq v_j) \to (x_{v_i} \neq x_{v_j}))$. And for every turn t, unique nodes v_t and v_{t+1} are connected with the edge $e_{t,t+1}$. For every conversation stored in the graph, the start node $v_{t=0}$ and end node $v_{t=T}$ are equal for all conversations.

5.1.2 Node labels

In a GNN, all node labels in the dataset are expressed in a one-hot encoding vector, meaning that every unique label is depicted in binary as 1 if the state is present, and 0 otherwise. As a result for every increase in labels, the vector also expands. This is however not desirable for a GNN as this makes the amount of learning increase. This can make the model less accurate. Another issue with this is, is that the ideal training data needs to have graphs with the same amount of unique node labels. However, as explained earlier in the justification for using a GNN to compute graph similarity, for large graphs other methods of computing similarity becomes not practical due to computing time. This makes the creation of a dataset, say with synthetic data, also impractical. So, the one challenge with node labels is the number of unique instances in the action space and state space. To solve this a custom method was devised for reducing the number of node labels. This had to be custom-made for the chosen MultiWoz dataset, as there seems to be no established solution in existing research. Reducing the number of node labels is unavoidably going to result in a less detailed expression of the dialog state-action, and will make the conversation graphs simpler.

In the MultiWoz dataset, the dialog state-action consists of the following categories: `user_action`, `system_action`, `belief_state`, `book`, `degree`, and `terminated`. First, the categories get limited to only the `system_action`. This removes most information about the dialog state, however, from the `system_action` components of the state can be implicitly inferred. For example, occurrences of the `'Book'` action imply the number of slots and how many are filled. Additionally, the main goal of conversation graphs is to find how difficult certain user goals are to achieve, which is expressed by how many actions the system has to take to get there. In the `system_action`, its components take context. So, for example, one action can simultaneously `'Recommend'` something for the `'Hotel'` domain, and `'Inform'` about the `'Train'` domain. While informative, in the case where multiple `'Recommend'` actions happen in different domains it leads to redundancy as now `'Recommend'` is recorded twice in the dialog state. Instead of taking into consideration all possible combinations, the components will be represented in a one-hot encoded vector. This leads to a vector with 51 components. Next, remove the slot-specific data for intentions in a domain, like `'Destination'` for the intent `'Inform'` in the `'Taxi'` domain, which reduced the vector to 23 components.

Next, the domain data was removed. This seems counter-intuitive when dealing with a multi-domain dataset, but in practice, each domain is paired with an intention. So when there are multiple instances of `'Request'`, then multiple domains are likely part of this user goal. Therefore we can remove the domain category without much impact on the graph structure. Now there are 14 components.

After the node label reduction, each node label has 14 categories, meaning $16,384$ possible dialog states. In practice, this is less because in most natural conversations not every topic of conversation is discussed in one act, and not every topic is appropriate to be discussed next to each other. Lastly, one last optimisation was made by combining all dialog states where `'terminated'` was set as true. This effectively removes the specific actions `'greet'`, `'welcome'`, and `'thank'` and

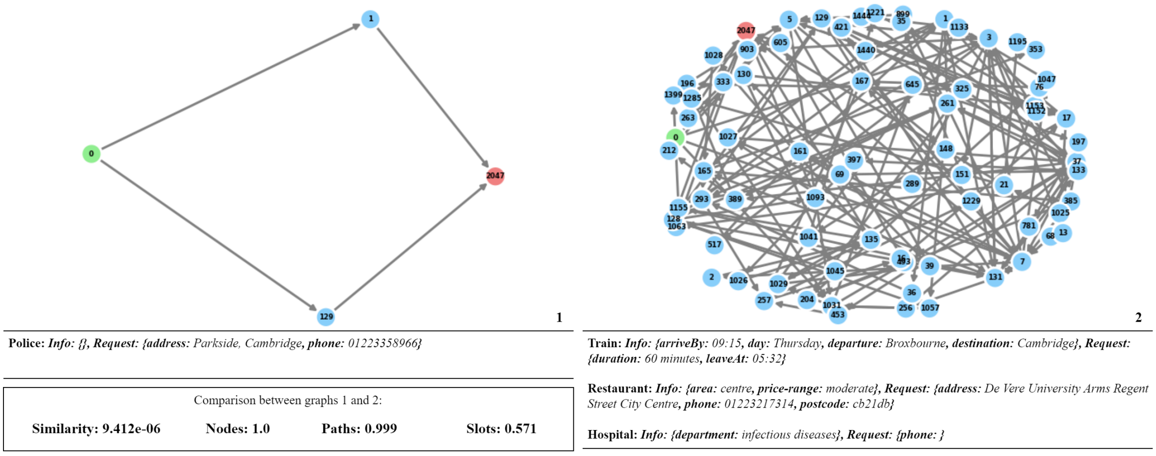


Figure 4: Comparison of the most dissimilar conversation graphs and user goals according to the SimGNN model.

other combinations where the user had one final request when terminating the dialog, because these were only used to give variation to the terminating message of dialogs. So, as a result, for the 795 generated graphs there were only 468 unique node labels. While this is still a lot of labels, it would be way more without a reduction because the number of nodes would increase too. In figure 4 it can be seen that even with this reduction in node labels there are nevertheless a lot of unique nodes in conversation graphs that have more complex user goals.

By using this kind of node label reduction there is clearly a less detailed expression of the state-action in the conversation graphs represented. Another issue is that custom node label reduction like this is in most cases not transferable between datasets. Meaning that with the use of each new dataset, depending on the number of permutations of the corresponding state-action, a new custom node label reduction method needs to be devised and tested.

5.1.3 Conversation generation

To create a conversation graph like in figure 5 for one single user goal, 10 different successful dialogs were generated with a task-oriented dialog system. The technique that we will be using for generating dialogs for the graph is utilising the user simulator and an existing (pre-trained) dialog policy. It is necessary to have a dialog policy that has enough variance such that not every dialog is exactly the same while being consistently successful in achieving the desired user goal. For example, a rule-based policy does not satisfy these criteria, because this policy always follows the specific actions defined by its included rules. Because these rules are constructed manually, generally, there is not a sufficient number of rules to generate enough significantly different conversations, as was the case with the rule-based policy for MultiWoz. Not every pre-trained policy, like a DQN model, is guaranteed to work either because the policy could be inconsistent. To this end, a policy obtained with Proximal Policy Optimization (PPO) (Schulman et al., 2017) is used. The PPO Policy action selection was slightly changed by enabling sampling with a multinomial distribution, instead of always taking the most probable action. The selection randomisation was necessary to get the varied dialogs required to create the conversation graph. The user and system agent used a rule-based policy.

A side effect of using a model to generate these conversation graphs is that they are not always

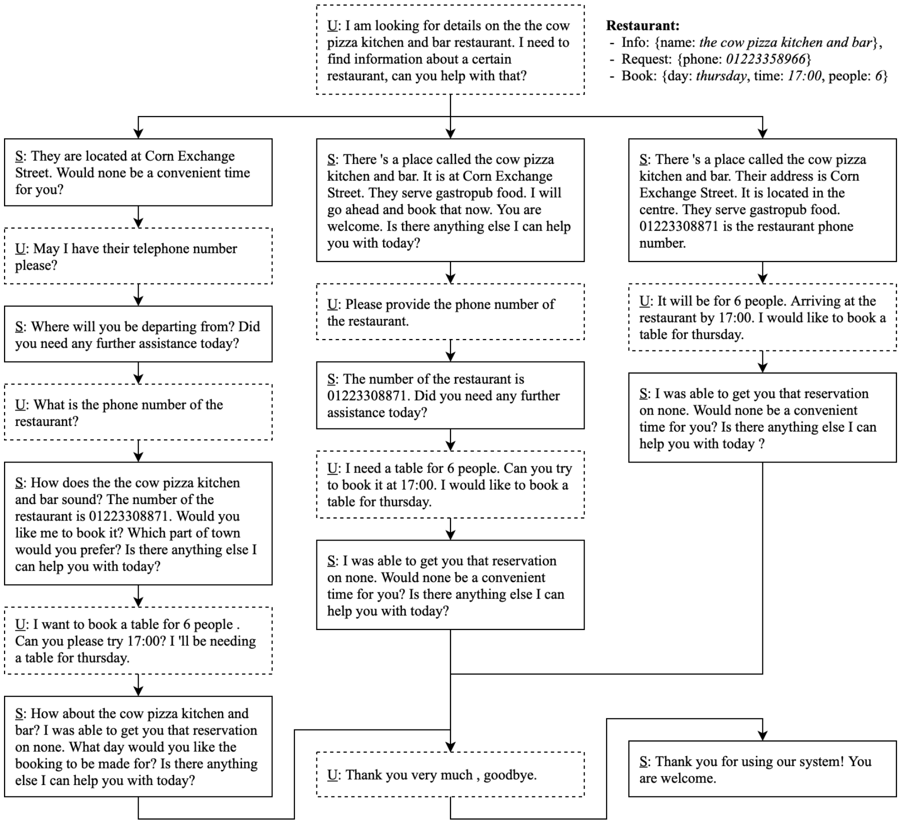


Figure 5: Three different conversations produced by one dialog policy. User utterances are indicated by the dotted lines, and the system responses by the solid lines. The user goal is shown next to the first node.

guaranteed to generate a successful dialog. And it is important for each dialog to be successful, otherwise, it can't be seen how difficult it is to accomplish a used goal. To give the model some room to fail a threshold is put in place of 10 attempts. After this limit is reached the goal is skipped. With the chosen method only 795 goals generated conversation graphs.

Another crucial aspect of generating good conversations is to remove repeated state-actions. These occur because the simulation can sometimes loop the action for the same state. This does not say much about the difficulty of the goal than it does about the quality of the policy models used for dialog generation, so repeating dialog action-states are compressed into only one state-action. Each edge is also weighed by the frequency of the action-state transition, although this is not factored in when determining graph similarity because of incompatibility with most available training datasets. In figure 5 an example of three different conversation paths taken from a generated conversation graph can be found. Each of the paths naturally corresponds to one simulated conversation. In the first conversation path it can be observed that the PPO dialog policy is not without its mistakes, as the dialog system failed to produce the phone number of the restaurant.

Algorithm 1 Conversation graph generation

Require: $G \leftarrow$ empty directed graph
$S \leftarrow$ dialog session
$n_{\text{attempts}} = 0$, $H \leftarrow$ empty array

for $g = 0$ to 10 **do**
  while $r_t < 2$ **do**
    for $t = 0$ to 40 **do**
      $u_t, o_t, s_t, r_t \leftarrow S.\text{next\_turn}(u_t)$
      Store s_t in H
    if $r_t < 2$ **then**
      $n_{\text{attempts}} = n_{\text{attempts}} + 1$
      $H \leftarrow$ empty array
      S.reset
    if $n_{\text{attempts}} > 10$ **then**
      S.terminate
for s_t, s_{t+1} in H **do**
  $G.\text{add\_edge}(s_t, s_{t+1})$

5.2 Conversation graph similarity

The following section is about how the dialog graphs are utilised to generate a curriculum. First, the detail of the GNN model is used to measure the similarity between the dialog graphs. Second, the other measures were utilised to classify the graphs. And last, how this comes together in a curriculum.

5.2.1 Graph neural network

The similarity between graphs can be computed by a variety of different methods. However, methods like Graph Edit Distance (GED) are computationally expensive and not practical with large graphs like conversation graphs. Therefore, a Graph Neural Network (GNN) (Z. Wu et al., 2020) will be used since with this the similarity between embedding graphs (Y. Li et al., 2019; Ma et al., 2021) can be learned, and can therefore group similar conversation graphs together. Pairwise similarity scores between graphs are determined by the SimGNN (Bai et al., 2019)[3] model, which approximates the GED between graphs. The codebase used for training a SimGNN model was an adaptation[4] of the at the time of writing most popular PyTorch SimGNN implementation[5].

Modifications were made to the graph node label one-hot encoding of the SimGNN model in order to make it compatible with the conversation graphs which have more node labels than the data set it was eventually trained on. The on-hot encoding vector length was 468 for the 795 generated graphs. The final model was trained for 1000 epochs with a 128 batch size, 0.5 dropout, 0.001 learning rate, and 5×10^{-4} weight decay. The neural network GCN convolution layers (Kipf & Welling, 2016) were set to sizes 64, 32, and 16.

The distance metric GED is defined as the number of transformations required for one graph to morph into another graph. Transformations include the deletion, insertion, or relabeling of edges and nodes. In SimGNN the GED is normalised to be a value from 0.0 to 1.0. Therefore the predicted similarity score by the model acts as a normalised distance measure where a score of 0.0 means the graphs are equal, and 1.0 means the graphs are not alike at all.

[3] `https://github.com/benedekrozemberczki/SimGNN`
[4] `https://github.com/gospodima/Extended-SimGNN`
[5] `https://github.com/benedekrozemberczki/SimGNN`

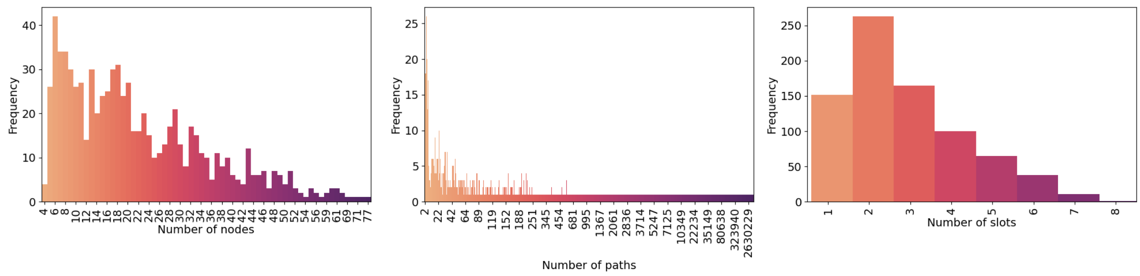


Figure 6: Distribution of the nodes, simple paths, and slots complexity values across all user goals.

When SimGNN (Bai et al., 2019) was introduced its performance was tested on three different datasets. So, to evaluate the impact the modifications to the one-hot encoding vector had on performance all these different datasets were attempted. First is the AIDS dataset which consists of 700 graphs randomly selected from the AIDS Antiviral Screen Data from the NCI/NIH Developmental Therapeutics Program. Graphs represent the chemical structure data from $42,687$ compounds. There are 29 different node labels across all graphs. Second, the LINUX dataset of 1000 graphs was randomly selected from LINUX Program Dependence Graphs that represent the relation between $48,747$ Linux kernels. And third, the IMDB dataset, which consists of all 1500 graphs found in the IMDB-MULTI dataset. These graphs represent if actors/actresses starred in the same movie. For each of these datasets, the GED label for each pair of graphs was approximated with an A algorithm.

5.3 Curriculum generation

5.3.1 Complexity metrics

To order user goals on complexity/difficulty three different complexity metrics were constructed. Two of these metrics are related to the generated conversation graphs $G = (V, E)$. The first of these metrics is the number of nodes in a dialog graph. The objective of this metric is to quantify the number of turns needed to achieve the user goal. For one graph this metric can be represented by the equation below.

$$\text{complexity}(G) = |V|$$

The second metric is the number of simple paths, from the start node $v_0 \in V$ to the end node $v_{|V|} \in V$ in the conversation graph G. A simple path is defined as a linear trajectory from one node to another along connecting edges which does not have repeating nodes. The purpose of the simple paths metric is to infer the conversation diversity for one user goal. Finding all simple paths from one node to another is achieved using a modified recursive algorithm adapted from Sedgewick, 2001. To compute the complexity it is then just a matter of counting the paths found by this algorithm. As such, for one graph this metric can be represented by the equation below.

$$\text{complexity}(G) = |\text{all\_simple\_paths}(G, v_0, v_{|V|})|$$

The third metric is the number of slots in the user goal, of which the intent is to represent the number of turns and conversation diversity of the goal. For example, when there are many slots, a lot of user preference information needs to be requested to achieve the user goal, and this could therefore lead to many turns. And in the case of many slots, it also implies more diversity, because filling each slot can be done in any order. Therefore, this metric can be represented by the equation below.

$$\text{complexity}(x) = |x_{\text{slots}}|$$

Each metric is computed and put into a distance matrix based on the pairwise Manhattan distance. This matrix is then normalized by dividing by the largest distance.

5.3.2 Curriculum generation

For curriculum generation, the aim is to minimise the difference in similarity and complexity between consecutive goals in the curriculum while beginning with simple user goals and ending with complex ones. As described earlier in a previous section, the trained SimGNN model predicts the similarity between the user goals from the corresponding conversation graphs, which are stored in a distance matrix. To group similar goals together, this distance matrix is used as the input for a clustering algorithm. An already computed distance matrix means some clustering methods (e.g. k-means) can't be used because these rely on computing the distance at run time. For this reason, the K-Mediods (KM) clustering algorithm (Park & Jun, 2009) is utilised because it can handle a distance matrix. The KM algorithm also requires the specification of the number of clusters k the data will be clustered into. In experimentation of the best value for parameter k, values between 2 and 10 were explored. The best value for k is determined by the quality of the clusters and therefore requires a scoring algorithm.

Evaluating the quality of clusters can be done using external evaluation or internal evaluation. External evaluation requires available class labels or another indicator that associates each data point to a cluster. This evaluation method is therefore not relevant because this information does not exist for the conversation graph data. Internal evaluation generally scores by prioritizing high similarity within all clusters and low similarity between clusters.

Three different scoring methods were tried: the Silhouette coefficient, the Calinski-Harabasz index (Caliński & Harabasz, 1974), and the Davies-Bouldin index (Davies & Bouldin, 1979). With these results, the best value for k per clustering method is selected. The Davies-Bouldin index and Silhouette coefficient produced the same best value for $k = 3$ and consequently the same clusters. However, the Calinski-Harabasz index produced a different result with the value $k = 6$. For both cluster configurations, the characteristics of the resulting curriculum are evaluated their performance is tested in the Results section.

User goals within the clusters are ordered by their complexity. To determine user goal complexity, three different metrics were considered. The number of slots in the goal, the number of nodes in the conversation graph, and the number of simple paths in the conversation graph. After examining the range of values that each complexity metric encapsulates in figure 6. , the simple paths metric was chosen because it expressed user goal complexity in the greatest range of values. The greater variance of complexity values allows for sorting user goals with more granularity. Therefore the paths metric is used to order the user goals. In algorithm 2 the pseudocode for the curriculum generation with clusters method can be found.

Algorithm 2 Cluster curriculum generation

Require: X a $n \times n$ distance matrix
A a length n complexity metric array
K of values for k to compare

$S \leftarrow S_2, \ldots, S_K$
for $k = 2$ to K **do**
    $y \leftarrow \texttt{cluster}(X, k)$
    $S_k \leftarrow \texttt{score}(\mathrm{X}, y)$
$k \leftarrow \texttt{max}(S_2, \ldots, S_K)$
$y \leftarrow \texttt{cluster}(X, k)$
$C \leftarrow c_0, \ldots, c_k$
for $i = 0$ to $\texttt{length}(y)$ **do**
    Store i in c_{y_i}
$V \leftarrow v_0, \ldots, v_k$
for $i = 0$ to k **do**
    $v_i \leftarrow 0$
    $F_k \leftarrow f_0, \ldots, f_{\texttt{length}(c_i)}$
    for j in c_i **do**
        $v_i \leftarrow v_i + A_j$
        Store A_j in F_k
    $c_i \leftarrow$ sort c_i according to F_k
$C \leftarrow$ sort C according to V
return C

Next, the obvious procedure is to order these clusters according to their average complexity. Ensuring that goals in easier clusters are learned first. However, this removes all complexity variation. So, we can also interleave the goals from each cluster, which keeps some variance. As described in algorithm 3 after the clusters are ordered, from each cluster, a goal can be picked and placed into the curriculum in every iteration. This results in an order of goals from three clusters in the sequence $[c_1, c_2, c_3, c_1, c_2, c_3, \ldots]$. Both configurations were used in experiments of which the outcome can be found in the Results section. the number of nodes.

Algorithm 3 Interleaved curriculum

Require: C_{ordered}

$C_{\text{interleaved}} \leftarrow$ empty
$m \leftarrow \texttt{max}(\texttt{length}(c) \text{ in } C)$
for $i = 0$ to m **do**
    for c in C **do**
        if $i < \texttt{length}(c)$ **then**
            Store c_i in $C_{\text{interleaved}}$
    return $C_{\text{interleaved}}$

5.4 Reinforcement Learning with a Curriculum

In this section, the reinforcement learning process of learning the optimal dialog policy with a curriculum will be discussed. The curriculum is integrated with the base for reinforcement learning in the task-oriented dialog system is a standard DQN model, of which a general detailed description can be found in the Background section.

Algorithm 4 shows pseudocode for curriculum learning. On line 3 of this algorithm, it can be seen that the curriculum learning process happens for E epochs. And the code between lines 4 and 7 shows that each epoch corresponds to a user goal g_e. Where this user goal g_e is sampled from depends on if the current epoch number e is below the curriculum size N, because the user goal g_e is sampled from the curriculum C at position e. So, when e exceeds N the user goal is sampled from the failed user goal list F at position $e - N$. With the user goal, the DQN-based reinforcement learning process starts. The user goal naturally affects the user simulator responses to the system, and also determines if the dialog was successful in reaching this goal. This part is covered further below, and pseudocode can be found in algorithm 5. The output of the DQN is the goal success state f, which is `true` if the goal has been achieved and `false` otherwise. The goal success state determines whether or not the goal is added to the failed used goal list F. Every 10 epochs the dialog policy is tested on a random sample of 50 user goals.

Algorithm 4 Curriculum Learning

1: $C \leftarrow$ curriculum of size N
2: $F \leftarrow$ empty array representing failed user goals
3: **for** epoch $e = 1$ to E **do**
4:     **if** $e < N$ **then**
5:         Sample g_e from C
6:     **else**
7:         Sample g_e from F
8:     $f \leftarrow$ DQN(g_e)
9:     **if** f is true **then**
10:         Store g_e in F

In algorithm 5 is the pseudocode for the DQN learning process, adapted from the DQN algorithm in Mnih et al., 2015. On line 4 of this algorithm, it can be seen that the entire Q-learning process happens over M episodes. Each episode corresponds to one dialog between the dialog system and the user simulator. For each of these episodes, the first state s_1 gets initialised as a series of x_1. The next part of the algorithm, starting from line 6, happens for T timesteps. The amount of timesteps corresponds to the amount of turns a single dialog takes. Between lines 7 and 10, the action a_t is selected with the ϵ-greedy policy. Meaning that with some probability ϵ a random action is chosen, otherwise, the action that by the action-value function Q is estimated to yield the most reward is chosen.

Then the dialog system takes action a_t, resulting in receiving the corresponding reward r_t and observing the user simulator response x_{t+1}. In turn, response x_{t+1} in combination with the chosen action a_t and current state s_t, determines the next state s_{t+1} of the environment. After this process, the experienced state transition (s_t, a_t, r_t, s_{t+1}) is saved to the replay buffer D. The replay buffer only stores the last N experiences. Between lines 15 and 20 the experience replay part is present. Here a random number of values j are generated which are used to sample the stored experiences in the experience buffer D. Line 21 is the periodic update of the target action value function

Algorithm 5 Deep Q-Network

Require: User goal Data

$D \leftarrow$ memory replay buffer of size N
$Q \leftarrow$ action-value function with random weights θ
$\hat{Q} \leftarrow$ target action-value function with random weights θ^-
for episode = 1 to M **do**
  $s_1 \leftarrow \{x_1\}$
  for $t = 1$ to T **do**
    if with probability ϵ **then**
      $a_t \leftarrow$ random action
    else
      $a_t \leftarrow \arg\max_a Q(s_t, a; \theta)$
    Take action a_i, observe r_t and user simulator response x_{t+1}
    $s_{t+1} \leftarrow s_t, a_t, x_{t+1}$
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    for each random j in minibatch **do**
      Sample of transitions (s_j, a_j, r_j, s_{j+1}) from D
      if episode at step $j + 1$ terminates **then**
        $y_j \leftarrow r_j$
      else
        $y_i \leftarrow r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a_j; \theta^-)$
      Gradient descent step on $(y_j - Q(s_j), a_j; \theta)^2$
    Reset weights $Q \leftarrow \hat{Q}$ every C steps

6 Results

In this section, some results of the 5 section, and the results of experiments performed with the curriculum-enhanced dialog policy learning will be discussed. First from the 5 section, the modified SimGNN performance will be evaluated and compared to the performance out of Bai et al., 2019. Next from the 5 section is the evaluation of the generated curriculum. After will be the results of four experiments that were conducted with the curriculum-enhanced task-oriented dialog system by changing the characteristics of the generated curriculum. And lastly, the best-performing curriculum configuration will be compared against a baseline DQN model which randomly samples user goals.

6.1 SimGNN

Model	**Dataset**	**mse(10^{-3})**	ρ	τ	**p@10**	**p@20**
SimGNN*	**AIDS**	2.950	0.820	0.642	0.365	0.460
SimGNN*	**LINUX**	1.802	0.966	**0.844**	0.938	0.932
SimGNN*	**IMDB**	**0.899**	0.866	0.742	**0.824**	**0.818**
SimGNN	**AIDS**	**1.189**	**0.843**	**0.690**	**0.421**	**0.516**
SimGNN	**LINUX**	**1.509**	**0.939**	0.830	**0.942**	**0.933**
SimGNN	**IMDB**	1.264	**0.878**	**0.770**	0.759	0.777

Table 1: Performance of the Modified SimGNN (indicated by *) compared to the reference SimGNN on the AIDS, LINUX, and IMDB datasets. For each dataset, the best performance is highlighted in bold.

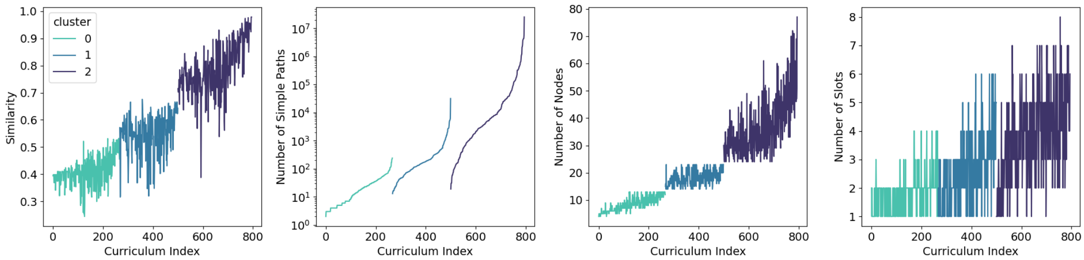
As mentioned in referred to in the 5 section, the modified SimGNN model was trained on three different datasets. These were the AIDS, LINUX, and IMDB datasets. The performance of this modified SimGNN can be found in table 1, and is here compared to the results of the SimGNN model presented by Bai et al., 2019.

Four different metrics were deployed to evaluate the performance of both SimGNN models. Of these, the first metric is the Mean Squared Error (mse), which measures the average squared difference between the model output similarity score and the actual GED value. To measure the relationship between the predicted and actual values, two rank correlation metrics Spearman's Rank Correlation Coefficient (ρ) and Kendall's Rank Correlation Coefficient (τ). Lastly, the Precision at k (p@k) metric gives the average precision of the top k predictions.

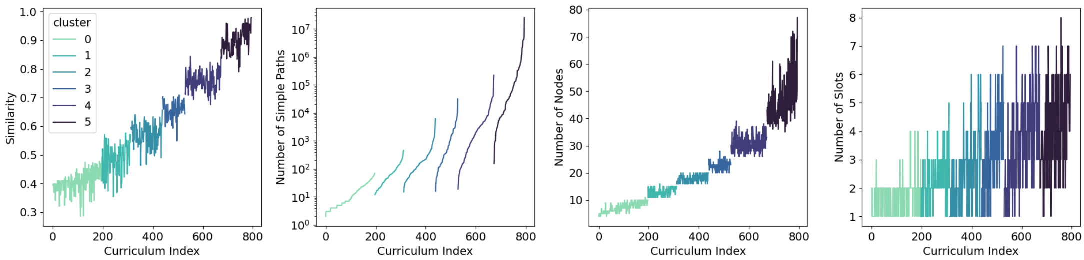
As can be seen in table 1, for the AIDS dataset the performance of the modified SimGNN model was worse across all metrics. The same follows for the LINUX dataset apart from τ. For the IMDB dataset, the modified SimGNN model performed better in terms of mean squared error. However, on this dataset, the modified model captured the relationship between the predicted and actual values worse according to ρ and τ. The overall performance decrease was expected as increasing the one-hot encoding of node labels, and mismatching the number of node labels in the dataset, leads to learning more variables than necessary.

Despite the modified model's poor performance on the AIDS dataset, because the AIDS dataset is the only one with node labels, this trained model was chosen. With the trained model, all pairwise similarity scores between graphs scores are put into a symmetrical distance matrix.

6.2 Curriculum Generation



(a) Curriculum with 3 clusters.



(b) Curriculum with 6 clusters.

Figure 7: Analysis of user goals in curricula according to the similarity score and *simple paths*, *nodes*, and *slots* complexity metrics respectively. The similarity is calculated between subsequent user goals in the curriculum.

In the Method section the curriculum generation method was described, of which, in figure 7 the evaluation of the generated curriculum can be found. Two different curricula were examined, one consisting of three clusters and one consisting of six clusters. The first metric deployed to evaluate these two curricula is the similarity between two subsequent user goals within the curriculum, as computed by the SimGNN model. The additional metrics were the *simple paths*, *nodes*, and *slots* complexity metrics as discussed in the Method section.

Both figure 7a and figure 7b show that the similarity between adjacent goals decreases with every cluster and slightly within one cluster. In both curricula, the similarity between graphs fluctuates but is less for the curriculum with six clusters. The relative consistency in similarity in each cluster is indicative that the utilized clustering method was competent. Across all clusters, a rise in user goal complexity can be seen in every metric examined. As expected the *simple paths* metric shows perfect ordering within all clusters because it was used to order goals within the clusters. In the transitions from one cluster to the next, the simple path values overlap, meaning that the similarity clustering did impact the curriculum order independently from using only the *simple paths* metric to order user goals. The six clusters do have more *simple paths* complexity overlap compared to the three clusters. The overlapping does not occur for the *nodes* metric, and for the *slots* metric there exists a significant overlap between and within each cluster. For *slots* complexity and *nodes* complexity, both curricula have around the same distribution across the clusters.

To explain the decrease in the similarity between graphs as the curriculum progresses we can look at how the size of graphs can impact graph similarity. Graphs can have the same number of edges, but not be similar because the edges connect to different nodes. And this can also lead to the graphs having the same number of simple paths. Increasing the number of connections in a graph would also give increased options for altering edges within the graph while still containing the same number of simple paths. So, the likelihood of two different graphs having the same number of simple paths increases with the size of the graph in terms of edges and nodes. This explains the decrease in the similarity between graphs in the curricula as the graph complexity rises. Additionally, figure 6 shows that the frequency of values for the *simple paths* metric becomes increasingly sparse as the values become higher. This means that the graphs with higher complexity tend to vary, resulting in the decreased similarity between two graphs of high complexity.

6.3 Dialog Policy Learning with Curriculum Learning

In this section, the impact of the generated curriculum on the dialog policy learning performance will be measured in four different experiments. The goal of the experiments is to find the best-performing configuration of parameters for the generated curriculum. The content of these experiments is contained below.

Experiment 1. This experiment concerns the effect of curricula generated with different clustering methods on performance. The curricula examined will be the ones generated with three and six clusters.

Experiment 2. In this experiment, the effect of curricula with distinct cluster ordering methods on performance will be explored. The cluster orders considered were sequenced or interleaved cluster orders.

Experiment 3. This experiment is about the performance ramifications of filtering a curriculum to contain only easy user goals.

Experiment 4. Here the effect of rerunning the whole curriculum or only failed goals after the curriculum has ended will be explored.

All experiments were performed for 1000 epochs. Because the curriculum has a length of 759 which is lower than the number of epochs, by default, the failed goals are remembered and used to fill in the remaining 241 goals. Testing of the model occurred every 10 epochs and is done on 50 randomly sampled user goals from the user goal set. Each test measures the success rate, reward, and dialog length in turns for each dialog. The performance of every configuration is averaged from five separate training sessions. The experiments conclude with a comparison of the best-performing model from the experiments and a baseline DQN model.

6.3.1 Experiment 1

Figure 8: The success rate of curricula with 3 and 6 clusters.

This experiment explores the impact on performance between curricula constructed from three versus six clusters. In figure 8 the success rate for both curricula is plotted. Here, it can be observed that neither curriculum performed significantly better than the other. In table 2 it shows that the dialog length stays between 30 and 40 turns. And the obtained reward fluctuates mostly between -50 and -60 apart from epoch 300, which is the epoch right before the fall in average success rate and obtained reward. This performance decrease happens for both curriculum configurations and happens once the complexity of the user goals increases. So, changing the number of clusters seems therefore not significant. The insignificant difference in performance between the curricula could be explained by that both clustering methods produced clusters that were comparable in figure 7a and figure 7b. But the six clusters had more separation between the complexity and included more similarity, which could explain the increase in success rate around epoch 800 when the default curriculum ends and the failed goals are repeated. For future experiments, the six-cluster curriculum is chosen.

6.3.2 Experiment 2

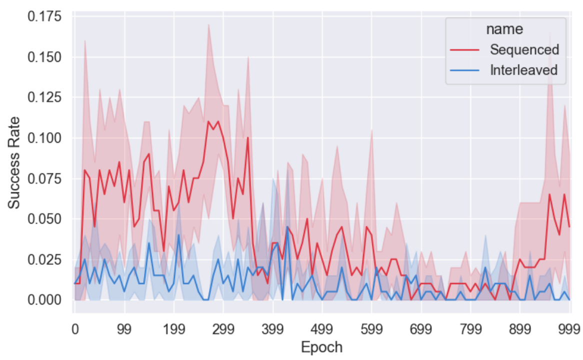


Figure 9: The success rate of curricula with sequenced and interleaved clusters.

To see if the sharp decrease in performance around epoch 300 in the previous experiment can be avoided, in this experiment the difference in performance between using sequenced and interleaved cluster order is evaluated. The aim was for the interleaved order to increase the user goal complexity variance, which would increase performance on complex user goals because these would be encountered earlier. Details of both cluster orders can be found in the 5 section. In figure 9 the results of this experiment can be seen in terms of success rate. The sequenced curriculum generally had a higher success rate than the interleaved curriculum order across all epochs. In the first 300 epochs, this separation is significantly more than after, although this remains until at least epoch 700 even with the sharp decrease also experienced in the previous experiment. The interleaved curriculum also experiences a slight decrease in success rate during the curriculum. This can be attributed to the fact that even with interleaving the user goals from each cluster, the goals do increase in complexity. In table 2 is can be seen that the average received rewards of the interleaved curriculum was also always lower. And the average dialog length also was higher but stayed in a similar 30 to 40 turns range.

6.3.3 Experiment 3

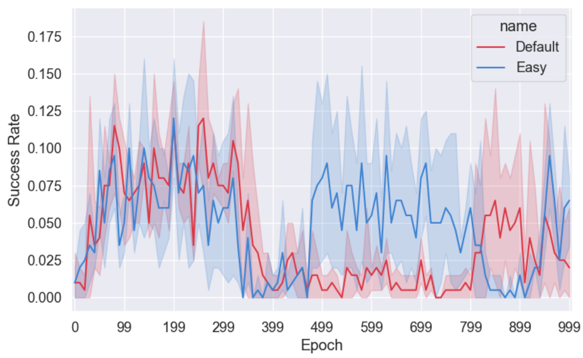


Figure 10: The success rate of curricula with default and easy clusters.

Because the last experiment did not resolve the sharp decrease in success rate, in this experiment it shall be examined if using the curriculum to filter for only easy user goals improves the overall success rate. From the six-cluster curriculum, the first three clusters were placed into a new shorter curriculum. Comparing easy and default clusters in 10 we can see in terms of success rate that the curriculum of easy clusters outperformed default clusters after the switch to complex goals in the default configuration. This increase in success rate happens when the easy curriculum ends at around 500 when the easier goals previously failed goals are restarted. Also, there is a significant decrease in performance when the easy curriculum begins to end. This is because even when selecting the three easiest clusters the tail end of the last cluster still has some complex goals, as shown in figure 7b. When this happens for the second time at epoch 700 there is no change in success rate, meaning that there is no additional information learned from the easier goals to aid in learning the more complex goals.

6.3.4 Experiment 4

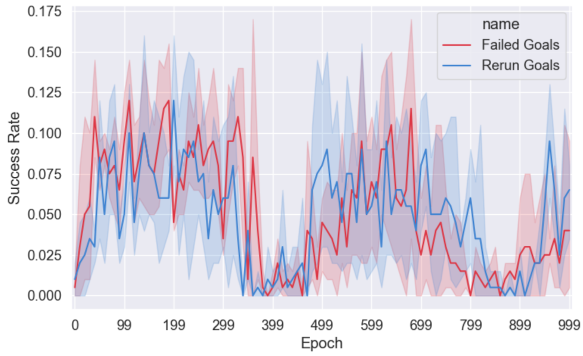


Figure 11: The success rate of curricula with repeating failed goals and restarting goals after the curriculum is over.

To expand on the previous experiment, the performance of the easy curriculum will be tested in the case when the whole curriculum is restarted, instead of only using previously failed goals to fill the full 1000 epochs. Figure 11 shows the performance difference between returning to failed goals and rerunning all goals of the curriculum. There seems to be a slight difference in the start of the curriculum restart, at around epoch 500 and epoch 950 with rerunning goals outperforming using failed goals. This is expected as so far training with only easy user goals seems to yield an increased success rate, and restarting the curriculum means that the easiest goals are again replayed, compared to previously failed goals that are slightly more difficult as the model failed on those goals. This is confirmed by the success rate equalising at the 600th epoch, and the difference increases again at epoch 700, where the harder failed goals are the majority but the easy goals continue for the easy curriculum.

6.3.5 Against the Basline DQN

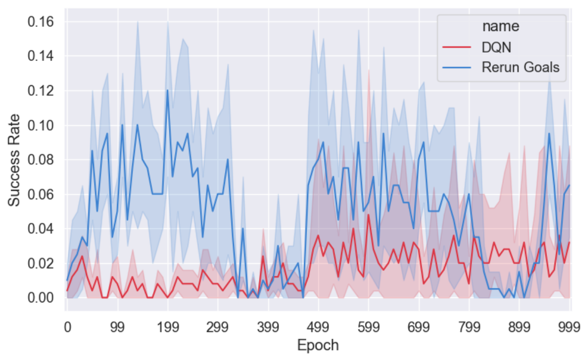


Figure 12: The success rate of the best model against the baseline DQN.

Agent	Epoch=100			Epoch=200			Epoch=300		
	Sucess	Reward	Turns	Sucess	Reward	Turns	Sucess	Reward	Turns
DQN	0.008	-57.708	39.336	0.000	-58.548	39.096	0.008	-57.748	39.416
Default 6c	0.070	-49.020	36.840	0.115	-42.800	35.200	0.075	-48.150	36.300
Default 3c	0.060	-50.560	37.520	0.055	-51.190	37.580	**0.100**	-44.785	35.570
Interleaved 3c	0.005	-58.130	39.460	0.010	-57.410	39.220	0.010	-57.465	39.330
Easy Failed	**0.095**	-45.640	36.080	0.045	-52.365	37.530	0.035	-53.845	38.090
Easy Rerun	0.050	-51.805	37.610	**0.120**	-42.245	35.290	0.060	-50.330	37.060

Epoch=400			Epoch=500			Epoch=600			Epoch=700		
Sucess	Reward	Turns	Sucess	Reward	Turns	Sucess	Reward	Turns	Sucess	Reward	Turns
0.004	-58.292	39.544	0.036	-53.824	38.288	0.048	-51.808	37.136	0.028	-54.768	38.256
0.005	-57.765	38.730	0.005	-58.045	39.290	0.015	-56.760	39.120	0.025	-55.660	39.320
0.035	-53.750	37.900	0.025	-55.125	38.250	0.040	-53.255	38.110	0.010	-57.250	38.900
0.030	-54.790	38.780	0.000	-58.765	39.530	0.000	-58.820	39.640	0.000	-58.715	39.430
0.005	-58.075	39.350	0.045	-52.365	37.530	**0.070**	-49.020	36.840	0.025	-54.180	36.360
0.005	-58.070	39.340	**0.080**	-47.845	36.890	0.055	-50.955	37.110	**0.080**	-47.895	36.990

Epoch=800			Epoch=900			Epoch=1000		
Sucess	Reward	Turns	Sucess	Reward	Turns	Sucess	Reward	Turns
0.008	-57.396	38.712	0.020	-56.064	38.928	0.032	-54.372	38.424
0.005	-57.605	38.410	**0.060**	-50.570	37.540	0.020	-56.095	38.990
0.005	-58.090	39.380	0.025	-55.445	38.890	0.045	-52.430	37.660
0.000	-58.775	39.550	0.000	-58.780	39.560	0.000	-58.585	39.170
0.000	-58.605	39.210	0.025	-55.465	38.930	0.040	-53.225	38.050
0.060	-50.525	37.450	0.015	-56.820	39.240	**0.065**	-49.790	37.180

Table 2: Result in terms of success rate, reward, and turns of all experimental models and the baseline DQN for 1000 epochs evaluated every 100 epochs. All results are averaged over 5 turns and each evaluation is on 50 dialogs. Highlighted in bold are the best scores.

In the end, the performance of the best model out of all experiments, the easy clusters with rerunning goals, outperformed the baseline DQN in terms of success rate on average. In the parts where the easy goals were not used, the performance of the best model was actually lower than the baseline DQN. The DQN also shows generally a rise in success rate over the 1000 epochs. Meanwhile, the best model experiences a slight decline around epoch 700. This is a decline because at this point the easy goals were used for the second time, the first time being at 300 epochs. In table 2 it can be confirmed that at every checkpoint the best model was higher in terms of success rate except for epoch 900 where the DQN eclipsed it.

7 Discussion

7.1 Research Question and Conclusion

The goal of this is to answer the previously stated research question: *How can the efficiency of reinforcement learning in task-oriented dialog policy learning be increased using conversation graph-based curriculum learning?* Before we answer this question and its sub-questions, it should be noted that an important limitation of the reported performance in the experiments is that these do not replicate the expected norm observed in other research on DQN-based dialog policy learning. This can be checked by comparing the baseline DQN performance in the success-rate graphs with the success-rate graphs of any paper on this subject (Peng et al., 2018). The cause of this issue as of yet not known. However, because this issue is consistent across all experiments, the results can be used to compare the model configurations against each other within this thesis. It does mean that most comparisons against other research are not viable. With this limitation noted, below in order the sub-questions will first be answered.

1. *How can an effective conversation graph for dialogs be created?* To answer this question, a proposed conversation graph creation method answer can be found in the Method section. In the Results section it is shown that the generated conversation graphs in combination with the SimGNN model are able to broadly sort the user goals in various terms of graphs/goal complexity. This would indicate that these conversation graphs are effective in representing user goal difficulty. However, because to our knowledge, this particular type of conversation graph is a novel concept, it is not possible to empirically measure the effectiveness of the creation method by comparing it to the performance of other methods. Comparison to other methods is also limited by the node labels reduction method necessary to create these graphs, which is dependent on the domain of the dataset used to generate the dialogs that populate the graphs. Therefore this method is not generalizable across different datasets without major modifications to the node labelling.

2. *How do conversation graphs impact curriculum generation?* In the curriculum generation, the conversation graphs proved in the Results section to be largely accurate in grouping user goals together in terms of user goal complexity. In terms of the defining characteristic of a curriculum to order tasks from least to most complex, the conversation graphs did not have much of a positive nor negative impact. However, the impact of conversation graphs can also be measured by the amount of pre-processing steps in the Method section required to create a curriculum suitable for dialog policy learning. This includes the development of a task-oriented dialog system that is able to output enough varied dialogs, and developing dialog state simplification to reduce the node labels to an amount that a GNN can handle. And also includes the time-intensive steps of generating of numerous conversations to fill a conversation graph, training the GNN model, and using the model to populate the similarity matrix of all conversation graphs. Although, currently this method of curriculum generation with conversation graphs is novel, and therefore without the optimisation that comes from a long history of development. Therefore, it is not possible to see the time-intensiveness of this method has enough impact to prevent this type of curriculum generation to be practical in the future.

3. *How does graph neural network-generated curriculum learning influence learning efficiency?* To answer this question, the experiment success-rate graphs in the Results section are used. When comparing against the baseline DQN, the curriculum did have a positive influence on the success rate in the first third of epochs. Although, Experiment 3 shows that this is a consequence of the curriculum to group the least complex user goals together, and the success rate does not carry over to the more complex goals. And in Experiment 4 replaying the grouping of easy user goals after a curriculum is even shown to be more effective than replaying previously failed user goals. However, across all experiments that used the curriculum, there was a substantial decrease in success rate when more complex goals were present. All of this means that the generated curriculum by using a graph neural network increased the efficiency of learning lower complexity goals, but did not increase efficiency on user goals of higher complexity

From these findings, the research question can be answered. It is certain that when comparing, within this thesis, the performance of the current best model in table 2 to the baseline DQN model, the curriculum learning model outperforms the DQN model. So, to conclude and answer the research question, this current iteration of the conversation-graph-based curriculum learning model does result in more efficient dialog policy learning in task-oriented dialog systems.

7.2 Future work

In the Results section, it shows that there is some promise in the proposed method. However, it is evident that changes are needed to bring the results up to par with related work. These could possibly be codebase-related issues, and would not really be relevant as an extension of the research topic of this thesis. A minor starting point for research on this method could be to apply it to different datasets commonly used in DPL like the Movie-Ticket (X. Li et al., 2018) dataset, which could highlight how generalisable this method is. What would be a more substantial starting point for future research is to quantify the several novel techniques in the applied DPL method which have at the time of writing no base for comparison. These novel techniques include node label reduction, conversation graph generation, and curriculum generation with a GNN. All of these could be improved by doing more extensive research by testing different parameters and coming up with metrics to quantify the quality of these techniques. As it stands, evaluating the method of this thesis on the final performance of DQN is not efficient because of the amount of time each step takes to complete for the amount of parameter iteration that would be needed to optimise the method.

Another substantial angle for future research is to alter the curriculum learning method. As was described in the 2 section several papers (Narvekar et al., 2017; Y. Wu et al., 2019) deploy some sort of curriculum manager that dynamically changes the curriculum at run time based on the performance of the learned dialog policy, and sometimes other factors. Instead of a static curriculum, it would then be an automatic curriculum.

A further research angle could be to alter the DQN algorithm itself by changing the experience replay to something like hindsight experience replay (Andrychowicz et al., 2017). One benefit is that HER would increase the performance on large state spaces, which is the case in the MultiWoz 2.1 (Eric et al., 2019) dataset. Additionally, the HER replay buffer allows for adapting the generation of conversation graphs to occur at run time, which could be stitched together similar to what was achieved in Lu et al., 2019. It would however be difficult to combine this with computing graph similarity because even approximating the graph-edit-distance is not efficient for large graphs.

In this thesis, the curriculum of user goals was created by leveraging a GNN to cluster the corresponding conversation graphs based on similarity. After this, the goals were further sorted on a complexity measure. In further research, it could maybe be worthwhile to eliminate both of these curriculum ordering steps and instead sort the goals by directly ordering the conversation graphs with methods that can be found in Graph CL (H. Li et al., 2023) research. The Graph CL methods are mostly applied to the learning of GNNs, however, because graphs can be mapped to any feature (such as user goals for a dialog system) these methods are not inherently limited to that application. As an example, in the CurGraph (Y. Wang et al., 2021) graph curriculum learning method the difficulty of graphs is measured by the intra-class and inter-class distributions of the graph embeddings gathered from an InfoGraph (Sun et al., 2019) model.

Some other hypothetical research direction could be to represent the conversation graphs as hypergraphs (Bretto, 2013). In a hypergraph, an edge can connect any number of nodes together rather than only two nodes. This graph structure could be leveraged to optimise the size of larger conversation graphs with many branching paths and cycles. The conversation hypergraphs could then be put through an HGNN (Feng et al., 2019) to compute graph similarity or some other complexity measure for a curriculum. Another change that can be made to the generation of the conversation graphs is to change the dialog policy used to generate conversations. In the Multi-Action Data Augmentation (MADA) framework (Y. Zhang et al., 2020) different responses are generated with multiple dialog policies in a one-to-many approach. This framework could be adapted to varied generate conversations instead of the currently used PPO dialog policy, with the possibility that it could generate more successful conversations which was a limitation in the current method.

In short, while the performance of this iteration of the method did not hold up in comparison, there are many potential and promising directions to take the concept of conversation graphs in dialog policy learning.

8 References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., & Zaremba, W. (2017). Hindsight experience replay. *Advances in neural information processing systems*, *30*.

Aumayr, E., Chan, J., & Hayes, C. (2011). Reconstruction of threaded conversations in online discussion forums. *Proceedings of the International AAAI Conference on Web and Social Media*, *5*(1), 26–33.

Bai, Y., Ding, H., Bian, S., Chen, T., Sun, Y., & Wang, W. (2019). Simgnn: A neural network approach to fast graph similarity computation. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 384–392.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. *Proceedings of the 26th annual international conference on machine learning*, 41–48.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural network. *International conference on machine learning*, 1613–1622.

Brambilla, M., Javadian Sabet, A., Kharmale, K., & Sulistiawati, A. E. (2022). Graph-based conversation analysis in social media. *Big Data and Cognitive Computing*, *6*(4), 113.

Bretto, A. (2013). Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*.

Budzianowski, P., Ultes, S., Su, P.-H., Mrkšić, N., Wen, T.-H., Casanueva, I., Rojas-Barahona, L. M., & Gašić, M. (2017). Sub-domain modelling for dialogue management with hierarchical reinforcement learning. *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, 86–92. https://doi.org/10.18653/v1/W17-5512

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, *3*(1), 1–27.

Cao, Y., Lu, K., Chen, X., & Zhang, S. (2020). Adaptive dialog policy learning with hindsight and user modeling. *arXiv preprint arXiv:2005.03299*.

Chen, H., Liu, X., Yin, D., & Tang, J. (2017). A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, *19*(2), 25–35.

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2), 224–227.

Durrett, R. (2007). *Random graph dynamics* (Vol. 200). Citeseer.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, *48*(1), 71–99.

Eric, M., Goel, R., Paul, S., Sethi, A., Agarwal, S., Gao, S., & Hakkani-Tür, D. (2019). Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines.

Fang, M., Zhou, T., Du, Y., Han, L., & Zhang, Z. (2019). Curriculum-guided hindsight experience replay. *Advances in neural information processing systems*, *32*.

Fatemi, M., Asri, L. E., Schulz, H., He, J., & Suleman, K. (2016). Policy networks with two-stage training for dialogue systems. *arXiv preprint arXiv:1606.03152*.

Feng, Y., You, H., Zhang, Z., Ji, R., & Gao, Y. (2019). Hypergraph neural networks. *Proceedings of the AAAI conference on artificial intelligence*, *33*(01), 3558–3565.

Gao, J., Galley, M., & Li, L. (2018). Neural approaches to conversational ai. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 1371–1374.

Gatt, A., & Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, *61*, 65–170.

Gong, C., Yang, J., & Tao, D. (2019). Multi-modal curriculum learning over graphs. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(4), 1–25.

Gordon-Hall, G., Gorinski, P. J., & Cohen, S. B. (2020). Learning dialog policies from weak demonstrations. *arXiv preprint arXiv:2004.11054*.

Gordon-Hall, G., Gorinski, P. J., Lampouras, G., & Iacobacci, I. (2020). Show us the way: Learning to manage dialog from demonstrations. *arXiv preprint arXiv:2004.08114*.

Gori, M., Monfardini, G., & Scarselli, F. (2005). A new model for learning in graph domains. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, *2*, 729–734.

Guo, S., Huang, W., Zhang, H., Zhuang, C., Dong, D., Scott, M. R., & Huang, D. (2018). Curriculumnet: Weakly supervised learning from large-scale web images. *Proceedings of the European conference on computer vision (ECCV)*, 135–150.

Hoy, M. B. (2018). Alexa, siri, cortana, and more: An introduction to voice assistants. *Medical reference services quarterly*, *37*(1), 81–88.

Huang, M., Zhu, X., & Gao, J. (2020). Challenges in building intelligent open-domain dialog systems. *ACM Transactions on Information Systems (TOIS)*, *38*(3), 1–32.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, *4*, 237–285.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kristianto, G. Y., Zhang, H., Tong, B., Iwayama, M., & Kobayashi, Y. (2018). Autonomous subdomain modeling for dialogue policy with hierarchical deep reinforcement learning. *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, 9–16. https://doi.org/10.18653/v1/W18-5702

Kumar, R., Mahdian, M., & McGlohon, M. (2010). Dynamics of conversations. *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 553–562.

Kwan, W.-C., Wang, H.-R., Wang, H.-M., & Wong, K.-F. (2023). A survey on recent advances and challenges in reinforcement learning methods for task-oriented dialogue policy learning. *Machine Intelligence Research*, 1–17.

Kwan, W.-C., Wang, H., Wang, H., & Wong, K.-F. (2022). A survey on recent advances and challenges in reinforcement learning methods for task-oriented dialogue policy learning. *arXiv preprint arXiv:2202.13675*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

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

Li, H., Wang, X., & Zhu, W. (2023). Curriculum graph machine learning: A survey. *arXiv preprint arXiv:2302.02926*.

Li, X., Chen, Y.-N., Li, L., Gao, J., & Celikyilmaz, A. (2017). End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Li, X., Wang, Y., Sun, S., Panda, S., Liu, J., & Gao, J. (2018). Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems. *arXiv preprint arXiv:1807.11125*.

Li, Y., Gu, C., Dullien, T., Vinyals, O., & Kohli, P. (2019). Graph matching networks for learning the similarity of graph structured objects. *International conference on machine learning*, 3835–3845.

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Lipton, Z., Li, X., Gao, J., Li, L., Ahmed, F., & Deng, L. (2018). Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

Liu, S., Zhang, J., He, K., Xu, W., & Zhou, J. (2021). Scheduled dialog policy learning: An automatic curriculum learning framework for task-oriented dialog system. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 1091–1102.

Lu, K., Zhang, S., & Chen, X. (2019). Goal-oriented dialogue policy learning from failures. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 2596–2603.

Ma, G., Ahmed, N. K., Willke, T. L., & Yu, P. S. (2021). Deep graph similarity learning: A survey. *Data Mining and Knowledge Discovery*, *35*(3), 688–725.

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.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., & Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*.

Narvekar, S., Sinapov, J., & Stone, P. (2017). Autonomous task sequencing for customized curriculum design in reinforcement learning. *IJCAI*, 2536–2542.

Nguyen, H., La, H. M., & Deans, M. (2019). Hindsight experience replay with experience ranking. *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 1–6.

Nimavat, K., & Champaneria, T. (2017). Chatbots: An overview types, architecture, tools and future possibilities. *Int. J. Sci. Res. Dev*, *5*(7), 1019–1024.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Park, H.-S., & Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, *36*(2), 3336–3341.

Peng, B., Li, X., Gao, J., Liu, J., Wong, K.-F., & Su, S.-Y. (2018). Deep dyna-q: Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176*.

Peng, B., Li, X., Li, L., Gao, J., Celikyilmaz, A., Lee, S., & Wong, K.-F. (2017). Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2231–2240. https://doi.org/10.18653/v1/D17-1237

Reiter, E., & Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, *3*(1), 57–87.

Ren, Z., Dong, D., Li, H., & Chen, C. (2018). Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning. *IEEE transactions on neural networks and learning systems*, *29*(6), 2216–2226.

Sacks, H., Schegloff, E. A., & Jefferson, G. (1978). A simplest systematics for the organization of turn taking for conversation. In *Studies in the organization of conversational interaction* (pp. 7–55). Elsevier.

Sanger, T. D. (1994). Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE transactions on Robotics and Automation*, *10*(3), 323–333.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Scheffler, K., & Young, S. (2000). Probabilistic simulation of human-machine dialogues. *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, *2*, II1217–II1220.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sedgewick, R. (2001). *Algorithms in c, part 5: Graph algorithms*. Pearson Education.

Skinner, B. F. (1958). Reinforcement today. *American Psychologist*, *13*(3), 94.

Sperduti, A., & Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, *8*(3), 714–735.

Stent, A., Marge, M., & Singhai, M. (2005). Evaluating evaluation methods for generation in the presence of variation. *CICLing*, *2005*, 341–351.

Su, S.-Y., Li, X., Gao, J., Liu, J., & Chen, Y.-N. (2018). Discriminative deep dyna-q: Robust planning for dialogue policy learning. *arXiv preprint arXiv:1808.09442*.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., & Fergus, R. (2017). Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*.

Sun, F.-Y., Hoffmann, J., Verma, V., & Tang, J. (2019). Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, *3*, 9–44.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990* (pp. 216–224). Elsevier.

Svetlik, M., Leonetti, M., Sinapov, J., Shah, R., Walker, N., & Stone, P. (2017). Automatic curriculum graph generation for reinforcement learning agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1).

Tang, D., Li, X., Gao, J., Wang, C., Li, L., & Jebara, T. (2018). Subgoal discovery for hierarchical dialogue policy learning. *arXiv preprint arXiv:1804.07855*.

Tay, Y., Wang, S., Tuan, L. A., Fu, J., Phan, M. C., Yuan, X., Rao, J., Hui, S. C., & Zhang, A. (2019). Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. *arXiv preprint arXiv:1905.10847*.

Tulshan, A. S., & Dhage, S. N. (2019). Survey on virtual assistant: Google assistant, siri, cortana, alexa. *Advances in Signal Processing and Intelligent Recognition Systems: 4th International Symposium SIRS 2018, Bangalore, India, September 19–22, 2018, Revised Selected Papers 4*, 190–201.

Walker, M. A., Litman, D. J., Kamm, C. A., & Abella, A. (1997). Paradise: A framework for evaluating spoken dialogue agents. *arXiv preprint cmp-lg/9704004*.

Wang, X., Chen, Y., & Zhu, W. (2021). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, Y., Wang, W., Liang, Y., Cai, Y., & Hooi, B. (2021). Curgraph: Curriculum learning for graph classification. *Proceedings of the Web Conference 2021*, 1238–1248.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*, 279–292.

Weld, H., Huang, X., Long, S., Poon, J., & Han, S. C. (2022). A survey of joint intent detection and slot filling models in natural language understanding. *ACM Computing Surveys*, *55*(8), 1–38.

Williams, J. D., Raux, A., & Henderson, M. (2016). The dialog state tracking challenge series: A review. *Dialogue & Discourse*, *7*(3), 4–33.

Wu, Y., Li, X., Liu, J., Gao, J., & Yang, Y. (2019). Switch-based active deep dyna-q: Efficient adaptive planning for task-completion dialogue policy learning. *Proceedings of the AAAI conference on artificial intelligence*, *33*(01), 7289–7296.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, *32*(1), 4–24.

Xu, Y., Zhu, C., Peng, B., & Zeng, M. (2020). Meta dialogue policy learning. *arXiv preprint arXiv:2006.02588*.

Zayats, V., & Ostendorf, M. (2018). Conversation modeling on reddit using a graph-structured lstm. *Transactions of the Association for Computational Linguistics*, *6*, 121–132.

Zhang, Y., Ou, Z., & Yu, Z. (2020). Task-oriented dialog systems that consider multiple appropriate responses under the same context. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(05), 9604–9611.

Zhang, Z., Liao, L., Zhu, X., Chua, T.-S., Liu, Z., Huang, Y., & Huang, M. (2020). Learning goal-oriented dialogue policy with opposite agent awareness. *arXiv preprint arXiv:2004.09731*.

Zhang, Z., Takanobu, R., Zhu, Q., Huang, M., & Zhu, X. (2020). Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, *63*(10), 2011–2027.

Zhao, Y., Wang, Z., & Huang, Z. (2021). Automatic curriculum learning with overrepetition penalty for dialogue policy learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(16), 14540–14548.

Zhu, Q., Zhang, Z., Fang, Y., Li, X., Takanobu, R., Li, J., Peng, B., Gao, J., Zhu, X., & Huang, M. (2020). Convlab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems. *arXiv preprint arXiv:2002.04793*.