Simulating Intrinsic and Extrinsic User Behaviour in Task-Oriented Dialogues

Inaugural-Dissertation

zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Hsien-Chin Lin
aus Taiwan

Düsseldorf, October 2023

aus dem Institut für Informatik
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Berichterstatter:

1. Prof. Dr. Milica Gašić

2. Prof. Dr. Laura Kallmeyer

Tag der mündlichen Prüfung: 21.02.2024

Declaration of Authorship

I, *Hsien-Chin Lin*, declare under oath that I have produced my thesis independently and without any undue assistance by third parties under consideration of the Principles for the Safeguarding of Good Scientific Practice at Heinrich Heine University Düsseldorf.

. . . .
Hsien-Chin Lin

Date: . . . .

Acknowledgements

I would like to thank those who helped me on my research journey.

First and foremost, I am deeply appreciative to my supervisor, Prof. Dr. Milica Gašić, for her invaluable insights, guidance, and constructive feedback. I am thankful to my second reviewer, Prof. Dr. Laura Kallmeyer, for her inspiring suggestions. I express my thanks to the chairman, Prof. Dr. Thomas J. J. Müller, and committee members, Prof. Dr. Stefan Conrad, Prof. Dr. Stefan Dietze, and Prof. Dr. Gabriella Lapesa. Their support and expertise were indispensable in making my defence possible.

A special acknowledgement goes to my team members, Dr. Nurul Fithria Lubis, Dr. Michael Heck, Dr. Benjamin Matthias Ruppik, Carel van Niekerk, Christian Geishauser, Shutong Feng, and Renato Vukovic, whose assistance in proofreading, experiment setting, and brainstorming are essential in my research. I also thank the assistance of our group's secretaries, Annette Peitsch and Verena Schlösser-Lewin.

I would like to thank the Junior Scientist and International Researcher Center (JUNO) at Heinrich Heine University Düsseldorf. Their assistance with the visa application process significantly facilitated my research experience in Germany.

Lastly, I could not have undertaken this journey without my family. Many thanks to my wife and my parents.

Abstract

Task-oriented dialogue systems guide users to accomplish their goals with respect to specific tasks, such as searching for restaurants or booking flight tickets. These systems need to deal with complex and ambiguous user queries, which cannot be simplified into a sequence of keywords but are conveyed in natural language and refined through conversation. Building such systems requires substantial amounts of example dialogues for training and evaluation, which means learning and testing from real users is time-consuming and labour-intensive. As a result, user simulation is essential in developing task-oriented dialogue systems since it facilitates efficient and cost-effective interactions with such systems.

User simulators can be categorised into two groups: rule-based and data-driven. Rule-based user simulators are composed of various manually designed rules, while data-driven user simulators learn user behaviour from data. Rule-based user simulators are highly interpretable, but their use for complex scenarios is not practicable since the designing of rules may rapidly become intractable. Data-driven user simulators circumvent the need for managing complex rules in realistic scenarios but have limited explainability. There are more challenges in user simulation. Firstly, domain adaptation in user simulation is not trivial. Adapting rule-based user simulators to new domains requires rewriting handcrafted rules, which is laborious and costly. Moving data-driven user simulators to unseen domains requires retraining or even re-engineering models since they may rely on domain-dependent feature representations. Secondly, existing user models cannot capture the richness of user behaviour properly, including the user's intrinsic status, e.g. user persona and emotions, and extrinsic behaviour, e.g. user actions and utterances. These shortcomings will cause sub-optimal performance in user simulation. For instance, without considering user emotions, user models cannot capture diverse human behaviour that is driven by emotions. Generating user responses in the form of semantic actions will lose the linguistic variation in natural language, while only generating utterances in natural language may constrain interpretability.

The main contributions of this thesis are as follows. Firstly, our novel proposed user simulators are domain-agnostic, overcoming the domain-dependency issue in existing user simulators and enabling transfer to unseen domains in a zero-shot fashion. Secondly, we introduce a framework for joint optimisation of user policy, which decides what actions the user model should take, and natural language generation, resulting in generating more human-like user utterances based on semantic actions and dialogue context. Thirdly, we integrate user emotions and personas into user simulation, thereby providing more fine-grained feedback beyond only task success and probing the impact of system behaviour on the user's emotional state. Our experimental results demonstrate that dialogue systems built with the help of the proposed user simulators improve user experience. In addition, modelling user behaviour in a wider spectrum, e.g. from external actions to internal status, not only allows us to explore the effects of system behaviour on the user experience but also provides a valuable tool for future research addressing ethical considerations.

Contents

List of Figures

List of Tables

Summary of Notation

This chapter summarises the notations used throughout this thesis.

General Mathematical Notation

a, b, c	Scalars
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$	Vectors
a_i	i-th element of vector $\boldsymbol{a}$
$\boldsymbol{a} = [a_i]_{i=1}^N$	vector written as a list of elements a_i
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$	Matrices
$\boldsymbol{a_i}$	i-th row of matrix $\boldsymbol{A}$
$\boldsymbol{A}^T$	Transposition of matrix $\boldsymbol{A}$
$\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$	Sequence of N vectors
$\boldsymbol{X} = (\boldsymbol{x}_i)_{i=1}^N$	Matrix written as a list of rows $\boldsymbol{x}_i$
$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	Sets
$x \in \mathcal{X}$	x is an element of set $\mathcal{X}$
$f(x)$	A function of x
$f(\cdot)$	A function f with one argument
$\mathbb{N}$	Natural numbers
$\mathbb{R}$	Real numbers
$\mathbb{R}^d$	Set of d-dimensional vectors with elements in $\mathbb{R}$
$\mathbb{R}^{d_1 \times d_2}$	Real-valued matrices with d_1 rows and d_2 columns
$\boldsymbol{a} \odot \boldsymbol{b}$	element-wise multiplication of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
$\boldsymbol{AB}$	Matrix-matrix multiplication of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$
e^x	Exponential function
$\log(x)$	Natural logarithm of x
Σ	Summation
∇f	Gradient of a function f
$\max_x f(x)$	Maximum value of function f
$\arg\max_x f(x)$	Vlaue of x that maximises the function $f(x)$
$\lceil x \rceil$	Ceiling function, mapping $x \in \mathbb{R}$ to the least integer greater than or equal to x

Deep Learning Notation

$f(\boldsymbol{x};\boldsymbol{\theta})$	function f with input $\boldsymbol{x}$, parameterised with parameters $\boldsymbol{\theta}$
f_l	l-th layer of a neural network
$\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$	Dataset of size N with input-output pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$
$\mathcal{L}(\mathcal{D};\boldsymbol{\theta})$	Loss function dependent on parameters $\boldsymbol{\theta}$ and dataset $\mathcal{D}$
$\nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathcal{D};\boldsymbol{\theta})$	Gradient of $\mathcal{L}$ with respect to $\boldsymbol{\theta}$
η	Learning rate
ϕ	Activation function
$\boldsymbol{h}_l$	Output of the l-th layer in a neural network
$\hat{\boldsymbol{y}}$	Output vector of a neural network $f(\boldsymbol{x};\boldsymbol{\theta})$ for an input $\boldsymbol{x}$
$a \leftarrow b$	Assigning value b to a

Reinforcement Learning Notation

$\boldsymbol{s}, \boldsymbol{s}'$	State and next state
$\boldsymbol{a}$	Action
r	Reward
$P(\boldsymbol{s}'\|\boldsymbol{s}, \boldsymbol{a})$	Transition probability of $\boldsymbol{s}'$ given $\boldsymbol{s}, \boldsymbol{a}$
$P_0(\boldsymbol{s})$	Initial probability of starting in state $\boldsymbol{s}$
t	discrete time-step
$\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t$	state, action, reward at time-step t
π	Policies
$\pi(\boldsymbol{a}\|\boldsymbol{s})$	Probability of action $\boldsymbol{a}$ given state $\boldsymbol{s}$ for policy π
$V^\pi(\boldsymbol{s})$	State-value function of π
$Q^\pi(\boldsymbol{s}, \boldsymbol{a})$	Action-value function of π
$A^\pi(\boldsymbol{s}, \boldsymbol{a})$	Advantage function of π
$V(\boldsymbol{s})$	Estimate of State-value function
τ	Trajectory of states, actions and rewards
$E_\pi[X]$	Expectation of X when generating experience using π

Chapter 1

Introduction

1.1 Overview

In the era of information explosion, it is crucial to develop systems that enable users to access specific information accurately while effectively managing information overload (Fischer, 2001). Task-oriented dialogue systems have emerged as an attractive solution, assisting users to achieve specific objectives within a particular task through conversations in natural language, such as finding the most affordable flight ticket or booking a hotel based on specific constraints.

Before deploying the system to real users, it is paramount to ensure that the system functions optimally. Testing with real users can be labour-intensive and may not provide sufficient coverage. Hence, user simulators are invaluable in facilitating comprehensive dialogue evaluation. User simulation not only plays an important role in evaluations but also accelerates the development of task-oriented dialogue systems. By reinforcement learning, the training of task-oriented dialogue systems has necessitated thousands of interactions to establish a functional policy (Li et al., 2017; Pietquin et al., 2011; Schatzmann et al., 2007; Shi et al., 2019). However, learning from humans for substantial numbers of conversations is time-consuming and costly. To address this issue, user simulation has gained significant attention. By generating a large number of dialogues using a user simulator, the training process becomes more efficient and cost-effective.

1.2 Task-oriented dialogue systems

Task-oriented dialogue systems are designed to help users accomplish specific goals within a particular task such as hotel booking or finding a flight. As shown in Figure 1.1, solving this problem typically requires tracking and planning (Young, 2002). In *tracking*, the system keeps track of information of the dialogue from the beginning until the current dialogue turn. In *planning*, the dialogue policy makes decisions at each turn and generates responses which lead the conversation toward the task success (Levin and Pieraccini, 1997). What a task-oriented dialogue system can understand and talk about is defined by an **ontology**. As shown in Figure 1.2, an ontology may include different **domains**, such as *restaurant* or *hotel*. Domains consist of a set of entities, e.g. the restaurant domain includes different restaurants. **Slots** describe the properties that entities can have e.g. *area* or *price* and **values** are specific values that these slots take e.g. *centre* or *expensive*.

The task-oriented dialogue system can be modularised with different components, consisting of a natural language understanding module, a dialogue state tracker, a dialogue policy, and a natural language generator (Geishauser et al., 2022; Henderson et al., 2013; Thomson and Young, 2010; Williams and Young, 2007; Young et al., 2010). It can also be

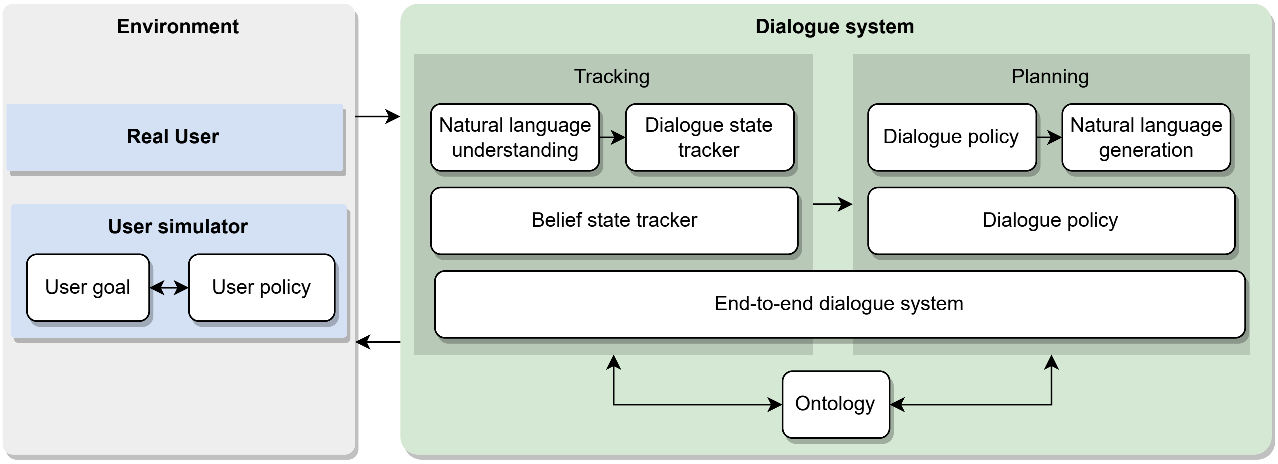


FIGURE 1.1: The overview of task-oriented dialogue systems and environments.

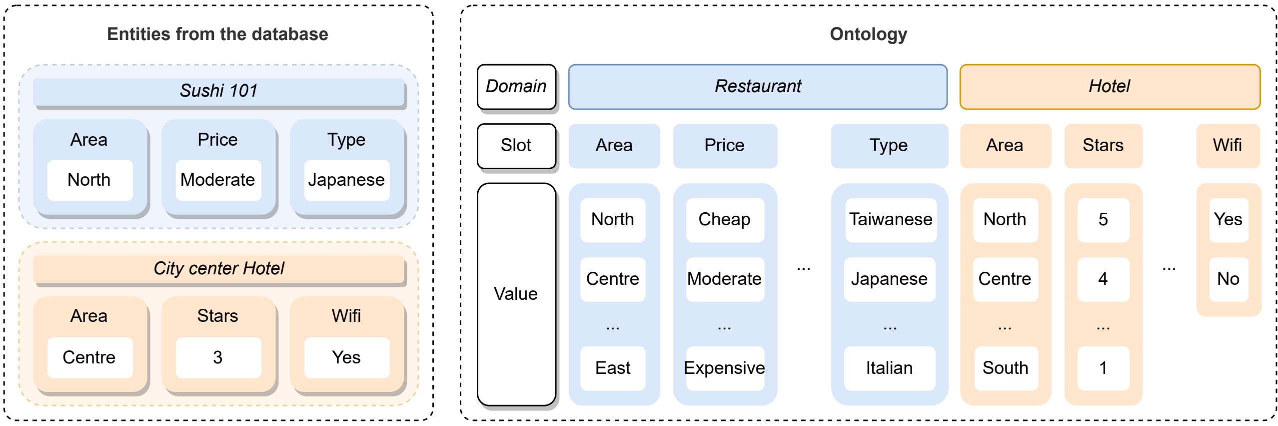


FIGURE 1.2: The ontology of task-oriented dialogues defines what a dialogue system can understand and talk about.

modelled as an end-to-end system (Bordes et al., 2017; Lubis et al., 2020; Madotto et al., 2018; Peng et al., 2021; Wen et al., 2017). These modules can be optimised by supervised learning, however, given a more complicated ontology introduced intractable possible dialogue paths, it is more reasonable to optimise planning modules, e.g. dialogue policy, by reinforcement learning.

Training a dialogue policy by reinforcement learning requires substantial numbers of interactions. Learning with real users is time-consuming, thus it is appealing to build a simulated environment, e.g. user simulation, for development and evaluation. In this thesis, we focus on the user simulation to model human behaviour during conversations. In the subsequent sections, we will outline the challenges of user simulation for task-oriented dialogue systems.

1.3 Challenges in user simulation

User simulators are widely used for training and testing task-oriented dialogue systems and impact the performance of reinforcement learning-based dialogue systems significantly (Schatzmann et al., 2005). However, existing user simulators suffer from several issues, as mentioned in the following subsections.

1.3.1 Limited scope of user behaviour in user simulation

Existing user simulators do not capture the richness of user behaviour properly. For example, they only model the extrinsic user behaviour, e.g. semantic actions, but neglect the intrinsic user status, e.g. user emotions and persona. As a result, these models are not able to capture diverse user behaviour triggered by different internal states. Optimising dialogue systems with these generic user policies may result in a high drop-off rate when deployed in the real world. Thus, it is important to simulate the full spectrum of user behaviour from extrinsic to intrinsic behaviour in user simulation.

1.3.2 Domain-dependency of user simulation

Adapting user simulators to a new domain is challenging. Rule-based user simulators are composed of various handcrafted rules, which require intensive expert knowledge and must be redesigned when adapting to an unseen domain. Conversely, data-driven user simulators can learn user behaviour from datasets, but they may still have domain-dependent input feature representation or output targets. As a result, we need to collect and annotate a new dataset and then retrain or reengineer the user model when moving to a new ontology.

The domain dependency of user simulators will create barriers to building dialogue systems on a new customised domain. Therefore, it is essential to design a domain-independent user simulator, which can learn ontology-agnostic user behaviours and generalise to unseen domains.

1.3.3 Linguistic variation

The user simulators at the semantic level provide dialogue acts as user responses only, limiting the language diversity of these models. Concatenating a natural language generator can transform the semantic dialogue acts into natural language utterances. However, if the generator solely relies on semantic actions as input, the generated results may not be sufficiently natural within a given context. For example, as shown in Figure 1.3, a natural language generator may be able to transfer the user semantic action, `[Inform, Hotel, Area, North]` to *"I want to find a hotel in the north"*, but in a different context, e.g. the system informs an incorrect hotel area, the natural language generator may not be able to transfer the same semantic action to *"No, I want one in the north, not south"*. Furthermore, the inability to optimise the user policy and the natural language generation jointly can lead to suboptimal performance. As a result, it is desirable to have a user simulator capable of generating both semantic actions and natural language utterances.

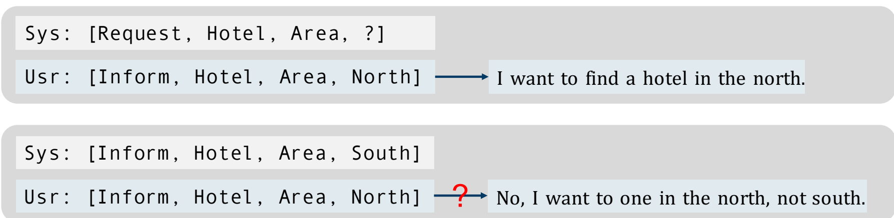


FIGURE 1.3: An example of the result of a separated natural language generation model is not natural enough in a given context.

1.3.4 Interpretability

Although rule-based user policies and template-based natural language generators composed of handcrafted rules cannot capture the richness of human behaviour, these modules are fully explainable. On the other hand, data-driven user simulators that rely on deep neural networks suffer from limited interpretability (Arrieta et al., 2020; Castelvecchi, 2016). It is hard to improve dialogue systems when training and testing with user simulators which are difficult to interpret. In addition, it is also difficult to control or change the behaviour of uninterpretable user simulators for a more diverse environment. As a result, it is crucial to build an interpretable user simulator.

1.4 Contributions

In this thesis, we focus on the user simulation of task-oriented dialogue systems. The contributions can be summarised as follows:

- **Domain-independent user simulation:** a semantic level user simulator, TUS, is proposed that uses domain-independent features and natural language level user simulators, GenTUS and EmoUS, that represent input and output as token sequences. All user simulators proposed in this thesis are domain-agnostic and are able to adapt to an unseen domain in a zero-shot fashion.

- **Joint optimisation of user policy and language generation:** GenTUS and EmoUS are proposed that are able to generate both semantic actions and natural language utterances as user responses. By jointly optimising the user policy and language generation, they are able to generate more natural language responses in a given context.

- **User emotions in user simulation:** The user persona and user emotions are included in EmoUS. By considering the user intrinsic status, EmoUS can capture various behaviour driven by different emotions and generate more diverse responses. In addition, EmoUS can be used as a probe to evaluate dialogue systems, especially for how dialogue systems affect the user's emotions. Developing such methods is important as the increasing usage of large language model chat-bots and rising ethical concerns.

1.5 Thesis Structure

This thesis consists of seven chapters. In Chapter 1, a comprehensive overview is provided to introduce the user simulation, specify the challenges, and outline the contributions of the thesis. Chapter 2 provides an introduction to deep learning, explaining how to use it in the downstream tasks. Chapter 3 is the literature review of user simulation, discussing the strengths and weaknesses of existing methods. In Chapter 4, TUS is introduced, a semantic-level transformer-based user simulator with domain-independent feature representations. In Chapter 5, GenTUS is proposed, a user simulator which simulates user behaviour and language jointly with generative transformers. In Chapter 6, EmoUS, which simulates user emotions for task-oriented dialogues, is proposed based on GenTUS. Chapter 7 summarises the results of this thesis and suggests directions for future study.

Chapter 2

Machine Learning Background

2.1 Neural Networks

Artificial neural networks, usually called neural networks, are inspired by the human brain. They are composed of a huge amount of interconnected simple processors, usually called neurons, that can run in parallel (Jain et al., 1996). The inputs and outputs of these neurons are connected by directed weighted edges, i.e. neural networks are weighted-directed graphs. In addition, neural networks can be viewed as function approximations, which means neural networks are used to estimate an unknown function using available observations from the task-specific domain. For instance, consider there is an unknown function f_{unk} that takes user utterances and predicts the user's intent, such as searching for a hotel or booking a restaurant. To approximate this function f_{unk}, we train a neural network f_{NN} with a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$, consisting of N pairs of user utterances x and corresponding intents y. In addition, this neural network f_{NN} should be able to generalise to user utterances which are unseen in the dataset.

In this chapter, the structure of feed-forward neural networks is mentioned in Section 2.2, the structure of recurrent neural networks is described in Section 2.3, the attention mechanism is discussed in Section 2.4, and the structure of transformers is presented in Section 2.5. The basic information of neural network optimisation is explained in Section 2.6 and a brief introduction of reinforcement learning is specified in Section 2.7.

2.2 Feed-forward Neural Networks

A feed-forward neural network, which maps the input $\boldsymbol{x}$ to the output $\hat{\boldsymbol{y}}$, can be defined as $\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the learnable parameter. The feed-forward neural network can be formed with multiple layers $f_1, f_2, \ldots, f_L$, as illustrated in Figure 2.1.

Each layer l can be formulated as:

$$f_l(\boldsymbol{h}_{l-1}) = \phi_l(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l), l = 1, \ldots, L \tag{2.1}$$

where its input $\boldsymbol{h}_{l-1} \in \mathbb{R}^{d_{l-1}}$ is a real number vector with dimension $d_{l-1} \in \mathbb{N}$ from the output of the previous layer $l-1$, $\boldsymbol{h}_0 = \boldsymbol{x}$, $\boldsymbol{h}_L = \hat{\boldsymbol{y}}$ and ϕ_l is an activation function which can be linear or nonlinear.

For an input vector $\boldsymbol{x}$ with dimension d, common activation functions are shown as follows:

$$\phi_{\texttt{Sigmoid}}(\boldsymbol{x}) = \left[\frac{1}{1+e^{-x_i}}\right]_{i=1}^{d} \tag{2.2}$$

$$\phi_{\texttt{ReLU}}(\boldsymbol{x}) = [\max(0, x_i)]_{i=1}^{d} \tag{2.3}$$

$$\phi_{\texttt{softmax}}(\boldsymbol{x}) = \left[\frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}\right]_{i=1}^{d} \tag{2.4}$$

Activation functions are typically nonlinear. According to the Universal Approximation Theorem (Cybenko, 1989), a two-layer neural network serves as a universal approximator when its activation functions are nonlinear. On the other hand, if all activation functions are linear, the entire network is equivalent to a single-layer model. The Sigmoid function (Equation 2.2) maps real-valued inputs into a value between $(0, 1)$ and was commonly used as an activation function. However, it suffers from the problem of gradient vanishing (see Appendix A), which makes training deep networks difficult (Pascanu et al., 2013). In contrast, the rectified linear unit (ReLU) (Equation 2.3) has fewer gradient vanishing problems and is more computation efficient, which makes it the most popular activation function (Fukushima, 1969; Glorot et al., 2011). In addition, the softmax function (Equation 2.4) is widely chosen to be the activate function of the last layer of neural networks because it can normalise a general vector into a probability distribution over output classes.

As shown in Figure 2.1, the network is parameterised with weights, $\boldsymbol{W_l} \in \mathbb{R}^{d_l \times d_{l-1}}$, and biases, $\boldsymbol{b_l} \in \mathbb{R}^{d_l}$. These parameters can be optimised based on a dataset by stochastic gradient descent algorithms, such as adaptive subgradient methods (AdaGrad) (Duchi et al., 2011) and adaptive moment estimation (Adam) (Kingma and Ba, 2015) (see Section 2.6).

Feed-forward neural networks have impressive generalisation ability and have been proposed to solve various tasks, such as pattern recognition. However, these networks are memoryless, which means different inputs are independent, thus it is not suitable for sequential prediction tasks, such as natural language generation, which should capture the relationship between various inputs in a sequence.

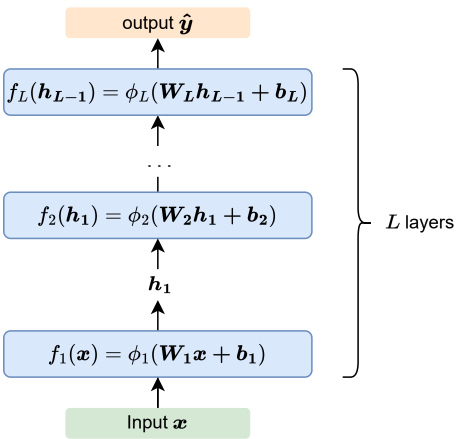


FIGURE 2.1: An example of a feed-forward neural network.

2.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are often employed to handle sequential or time series data, including tasks like language translation and speech recognition. Unlike feed-forward neural networks, RNNs have internal loops. By introducing connections from the previous time step, RNNs have "memory", which means they can leverage past inputs to impact the present input and output (Dey and Salem, 2017; Elman, 1990; Hochreiter and Schmidhuber, 1997; Jordan, 1997). This process can be formulated as:

$$\boldsymbol{h}_t, \hat{\boldsymbol{y}}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}; \boldsymbol{\theta}), t = 1, \ldots, N. \tag{2.5}$$

where the sequential input and output are represented as $\boldsymbol{X} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)$ and $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{y}}_0, \ldots, \hat{\boldsymbol{y}}_N)$ respectively, $\boldsymbol{h}_t$ is the hidden vector output by the RNN at time step t, and $\boldsymbol{\theta}$ is the parameters of the RNN. One example is shown in Figure 2.2.

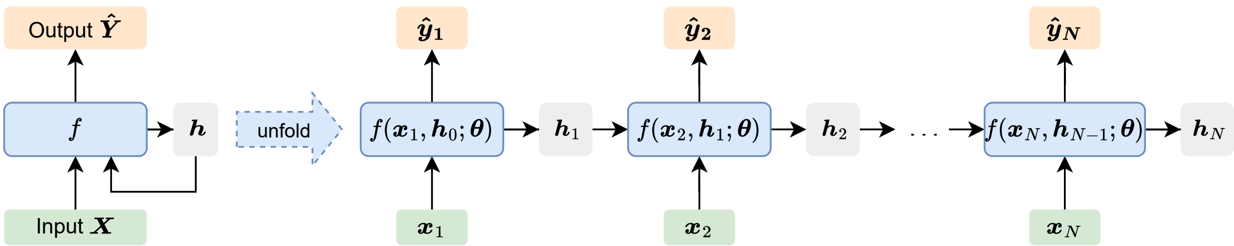
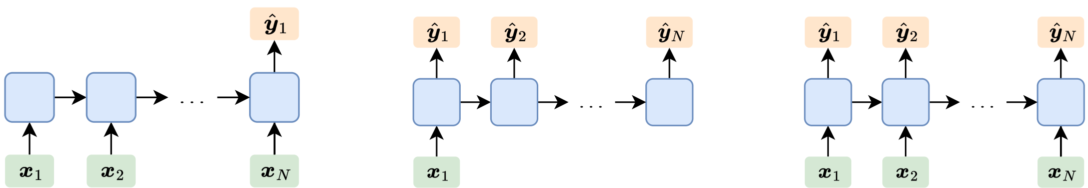


FIGURE 2.2: An example of a recurrent neural network.

Various sequential learning problems may have different structures (Lipton et al., 2015), as shown in Figure 2.3. The many-to-one structure (Figure 2.3 (A)) maps a sequence into a fixed-length vector, which can be used for sequence classification, e.g. text classification. The one-to-many structure (Figure 2.3 (B)) generates sequential outputs based on a single input vector, which can be used for image caption generation. The many-to-many structure (Figure 2.3 (C)) can be used for language generation, where the model predicts the next token at each time step. In the following section, we will describe the encoder-decoder structure, which is one of the most common structures.



(A) many-to-one (B) one-to-many (C) many-to-many

FIGURE 2.3: Various structures of RNNs. Green boxes are inputs and orange boxes are outputs.

2.3.1 Encoder-decoder in RNNs

The encoder-decoder structure is designed for sequence-to-sequence tasks where the input sequence and output sequence have varying lengths, such as machine translation (Bahdanau et al., 2015), speech recognition (Weninger et al., 2015) and user simulation (El Asri et al., 2016; Kreyssig et al., 2018). This structure consists of two main components, the *encoder* and the *decoder*. The encoder f_{enc} will convert the input sequence $\boldsymbol{X} = (\boldsymbol{x}_i)_{i=1}^N$ into a fixed-length vector, also called a context vector $\boldsymbol{c}$. Then the decoder f_{dec} will generate an output sequence based on this vector, which means the context vector $\boldsymbol{c}$ is the initial hidden vector for the

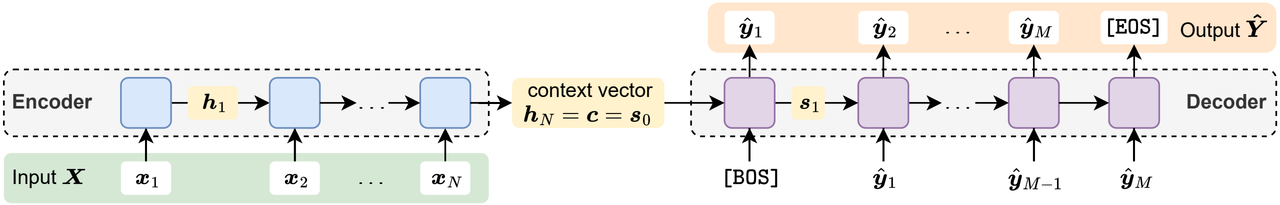


FIGURE 2.4: The encoder-decoder structure in RNNs. `[BOS]`, `[EOS]` are vectors that represent the start and the end of a sequence respectively.

decoder. As shown in Figure 2.4, this process can be formulated as:

$$\begin{aligned} h_i &= f_{enc}(x_i, h_{i-1}), i = 1, \dots, N \\ s_0 &= c = h_N \\ s_t, \hat{y}_t &= f_{dec}(\hat{y}_{t-1}, s_{t-1}), t = 1, \dots, M \end{aligned} \tag{2.6}$$

where h_i is the hidden vector in the encoder f_{enc} of input token x_i, and s_t is the hidden vector in the decoder f_{dec} of output $\hat{y}_t$ at time step t. N, M are the length of input and out sequences respectively.

One possible drawback of the encoder-decoder structure in Figure 2.4 is that the encoder may lose information when compressing whole input sentences into a fixed-length vector, particularly if the input sequences are longer than the sequences in the training dataset (Bahdanau et al., 2015). Therefore, it is appealing to introduce the attention mechanism to help the decoder access the entire input sequence, reducing the risk of information loss.

2.4 Attention Mechanism

The attention mechanism enables models to focus on specific parts of input data when making predictions or decisions. This mechanism highlights relevant data, functioning like a spotlight to improve the overall model's effectiveness, especially for capturing important information in long sequences.

In general, the input of the attention layer includes three parts, the *query*, the *keys*, and the *values*. The *query* specifies what the model is looking for in the input sequence, the *keys* represent information about the input sequence, and the *values* contain the actual information that the model should use. The similarity or relevance scores between the *query* and *keys* are referred to as the *attention weights*. The weighted sum of the *values* based on the *attention weights* is often referred to as the *context vector*, which is used in subsequent computations in the model. The attention mechanism can be implemented sequentially (Section 2.4.1) or in parallel (Section 2.4.2).

2.4.1 Attention Mechanism in RNNs

The attention mechanism in RNNs in encoder-decoder architecture proposed by Bahdanau et al. (2015) is shown in Figure 2.5. The encoder transforms the input sequence X into hidden vectors $H = (h_1, h_2, \dots, h_N)$ where N is the length of the input sequence. At time step t, the attention layer takes the hidden vector s_{t-1} from the decoder in the previous time step $t-1$ as the *query* and the hidden vectors H from the encoder as the *keys*. The relevance scores

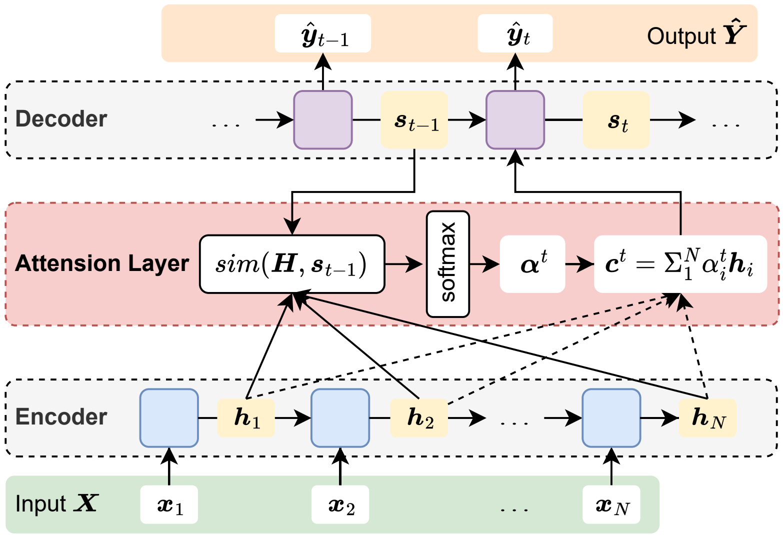


FIGURE 2.5: The attention mechanism in RNNs (Bahdanau et al., 2015)

between the query $\boldsymbol{s}_{t-1}$ and the keys $\boldsymbol{H}$ are the *attention weights* $\boldsymbol{\alpha}^t$, calculated by a similarity function $\texttt{sim}(\cdot,\cdot)$ and normalised by a softmax function (Equation 2.4). This process can be formulated as:

$$\begin{aligned} e_i^t &= \texttt{sim}(\boldsymbol{h}_i, \boldsymbol{s}_{t-1}), i = 1, \ldots, N \\ \boldsymbol{\alpha}^t &= \phi_{\texttt{softmax}}(\boldsymbol{e}^t), \end{aligned} \tag{2.7}$$

where the similarity function $\texttt{sim}(\cdot,\cdot)$ can be any distance measurement, such as a cosine similarity function or a feed-forward layer trained with the other components jointly. The higher attention weight α_i^t means the token $\boldsymbol{x}_i$ is more relevant to the output token $\hat{\boldsymbol{y}}_t$. The context vector $\boldsymbol{c}^t$ at time step t is a linear combination of the value vectors, i.e. hidden vectors $\boldsymbol{H}$ from the encoder, weights by the attention weights $\boldsymbol{\alpha}^t$, which can be formulated as:

$$\boldsymbol{c}^t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i, \tag{2.8}$$

then this context vector $\boldsymbol{c}^t$ is fed into the decoder for the output $\hat{\boldsymbol{y}}^t = f_{dec}(\boldsymbol{s}_{t-1}, \boldsymbol{c}^t)$.

2.4.2 Scaled Dot-Product Attention

The sequential attention mechanism in RNNs processes input sequences one token at a time, which is computationally inefficient. Therefore, it is appealing to calculate the attention for all queries, keys and values at the same time.

The scaled dot-product attention proposed by Vaswani et al. (2017) is an attention mechanism in parallel and widely used in various transformer-style language models, such as BERT (Devlin et al., 2019), T5 (Raffel et al., 2020) and GPT (Radford et al., 2018). The three inputs for the attention layer are the *queries* $\boldsymbol{Q} = (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots)$, *keys* $\boldsymbol{K} = (\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots)$, and *values* $\boldsymbol{V} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots)$. The *attention weights* are the dot products of the query vector with all keys, divided by the square root of the dimension of the key vector $\sqrt{d_k}$, and passed through a softmax function. This attention mechanism can be formulated as

$$\texttt{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \phi_{\texttt{softmax}}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{2.9}$$

where all operations are matrix multiplication, which is highly parallelisable.

Why do we need to divide by the scale $\sqrt{d_k}$ Assume each element of the query vector $\boldsymbol{q}$ and the key vector $\boldsymbol{k}$ are sampled from a standard normal distribution $\mathcal{N}(0,1)$ and their dimension is d_k, then their dot product $\boldsymbol{q}\cdot\boldsymbol{k} = \sum_{i=1}^{d_k} q_i k_i$ follows a normal distribution with mean 0 and variance d_k. In other words, the result of the dot product may grow large with a large value d_k, which can cause small gradients in the softmax function. Therefore, scaling the dot product with $\frac{1}{\sqrt{d_k}}$ results in a variance of 1 and as a result, stabilises the training process.

2.5 Transformers

The transformer, proposed by Vaswani et al. (2017), is composed purely of attention mechanisms (Section 2.4.2) without recurrent components. With the attention mechanism, transformers are able to capture the global relationship between input and output sequences. There are four important components in the transformers, the multi-head attention mechanism which is composed of the self-attention mechanism (Section 2.5.1), residual connections (Section 2.5.2), layer normalisation (Section 2.5.3), and position encodings (Section 2.5.4). The whole structure of the transformer is shown in Section 2.5.5 and the application of pretrained transformers is mentioned in Section 2.5.6.

2.5.1 Self-Attention Mechanism

The self-attention mechanism computes a weighted representation of each token by considering all other tokens in the same sequence by an attention mechanism. Here we take the scale dot-product attention mechanism (Equation 2.9) as an example. The queries, keys and values are the projection of the input sequence $\boldsymbol{X}$ with learnable parameters $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_V$, which can be formulated as

$$\texttt{SelfAttention}(\boldsymbol{X}) = \phi_{\texttt{softmax}}\left(\frac{\boldsymbol{X}\boldsymbol{W}_Q(\boldsymbol{X}\boldsymbol{W}_K)^T}{\sqrt{d_k}}\right)\boldsymbol{X}\boldsymbol{W}_V, \tag{2.10}$$

where the representations of the input sequence is $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots)$, the learnable parameters $\boldsymbol{W}_Q \in \mathbb{R}^{d_{model}\times d_k}$, $\boldsymbol{W}_K \in \mathbb{R}^{d_{model}\times d_k}$, $\boldsymbol{W}_V \in \mathbb{R}^{d_{model}\times d_v}$, and $d_{model} = d_k = d_v$.

A large attention weight α_{ij} from token i to token j means token i attends to token j. The attention relationship can be asymmetric ($\alpha_{ij} \neq \alpha_{ji}$) because the query weights $\boldsymbol{W}_Q$ and key weights $\boldsymbol{W}_K$ may be different and the matrix multiplication is not commutative.

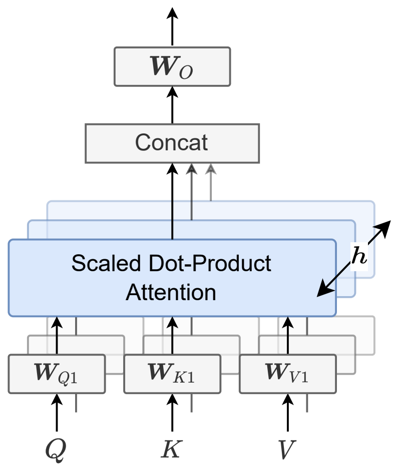


FIGURE 2.6: Multi-Head Attention

Multi-head attention The multi-head attention, as shown in Figure 2.6, is composed of multiple attention units, which can attend to different information, e.g. long-term or short-term relationships. It can be formulated as:

$$\begin{aligned} \texttt{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \texttt{Concat}(\boldsymbol{head}_1, \ldots, \boldsymbol{head}_h)\boldsymbol{W}_O \\ \boldsymbol{head}_i &= \texttt{Attention}(\boldsymbol{Q}\boldsymbol{W}_{Qi}, \boldsymbol{K}\boldsymbol{W}_{Ki}, \boldsymbol{V}\boldsymbol{W}_{Vi}), \end{aligned} \tag{2.11}$$

where h is the number of attention units, and $\boldsymbol{W}_{Qi}, \boldsymbol{W}_{Ki}, \boldsymbol{W}_{Vi}$, and $\boldsymbol{W}_O$ are projection matrices, which are learnable parameters.

2.5.2 Residual Connection

A residual connection (He et al., 2016) is a connection that skips a subnetwork $f_{sub}(\boldsymbol{x})$, performing identity mappings and adding to the output of the subnetwork, which can be formulated as

$$\texttt{Res}(\boldsymbol{x}) = \boldsymbol{x} + f_{sub}(\boldsymbol{x}). \tag{2.12}$$

Deep neural networks, which have tens or hundreds of layers, with residual connections are easier to train and get rid of the problem of gradient vanishing (see Appendix A). Without the residual connections, very deep transformers cannot be trained (Dong et al., 2021).

2.5.3 Layer Normalisation

During the training of a deep neural network, the distribution of inputs between each layer may change because the parameters of the model are updated, which is called *internal covariate shift* (Ioffe and Szegedy, 2015). This phenomenon may make training a deep neural network unstable and difficult. The layer normalisation (Ba et al., 2016) is one solution.

For an input vector $\boldsymbol{x} = [x_i]_{i=1}^d$ with dimension d, the layer normalisation calculates mean μ and variance σ as follows:

$$\begin{aligned}\mu &= \frac{1}{d}\sum_{i=1}^{d} x_i \\ \sigma &= \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i-\mu)^2+\epsilon},\end{aligned} \tag{2.13}$$

where ϵ is a small value added to the denominator for numerical stability, then the output of the layer normalisation is

$$\texttt{Norm}(\boldsymbol{x}) = \frac{\boldsymbol{w}}{\sigma}(\boldsymbol{x}-\mu)+\boldsymbol{b}, \tag{2.14}$$

where $\boldsymbol{w}$ and $\boldsymbol{b}$ are trainable parameters which are set to be $\mathbf{1}$ and $\mathbf{0}$ respectively in general.

2.5.4 Positional encoding

As described above, the transformers are composed of pure attention mechanisms without recurrent units. As a result, they do not have an inherent sense of sequential order. Therefore, positional encoding is introduced to provide information on the relevant or absolute position of each token, which can be learned or fixed. The positional encoding method used by Vaswani et al. (2017) is sine and cosine functions with different frequencies. For token in the position j represented by d-dimensional $\boldsymbol{x}_j = [x_{j,i}]_{i=1}^d$, its positional encoding $\boldsymbol{p}_j = [p_{j,i}]_{i=1}^d$ can be formulated as follows:

$$p_{j,i} = \begin{cases}\sin(\omega_k \cdot j), & \text{if } i = 2k \\ \cos(\omega_k \cdot j), & \text{if } i = 2k+1,\end{cases} \tag{2.15}$$

where $\omega_k = \frac{1}{10000^{2k/d}}$ and $k = 0, 1, \dots, \lceil \frac{d}{2} \rceil - 1$. This positional encoding can properly represent the absolute and relative position information, allowing the model to learn the sequence order information.

Absolute position information As shown in Equation 2.15, the elements in the positional encoding $\boldsymbol{p}$ are sine and cosine functions with different periods:

$$2\pi, 2\pi\cdot 10000^{\frac{2}{d}}, 2\pi\cdot 10000^{\frac{4}{d}}, \dots, 2\pi\cdot 10000^{1-\frac{2}{d}}.$$

If the sequence length is shorter than $10000^{1-\frac{2}{d}}$, the positional encoding of tokens in each position is unique, giving the model the absolute position information.

Relative position information For any fixed offset $\delta \in \mathbb{N}$, a linear transformation $\boldsymbol{T}^\delta \in \mathcal{R}^{d\times d}$ exists, such that

$$\boldsymbol{T}^\delta \boldsymbol{p}_j = \boldsymbol{p}_{j+\delta}. \tag{2.16}$$

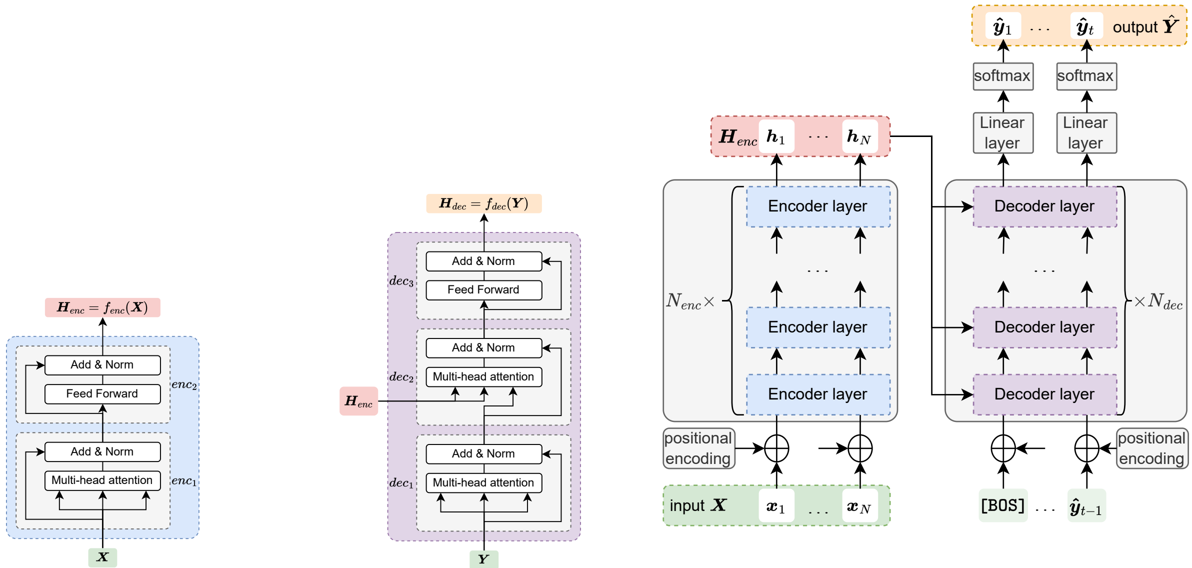
In addition, the length of each positional encoding is the same

$$\|\boldsymbol{p}_j\|_2 = \sqrt{\frac{d}{2}} \tag{2.17}$$

and the distance between two tokens is symmetric (see Appendix B.1).

$$\|\boldsymbol{p}_j - \boldsymbol{p}_{j+\delta}\|_2 = \|\boldsymbol{p}_j - \boldsymbol{p}_{j-\delta}\|_2 \tag{2.18}$$

2.5.5 Encoder-decoder structure of the Transformer



(A) The encoder layer (B) The decoder layer (C) The encoder-decoder structure of transformers. [BOS] is a vector representing the start of a sequence.

FIGURE 2.7: The encoder layer f_{enc}, the decoder layer f_{dec}, and the encoder-decoder structure of transformers.

The Transformer is composed of the encoder and the decoder (see Figure 2.7 (C)). The encoder is composed of identical encoder layers f_{enc} and the decoder is also composed of identical decoder layers.

Encoder layer The encoder layer f_{enc} is composed of two sublayers (see Figure 2.7 (A)). The first sublayer $\texttt{enc}_1$ takes the input sequence $\boldsymbol{X} = (\boldsymbol{x}_i)_{i=1}^N$ as the queries, keys and values of a multi-head attention layer (Equation 2.11), following by a residual connection (Equation 2.12) and a layer normalisation (Equation 2.14). The second sublayer $\texttt{enc}_2$ consists of a feed-forward network f_{ffn} with a residual connection and a layer normalisation. The encoder layer can be formulated as follows:

$$\begin{aligned} \boldsymbol{X}' &= \texttt{enc}_1(\boldsymbol{X}) = \texttt{Norm}(\boldsymbol{X} + \texttt{MultiHead}(\boldsymbol{X}, \boldsymbol{X}, \boldsymbol{X})) \\ \boldsymbol{H}_{enc} &= \texttt{enc}_2(\boldsymbol{X}') = \texttt{Norm}(\boldsymbol{X}' + f_{ffn}(\boldsymbol{X}')) \end{aligned} \tag{2.19}$$

where $\boldsymbol{X}'$ is the output of the first sublayer $\texttt{enc}_1$ and $\boldsymbol{H}_{enc} = (\boldsymbol{h}_i)_{i=1}^N$ is the output of the encoder layer f_{enc}.

Decoder layer The decoder layer f_{dec} includes three sublayers (see Figure 2.7 (B)). The first sublayer $\texttt{dec}_1$ and the second sublayer $\texttt{dec}_2$ are composed of a multi-head attention layer with a residual connection and layer normalisation. In the first sublayer, the queries, keys and values are $\boldsymbol{Y} = (\boldsymbol{y}_i)_{i=1}^M$, but the second sublayer takes the output of the first sublayer as queries and the output from the last encoder layer as keys and values. The third sublayer $\texttt{dec}_3$ includes a feed-forward network f_{ffn} with a residual connection and a layer normalisation. The decoder layer can be formulated as follows:

$$\begin{aligned} \boldsymbol{Y}' &= \texttt{dec}_1(\boldsymbol{Y}) = \texttt{Norm}(\boldsymbol{Y} + \texttt{MultiHead}(\boldsymbol{Y}, \boldsymbol{Y}, \boldsymbol{Y})) \\ \boldsymbol{Y}'' &= \texttt{dec}_2(\boldsymbol{Y}') = \texttt{Norm}(\boldsymbol{Y}' + \texttt{MultiHead}(\boldsymbol{Y}', \boldsymbol{H}_{enc}, \boldsymbol{H}_{enc})) \\ \boldsymbol{H}_{dec} &= \texttt{dec}_3(\boldsymbol{Y}'') = \texttt{Norm}(\boldsymbol{Y}'' + f_{ffn}(\boldsymbol{Y}'')) \end{aligned} \tag{2.20}$$

where $\boldsymbol{Y}', \boldsymbol{Y}''$ are the output of the first and second sublayer $\texttt{dec}_1, \texttt{dec}_2$ respectively and $\boldsymbol{H}_{dec} = (\boldsymbol{h}_i)_{i=1}^{N}$ is the output of the decoder layer f_{dec}.

As shown in Figure 2.7 (C), the encoder is composed of N_{enc} encoder layers and the decoder includes N_{dec} decoder layers. The input sequence $\boldsymbol{X} = (\boldsymbol{x}_i)_{i=1}^{N}$ with N tokens added with the positional encoding (Equation 2.15) is fed into the encoder, then the encoder generates $\boldsymbol{H}_{enc}$, which represents the input sequence with context information.

In the decoding step t, the decoder takes the encoded feature of the input sequence $\boldsymbol{H}$ and the generated output sequence $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{y}}_i)_{i=1}^{t-1}$ added with the positional encoding as input, where $\boldsymbol{H}$ is fed into each decoder layer directly. The output of the final decoder layer passes through a linear layer and a softmax function (Equation 2.4), which represents the probability distribution of output token t over the whole vocabulary.

In summary, the transformer can capture the relationship between two tokens at a long distance with multi-head attention. Including the residual connection and layer normalisation, the transformer can be deep and stable for training. Therefore, it can model sequential tasks properly.

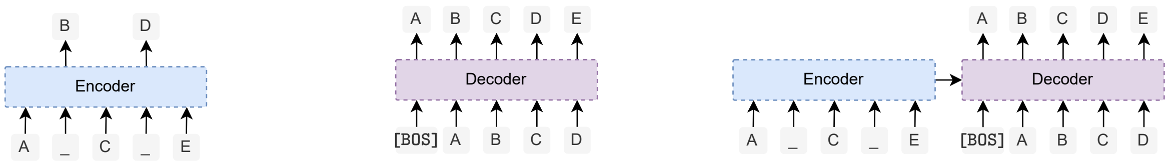
2.5.6 Pre-training and Fine-tuning for Downstream Tasks

Transformer-based language models are trained unsupervisedly or self-supervisedly on unlabeled corpora as initialisation. The pre-training methods can be divided into three groups, masked language modelling, next-token prediction, and sequence reconstruction.

- **Masked Language Modelling** is commonly used to pre-train encoder-only models, e.g. BERT (Devlin et al., 2019). As shown in Figure 2.8 (A), a portion of the tokens in the input sequence is randomly masked. The model should learn to predict masked tokens based on the left and right context information within the sequence, which means tokens are encoded bi-directionally, i.e. in an input sequence $X = (\boldsymbol{x})_{i=1}^{N}$, which has N tokens, a token $\boldsymbol{x}_i$ can attend to future tokens $\boldsymbol{x}_j$ where $j > i$, and past tokens $\boldsymbol{x}_j$ where $j < i$. However, without recurrent units, these models cannot be used for generation easily.

- **Next-Token Prediction** is often used to pre-train decoder-only models, e.g. GPT (Radford et al., 2018). As shown in Figure 2.8 (B), the model should learn to predict the next token according to the given sequence. As a result, the next token prediction is suitable for pre-training generative models. However, since tokens can only be conditioned on leftward tokens, the model is not able to learn bi-directional relationships.

- **Sequence Reconstruction** can be used to pre-train encoder-decoder models, e.g. BART (Lewis et al., 2020). It combines the concept of masked language modelling and next-token prediction. As shown in Figure 2.8 (C), an input sequence is masked partially and compressed by an encoder, and then the decoder should reconstruct the input sequence. The input sequence does not need to be aligned with the decoder outputs, which means arbitrary noise transformation can apply to the input sequence, e.g. masking or removing certain tokens. By this method, the model can learn both bi-directional contextual relationships and it is able to generate coherent sequences.



(A) Masked Language Modelling for encoder-only models

(B) Next Token Prediction for decoder-only models

(C) Sequence reconstruction for encoder-decoder models

FIGURE 2.8: Different structures and pre-training methods of transformer-based language models. The example input sequence includes 5 tokens, $A, B, C, D,$ and E, [BOS] represents the start of a sequence, and "_" stands for masked tokens.

These language models are built for general purposes and can capture contextual word representations (Reif et al., 2019) where each layer may learn linguistically founded information (Niu et al., 2022; Vries et al., 2020). In addition, transformer-based models pre-trained on text data also improve their performance on non-text sequential tasks, such as protein classification or music composer classification (Kao and Lee, 2021). Therefore, fine-tuning pre-trained transformer-based models for specified downstream tasks is appealing (Chiang et al., 2022). The pre-trained model can be directly optimised on the target dataset with a small learning rate, update part of the model by an adaptor (Hu et al., 2022), leverage prefix-tuning (He et al., 2022) or prompt-tuning (Lester et al., 2021) to modify part of the hidden layers. In this thesis, user simulators are built on pre-trained large language models, e.g. BART (Lewis et al., 2020), fine-tuning the language model on dialogue corpora with different representations of input and output sequences.

Large Language Models Large language models (LLMs) are language models that have millions to trillions of parameters and are pre-trained on comprehensive corpora by unsupervised or self-supervised learning. LLM-based systems have been quickly deployed to real users and achieve impressive performance, comparable to or even surpassing humans, across a wide range of natural language process tasks (Brown et al., 2020). The remarkable capabilities of LLMs can often be attributed to in-context learning (Brown et al., 2020; Dong et al., 2023) and reinforcement learning through human feedback (Ouyang et al., 2022). With in-context learning, users can teach LLMs to perform a specific task by providing some input-output examples during inference, without optimising any parameters of LLMs. In addition, reinforcement learning through human feedback (RHLF) improves LLMs to follow user instructions, achieving better performance on in-context learning.

As shown in Figure 2.8, LLMs can be divided into three groups according to their structure, *encoder-only*, e.g. BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), *encoder-decoder*, e.g. Transformer (Vaswani et al., 2017), BART (Lewis et al., 2020), T5 (Raffel et al., 2020), and AlexaTM (Soltan et al., 2022), and *decoder-only*, e.g. GPT-family models (Brown et al., 2020; OpenAI, 2023; Radford et al., 2018; Solaiman et al., 2019; Wang and Komatsuzaki, 2021), Gopher (Rae et al., 2022), LaMDA (Thoppilan et al., 2022), Chinchilla (Hoffmann et al., 2022), BLOOM (Scao et al., 2022), PaLM (Chowdhery et al., 2022), and LLaMA (Touvron et al., 2023).

The model size and the training cost of large language models are growing exponentially (see Figure 2.9). The training cost C denotes how many floating point operations (FLOP) are required for training the model, which is commonly approximated as $C = 6D\Theta$ where D represents how many tokens there are in the training corpus and Θ stands for the number of the parameters of the model (Kaplan et al., 2020). The serving cost and latency time also increase considerably as the model size grows, therefore it is still very challenging to properly leverage the power of large language models.

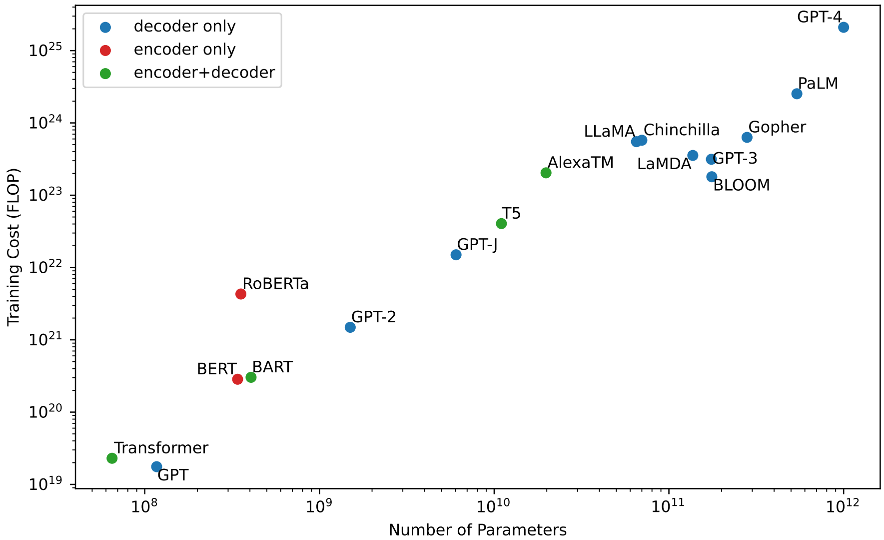


FIGURE 2.9: The parameters and training cost of various large language models.

2.6 Neural Network Optimisation

2.6.1 Gradient Descent

The optimisation algorithm changes the parameters of the model (e.g. the weights and biases of a neural network) to minimise the errors based on a loss function (Section 2.6.2) iteratively. The most basic optimisation algorithm is *Gradient Descent*, which can be formulated as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}; \boldsymbol{\theta}_k), \tag{2.21}$$

where $\boldsymbol{\theta}_k$ is the parameters of the model at epoch k, $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}; \boldsymbol{\theta}_k)$ is the gradient of the loss function $\mathcal{L}$ with respect to the parameters $\boldsymbol{\theta}_k$ based on the whole dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ which is composed of N pairs of input $\boldsymbol{x}$ and target $\boldsymbol{y}$, and η is the learning rate which is a small positive number and controls the step-size of update. The learning rate η should be selected carefully since a high learning rate can hinder convergence and an excessively small learning rate can lead to time-consuming training or suboptimal parameters.

The gradient descent algorithm updates parameters based on the whole dataset, also called **Batch Gradient Descent**. When dealing with sizable datasets, it can lead to lengthy training times and memory-intensive computations. In addition, it may be trapped in local minima. **Stochastic Gradient Descent** is a variant of the gradient descent. Instead of optimising parameters for the whole dataset $\mathcal{D}$, the stochastic gradient descent computes gradients and updates parameters for each individual data point, $\mathcal{D}_i = (\boldsymbol{x}_i, \boldsymbol{y}_i)$, one at a time. While stochastic gradient descent requires less memory than batch gradient descent, frequent updates can result in noisy gradients and cannot fully utilise parallel computation capabilities. **Mini-batch Gradient Descent** is a compromise, combining the concept of both batch gradient descent and stochastic gradient descent. It splits the complete dataset into small batches, i.e. mini-batch, and updates the model on each mini-batch. By selecting an appropriate batch size, a balance between memory efficiency and computational speed can be achieved. As shown in Algorithm 1, for a dataset $\mathcal{D}$ with N data points, the batch gradient descent has batch size $N_{\text{batch}} = N$, the stochastic gradient descent has batch size $N_{\text{batch}} = 1$, and the batch size of mini-batch gradient descent is $0 < N_{\text{batch}} < N$.

Algorithm 1 Gradient Descent. The batch gradient descent updates parameters based on the whole dataset, the stochastic gradient descent updates parameters per each data point, and the mini-batch gradient descent updates parameters per each mini-batch data.

Input: Learning rate η, Batch size N_{batch}, Number of epochs N_{epochs}
Input: Training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, Model parameters $\boldsymbol{\theta}$
Output: Optimised parameters $\boldsymbol{\theta}$
Initialise $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ randomly
for $k \leftarrow 1$ **to** N_{epochs} **do**
  Shuffle the training dataset $\mathcal{D}$
  Initialise $j = 0$
  while $j < N$ **do**
    $\mathcal{D}_{\text{batch}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=j}^{j+N_{\text{batch}}}$ (batch data)
    $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}_{\text{batch}}; \boldsymbol{\theta}_{k-1})$ (update parameters)
    $j \leftarrow j + N_{\text{batch}}$
  end while
end for
Return Optimised $\boldsymbol{\theta}^*$

2.6.2 Loss function

The loss function $\mathcal{L}$, also referred to as objective function or cost function, quantifies the difference between the predicted values generated by a model and the actual ground truth or target values in a dataset, measuring how well or poorly a model is performing on a given task. The choice of an appropriate loss function depends on the specific task and the desired characteristics of the model output. For example, the Mean Squared Error (MSE) is often used for regression tasks, which can be formulated as:

$$\mathcal{L}_{\texttt{MSE}}(\mathcal{D}; \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 , \tag{2.22}$$

where the dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$ includes N pairs of input-output pairs $(\boldsymbol{x}_i, y_i)$ and the model parameterised by $\boldsymbol{\theta}$ outputs $\hat{y}_i = f(\boldsymbol{x}_i; \boldsymbol{\theta})$ for the input $\boldsymbol{x}_i$. The mean squared error loss function encourages the model's output closer to actual data. For classification problems, the

Cross-Entropy Loss (CE) is commonly used, which can be formulated as:

$$\mathcal{L}_{\texttt{CE}}(\mathcal{D};\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\left(y_{ij}\log\hat{y}_{ij}\right), \tag{2.23}$$

where the target $\boldsymbol{y} = [y_j]_{j=1}^{C}$ is a one-hot encoding for C classes, i.e. if $\boldsymbol{x}_i$ belongs to class c, then

$$y_{i,j} = \begin{cases} 1 & \text{if } j = c \\ 0 & \text{otherwise} \end{cases} \tag{2.24}$$

and $\hat{\boldsymbol{y}} = f(\boldsymbol{x};\boldsymbol{\theta})$ represents the probability scores of different classes predicted by the model, i.e. $\hat{y}_{ij} = P(\boldsymbol{x}_i = j)$ is the probability of $\boldsymbol{x}_i$ belongs to class j. The cross-entropy loss function encourages the output probability distribution aligned with the target labels.

2.6.3 Adaptive Subgradient Methods

Gradient descent methods have a fixed learning rate, although different parameters may benefit from varying learning rates. Furthermore, the large gradients in the initial epochs can detrimentally impact optimisation, emphasising the need for a smaller learning rate for larger gradients. Adaptive Subgradient methods (AdaGrad) dynamically adjust updates to each individual parameter, allowing for larger or smaller updates based on their respective importance (Duchi et al., 2011). Given the parameters of the model $\boldsymbol{\theta}$, the loss function $\mathcal{L}$, the dataset for optimisation $\mathcal{D}$, AdaGrad can be formulated as follows:

$$\begin{aligned} \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k - \frac{\eta}{\sqrt{\sum_{i=1}^{k}\boldsymbol{g}_i^2}}\boldsymbol{g}_k \\ \boldsymbol{g}_k &= \nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathcal{D};\boldsymbol{\theta}_k), \end{aligned} \tag{2.25}$$

where the learning rate η is normalised by the summation of history gradients and $\boldsymbol{g}^2 = \boldsymbol{g}\odot\boldsymbol{g}$ is the element-wise square of gradient $\boldsymbol{g}$. AdaGrad automatically adjusts the learning rate for each training parameter, but one drawback is that the learning rate continually decreases because $\sum_{i=1}^{k}\boldsymbol{g}_i^2$ is a monotonic increasing function, leading to slower training.

2.6.4 Adaptive Moment Estimation

The adaptive moment estimation (Adam) (Kingma and Ba, 2015) is the most commonly used optimisation algorithm. Adam is inspired by AdaGrad and RMSProp (Hinton et al., 2012), computing adaptive learning rates for each parameter and incorporating the concept of momentum to overcome local minima. Given the parameters of the model $\boldsymbol{\theta}$, the loss function $\mathcal{L}$ and the data for optimisation $\mathcal{D}$, the gradient at epoch k is $\boldsymbol{g}_k = \nabla\mathcal{L}(\mathcal{D};\boldsymbol{\theta}_{k-1})$, the Adam algorithm updates the moving average of the gradients $\boldsymbol{m}_k$ and the squared gradients $\boldsymbol{v}_k$, which can be formulated as follows:

$$\begin{aligned} \boldsymbol{m}_k &= \beta_1\cdot\boldsymbol{m}_{k-1} + (1-\beta_1)\cdot\boldsymbol{g}_k \\ \boldsymbol{v}_k &= \beta_2\cdot\boldsymbol{v}_{k-1} + (1-\beta_2)\cdot\boldsymbol{g}_k^2, \end{aligned} \tag{2.26}$$

where $\boldsymbol{g}^2$ is the element-wise square of gradient $\boldsymbol{g}$ and the hyperparameters $\beta_1, \beta_2 \in (0, 1)$ controls the exponential decay of these moving averages. $\boldsymbol{m}$ estimates the mean and $\boldsymbol{v}$ estimates the uncentered variance of the gradients. However, since $\boldsymbol{m}$ and $\boldsymbol{v}$ are initialised to 0, i.e. $\boldsymbol{m}_0 = \mathbf{0}, \boldsymbol{v}_0 = \mathbf{0}$, they are biased to zero, especially during the initial training epochs with small decay rates, i.e. β_1, β_2 are close to 1. To overcome this issue, the bias-corrected estimates $\hat{\boldsymbol{m}}_k, \hat{\boldsymbol{v}}_k$ are formulated as follows:

$$
\begin{aligned}
\hat{\boldsymbol{m}}_k &= \frac{\boldsymbol{m}_k}{1-\beta_1^k} \\
\hat{\boldsymbol{v}}_k &= \frac{\boldsymbol{v}_k}{1-\beta_2^k}
\end{aligned}
\tag{2.27}
$$

where β_1^k, β_2^k are β_1, β_2 to the power k. More details for bias correction can be found in Appendix B.2. The parameters $\boldsymbol{\theta}$ are optimised by

$$
\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta \frac{\hat{\boldsymbol{m}}_k}{\sqrt{\hat{\boldsymbol{v}}_k} + \epsilon} \tag{2.28}
$$

where η is the learning rate, ϵ is set to prevent divide by 0. The pseudo-code of Adam is shown in Algorithm 2, where the default hyperparameters are $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ and the default learning rate is $\eta = 0.001$.

Algorithm 2 Adam

Input: Hyperparameters $\beta_1, \beta_2, \epsilon$, Number of epochs N_{epochs}
Input: Learning rate η, Model parameters $\boldsymbol{\theta}$, Dataset $\mathcal{D}$
Output: Optimised parameters $\boldsymbol{\theta}$
Initialise $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ randomly
Initialise $\boldsymbol{m}_0 = \mathbf{0}, \boldsymbol{v}_0 = \mathbf{0}$
for $k \leftarrow 1$ **to** N_{epochs} **do**
    $\boldsymbol{g}_k = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}; \boldsymbol{\theta}_{k-1})$ (calculate the gradients)
    $\boldsymbol{m}_k = \beta_1 \cdot \boldsymbol{m}_{k-1} + (1-\beta_1) \cdot \boldsymbol{g}_k$ (estimates the mean of the gradients)
    $\boldsymbol{v}_k = \beta_2 \cdot \boldsymbol{v}_{k-1} + (1-\beta_2) \cdot \boldsymbol{g}_k^2$, (estimates the uncentered variance of the gradients)
    $\hat{\boldsymbol{m}}_k = \frac{\boldsymbol{m}_k}{1-\beta_1^k}$ (bias correction of $\boldsymbol{m}_k$)
    $\hat{\boldsymbol{v}}_k = \frac{\boldsymbol{v}_k}{1-\beta_2^k}$ (bias correction of $\boldsymbol{v}_k$)
    $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta \frac{\hat{\boldsymbol{m}}_k}{\sqrt{\hat{\boldsymbol{v}}_k}+\epsilon}$ (update parameters)
end for
Return Optimised $\boldsymbol{\theta}^*$

2.7 Reinforcement Learning

Reinforcement learning has achieved impressive breakthroughs in various areas, such as self-driving (Kiran et al., 2022), robot control (Kober et al., 2013), and AlphaGo (Silver et al., 2016), a computer program that defeated professional human Go players. The key concept of reinforcement learning is shown in Figure 2.10, where an agent interacts with the environment in discrete time steps. In each time step t the agent is in the state $\boldsymbol{s}_t$, takes an action $\boldsymbol{a}_t$, obtains a reward r_{t+1} from the environment and makes a transition to the next state s_{t+1}. The goal of the agent is to maximise the sum of rewards, learning which action to take through trial and error.

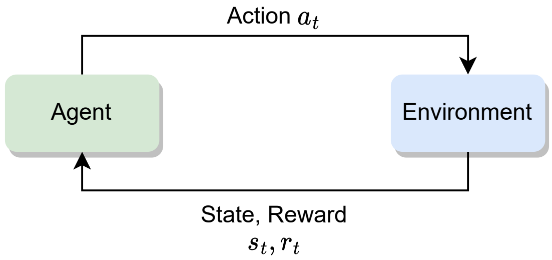


FIGURE 2.10: The key concept of reinforcement learning: the agent takes actions and interacts with the environment, learning from trial and error.

Terminology The **state** s describes the state of the environment. It can be fully observed without any hidden information, e.g. the state in a Go game, or partially observed, e.g. the dialogue state where the user goal is unseen. The **policy** π decides which **action** a_t the agent should take in turn t based on state s_t:

$$a_t \sim \pi(\cdot|s_t) \tag{2.29}$$

A **trajectory** $\tau = (s_0, a_0, s_1, a_1, \dots)$ is a sequence of states and actions, where the probability of a T-turn trajectory τ is:

$$P(\tau|\pi) = P_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t), \tag{2.30}$$

where P_0 is the **starting state probability** of the initial state s_0 and $P(s_{t+1}|s_t, a_t)$ is the **transition probability**, representing the probability of moving to state s_{t+1} after taking action a_t in state s_t. The **reward function** R gives the reward signal based on the state and the agent's action, e.g. $r_t = R(s_t, a_t)$. The **return**, cumulative rewards over a trajectory τ, is often modelled as the summation of all rewards with a discounted factor $\gamma \in (0, 1)$

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{2.31}$$

With the discounted factor, the agent will focus more on the current reward and the sum of all rewards will converge to a finite value. The goal of reinforcement learning is to maximise the expected return $J(\pi)$:

$$J(\pi) = \int_\tau P(\tau|\pi)R(\tau) = E_{\tau\sim\pi}[R(\tau)] \tag{2.32}$$

with the optimal policy π^*

$$\pi^* = \arg\max_\pi J(\pi) \tag{2.33}$$

Value Functions Value functions are used to represent the expected return an agent can get when it starts in the state s or state-action pair (s, a) and then acts according to its policy.

The **state-value function**, which calculates the expected return if the agent starts in state $\boldsymbol{s}$ and then acts according to its policy π, can be formulated as

$$V^{\pi}(s) = E_{\tau\sim\pi}[R(\tau)|\boldsymbol{s}_0 = s], \tag{2.34}$$

the **action-value function**, which gives the expected return if the agent starts in state $\boldsymbol{s}$, takes the action $\boldsymbol{a}$, and then acts according to its policy π, can be formulated as

$$Q^{\pi}(s,a) = E_{\tau\sim\pi}[R(\tau)|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}], \tag{2.35}$$

and the **advantage function**, which estimates how much better an action is than the others on average, can be formulated as

$$A^{\pi}(\boldsymbol{s},\boldsymbol{a}) = Q^{\pi}(\boldsymbol{s},\boldsymbol{a}) - V^{\pi}(\boldsymbol{s}) \tag{2.36}$$

2.7.1 Policy Gradient Optimisation

In the policy gradient optimisation algorithms, the agent's policy π is parameterised by $\boldsymbol{\theta}$. To maximise the expected return $J(\pi_{\boldsymbol{\theta}}) = E_{\tau\sim\pi_{\boldsymbol{\theta}}}[R(\tau)]$, the policy $\pi_{\boldsymbol{\theta}}$ can be optimised by the gradient ascent

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}_k}), \tag{2.37}$$

where η is the learning rate and k is the learning step. We can derive the policy gradient as follows:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) &= \nabla_{\boldsymbol{\theta}} E_{\tau\sim\pi_{\boldsymbol{\theta}}}[R(\tau)] \\
&= \nabla_{\boldsymbol{\theta}} \int_{\tau} P(\tau|\pi_{\boldsymbol{\theta}})R(\tau)d\tau \\
&= \int_{\tau} \nabla_{\boldsymbol{\theta}} P(\tau|\pi_{\boldsymbol{\theta}})R(\tau)d\tau \\
&= \int_{\tau} P(\tau|\pi_{\boldsymbol{\theta}})\nabla_{\boldsymbol{\theta}} \log P(\tau|\pi_{\boldsymbol{\theta}})R(\tau)d\tau \\
&= E_{\tau\sim\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log P(\tau|\pi_{\boldsymbol{\theta}})R(\tau)],
\end{aligned} \tag{2.38}$$

where the gradient of the log-probability of a trajectory τ can be derived as follows:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log P(\tau|\pi_{\boldsymbol{\theta}}) &= \cancel{\nabla_{\boldsymbol{\theta}} \log P_0(\boldsymbol{s}_0)} + \sum_{t=0}^{T}(\cancel{\nabla_{\boldsymbol{\theta}} \log P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)} + \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)) \\
&= \sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)
\end{aligned} \tag{2.39}$$

According to Equation 2.38 and Equation 2.39, we can get

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = E_{\tau\sim\pi_{\boldsymbol{\theta}}}[\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)R(\tau)]. \tag{2.40}$$

The reward function can also be replaced by the advantage function (Schulman et al., 2016)

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = E_{\tau \sim \pi_{\boldsymbol{\theta}}}[\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t) A^{\pi_{\boldsymbol{\theta}}}] \tag{2.41}$$

However, without any constraint or penalty, the gradient can be excessively large. It may cause the training unstable and suboptimal performance (Schulman et al., 2017).

2.7.2 Proximal Policy Optimisation

To get rid of large policy updates which may destroy the optimisation, Schulman et al. (ibid.) apply a specialised clipping on the loss function

$$\begin{aligned} \rho(\boldsymbol{\theta}) &= \frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\boldsymbol{\theta}'}(\boldsymbol{a}_t|\boldsymbol{s}_t)} \\ \mathcal{L}^{CLIP}(\boldsymbol{\theta}) &= E_{\tau \sim \pi_{\boldsymbol{\theta}}}[\min(\rho(\boldsymbol{\theta})A^{\pi_{\boldsymbol{\theta}}}, g(\epsilon, A^{\pi_{\boldsymbol{\theta}}}))] \end{aligned} \tag{2.42}$$

where $\epsilon \in (0, 1)$ is a hyperparameter with a small value which constrains how far the new policy is able to go from the old policy, the parameters of the new and the old policy are $\boldsymbol{\theta}, \boldsymbol{\theta}'$ respectively, $\rho(\boldsymbol{\theta})$ is the ratio of the probability of action $\boldsymbol{a}$ given state $\boldsymbol{s}$ for the new and old policy $\pi_{\boldsymbol{\theta}}, \pi_{\boldsymbol{\theta}'}$ and

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & \text{otherwise} \end{cases} \tag{2.43}$$

The behaviour of $\mathcal{L}^{CLIP}(\boldsymbol{\theta})$ is summarised in Table 2.1. There is no clipping when $\rho(\boldsymbol{\theta}) \in [1-\epsilon, 1+\epsilon]$. The clipping happens in two situations, the first one is when A^{π_θ} is negative and $\rho(\boldsymbol{\theta})$ exceeds the lower boundary, i.e. $\rho(\boldsymbol{\theta}) < 1-\epsilon$, and the second one is when A^{π_θ} is positive and $\rho(\boldsymbol{\theta})$ exceeds the upper boundary, i.e. $\rho(\boldsymbol{\theta}) > 1+\epsilon$. This method prevents the algorithm from getting too greedy. Taking the first scenario as an example, although the advantage estimate implies the probability of taking action $\boldsymbol{a}$ given state $\boldsymbol{s}$ should be decreased ($A^{\pi_\theta} < 0$), there is still no update since it is already much less likely to take in the new policy than the old policy ($\rho(\boldsymbol{\theta}) < 1-\epsilon$). On the other hand, if $\rho(\boldsymbol{\theta})$ exceeds the lower boundary, there is no clipping when $A^{\pi_\theta} > 0$ and vice versa. The gradient is 0 when the loss function is clipped because the gradient is applied on $(1-\epsilon)A^{\pi_\theta}$ or $(1+\epsilon)A^{\pi_\theta}$ instead of $\rho(\boldsymbol{\theta})A^{\pi_\theta}$.

In this way, the model can be updated in two situations, the probability ratio $\rho(\boldsymbol{\theta})$ is in the interval $[1-\epsilon, 1+\epsilon]$ or the probability ratio exceeds the interval but the advantage estimate A^{π_θ} makes it closer to the interval, which can prevent the new policy from going too far away from the previous step.

The algorithm of PPO is shown in Algorithm 3. There are two neural networks that will be updated, the policy network $\pi_{\boldsymbol{\theta}}$ and the value network $V_{\boldsymbol{v}}$, where the value network is parameterised by $\boldsymbol{v}$ and trained to model the state-value function V^{π_θ} of the policy $\pi_{\boldsymbol{\theta}}$. Both networks can be updated by any gradient descent method, e.g. Adam.

$\rho(\boldsymbol{\theta}) > 0$	$A^{\pi_{\boldsymbol{\theta}}}$	return of *min* in $\mathcal{L}^{CLIP}$	$\mathcal{L}^{CLIP}$ is clipped	Gradient
$\rho(\boldsymbol{\theta}) \in [1-\epsilon, 1+\epsilon]$	$+$	$\rho(\boldsymbol{\theta})A^{\pi_{\boldsymbol{\theta}}}$	no	✓
$\rho(\boldsymbol{\theta}) \in [1-\epsilon, 1+\epsilon]$	$-$	$\rho(\boldsymbol{\theta})A^{\pi_{\boldsymbol{\theta}}}$	no	✓
$\rho(\boldsymbol{\theta}) < 1-\epsilon$	$+$	$\rho(\boldsymbol{\theta})A^{\pi_{\boldsymbol{\theta}}}$	no	✓
$\rho(\boldsymbol{\theta}) < 1-\epsilon$	$-$	$g(\epsilon, A^{\pi_{\boldsymbol{\theta}}}) = (1-\epsilon)A^{\pi_{\boldsymbol{\theta}}}$	yes	0
$\rho(\boldsymbol{\theta}) > 1+\epsilon$	$+$	$g(\epsilon, A^{\pi_{\boldsymbol{\theta}}}) = (1+\epsilon)A^{\pi_{\boldsymbol{\theta}}}$	yes	0
$\rho(\boldsymbol{\theta}) > 1+\epsilon$	$-$	$\rho(\boldsymbol{\theta})A^{\pi_{\boldsymbol{\theta}}}$	no	✓

TABLE 2.1: The behaviour of the clip loss function $\mathcal{L}^{CLIP}(\boldsymbol{\theta})$ summarised by Bick and Wiering (2021), where the probability ratio $\rho(\boldsymbol{\theta})$ and the advantage estimate $A^{\pi_{\boldsymbol{\theta}}}$ are not 0. The first column represents the value of $\rho(\boldsymbol{\theta})$ in which region and the second column shows $A^{\pi_{\boldsymbol{\theta}}}$ is positive ($+$) or negative ($-$). The third column is the return of the *min* function in $\mathcal{L}^{CLIP}$ (Equation 2.42), the fourth column indicates whether the loss function is clipped (yes) or not (no), and the fifth column mentions whether the gradient of the loss function is zero (0) or not (✓).

Algorithm 3 Proximal Policy Optimisation. The policy network $\pi^{\boldsymbol{\theta}}$ is parameterised by $\boldsymbol{\theta}$, the value network $V^{\boldsymbol{v}}$ is parameterised by $\boldsymbol{v}$, and the reward function is R.

Input: Policy network parameters $\boldsymbol{\theta}$, Value network parameters $\boldsymbol{v}$
Input: Number of epochs N_{epochs}, Number of trajectories per epoch N
Output: Optimised Policy network parameters $\boldsymbol{\theta}$
Initialise $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ and $\boldsymbol{v} = \boldsymbol{v}_0$
for $k \leftarrow 1$ **to** N_{epochs} **do**
    Run policy $\pi_{\boldsymbol{\theta}_k}$ in the environment and collect a set of trajectories $\mathcal{D}_k = \{\tau_i\}_{i=1}^{N}$
    Compute rewards $R_k(\tau)$
    Compute advantage estimates $A^{\pi_{\boldsymbol{\theta}_k}}$ based on the value network $V_{\boldsymbol{v}_k}$
    Update $\boldsymbol{\theta}_k$ based on the loss function $\mathcal{L}^{CLIP}(\boldsymbol{\theta})$ (Equation 2.42)
    Update $\boldsymbol{v}_k$ based on mean-squared error: $\mathcal{L}(\boldsymbol{v}) = \sum_{\tau \in \mathcal{D}_k} \sum_{i=0}^{T} (V_{\boldsymbol{v}_k}(s_i) - R_k(\tau))^2$
end for
Return Optimised $\boldsymbol{\theta}^*$

Chapter 3

User simulation in task-oriented dialogues

3.1 Overview

The quality of a user simulator significantly impacts the performance of task-oriented dialogue systems trained by reinforcement learning (Schatzmann et al., 2005). As shown in Figure 1.1, user simulators should be able to model real users' behaviour and interact with dialogue systems following given user goals. User simulators may interact with the dialogue system on the semantic level, e.g. dialogue actions, or via natural language. In addition, to fully capture the richness of user behaviour, the user simulation should not only model user extrinsic behaviour, e.g. actions or utterances, but also the intrinsic status, e.g. user persona and emotions. Figure 3.1 illustrates the difference among user simulators, showcasing whether they are data-driven models and their varying levels of domain dependency, interpretability and capibility to capture diverse aspects of user intrinsic behaviour. More details are given in the following sections.

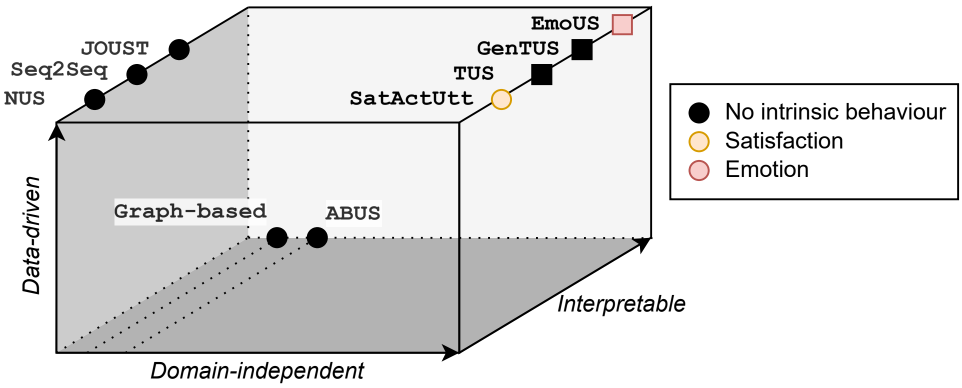


FIGURE 3.1: The difference between user simulators, comparing to which extent a model is data-driven, domain-independent and interpretable. Different degrees of user intrinsic behaviour modelling are also presented. The proposed models in this thesis are noted as squares.

3.2 Rule-based user simulators

Rule-based user simulators are commonly used in many experiments. These models are composed of various handcrafted rules, for example, Scheffler and Young (2002) combines all possible dialogue paths in a graph to build a graph-based user simulator. Despite this model being able to generate reasonable and consistent behaviour, implementing a graph-based user model for more realistic scenarios is impractical due to the extensive domain knowledge it requires.

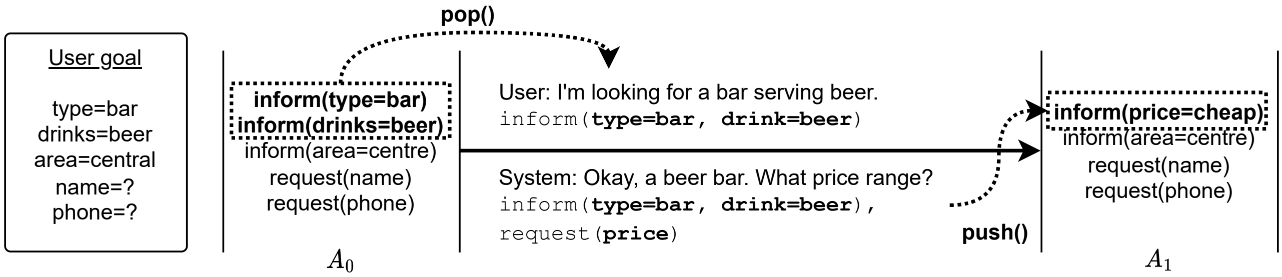


FIGURE 3.2: Sample dialogue and agenda sequence. The agenda is initialised based on the user goal. At each turn, a certain number of actions will be popped out from the agenda and new constraints might be pushed into it.

The agenda-based user simulator (ABUS) (Schatzmann et al., 2007) represents the user state as a stack-like agenda, as shown in Figure 3.2. The order of the agenda follows the user's priorities. The probability of updating the agenda and choosing user actions can be predefined manually or learned from data (Keizer et al., 2010). However, these stacking and popping rules are domain-dependent and need to be designed carefully. In addition, these rules are not able to fully capture the richness of real-user behaviour, as they overlook important aspects such as the user persona and emotions. Real-user behaviour cannot be represented by rigid rules alone.

3.3 Data-driven user simulators

Learning user behaviour from data is appealing for two reasons. Firstly, it reduces the burden of managing a considerable set of handcrafted rules, particularly in complicated and realistic scenarios. Secondly, it enables the modelling of more fine-grained user behaviour, which manually designed rules may not adequately capture.

The sequence-to-sequence (Seq2Seq) model structure is widely used to build data-driven models. El Asri et al. (2016) propose a Seq2Seq semantic-level user simulator with an encoder-decoder structure, as shown in Figure 3.3. Each turn i is represented as a vector $\boldsymbol{v}_i$ and fed into the encoder recurrent neural network (RNN), embedded as a context vector $\boldsymbol{c}$. Then this context vector $\boldsymbol{c}$ is passed to the decoder RNN to generate user actions. The feature representation of each turn is domain-dependent and the action vectors are one-hot encoding. Therefore, it is necessary to modify the feature representation and retrain the model when adapting to new domains. In addition, it only operates on the semantic level without generating natural language utterances.

Instead of generating semantic level output, Kreyssig et al. (2018) proposed the neural user simulator (NUS), that generates responses in natural language, as shown in Figure 3.4. At turn t, the feature history $\boldsymbol{V} = (v_i)_{i=1}^{t}$, where v_i is the feature of turn i, is fed into the sequence-to-sequence model. Then the model generates user utterances, which are then communicated to the dialogue system. By generating responses in natural languages instead of semantic actions, NUS requires less labelling, at the expense of interpretability. In addition, the feature representation of each turn is still domain-dependent and NUS does not consider the user intrinsic status, e.g. user persona and emotions.

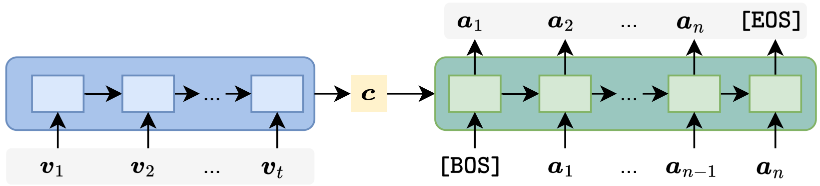


FIGURE 3.3: An RNN-based Seq2Seq user simulator. The turn i is represented as a vector v_i. The user model encodes the dialogue history as a vector c and generates user semantic actions $A = (a_i)_{i=1}^n$ where n is the action length.

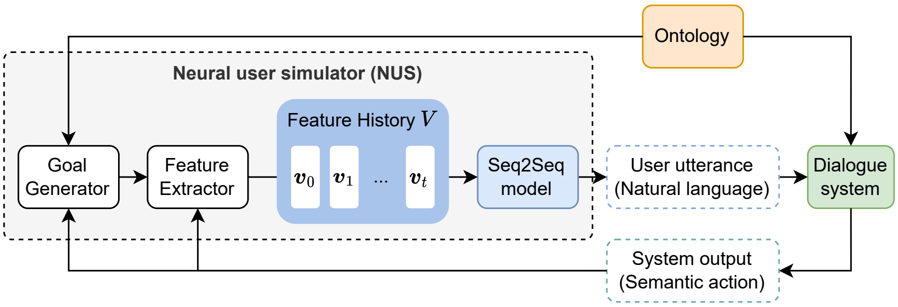


FIGURE 3.4: The architecture of the Neural User Simulator (Kreyssig et al., 2018). At the beginning of each dialogue, a goal is generated by the goal generator, which might change during the conversation. At turn t, NUS takes the feature history $V = (v_i)_{i=1}^t$, where v_i represents information of turn i, as input and generates the user utterance.

Tseng et al. (2021) proposed an end-to-end user simulator (JOUST) that has the capability to generate both dialogue acts and natural language utterances. As shown in Figure 3.5, the user goal is updated by the user actions and the goal at turn i is represented as a binary vector s_i, i.e. each dimension corresponds to a specific domain-slot pair in the ontology. To handle the context information, it has two encoders, i.e. sentence encoder and context encoder. In each turn t, the sentence encoder embeds the system utterance W_t^{sys} as a vector e_t^{sys}. At turn t, the context encoder compresses vectors of system and user utterances from previous turns into a context vector c_t. The decoder takes the goal state s_t, the system utterance embedding e_t^{sys}, and the context vector c_t as inputs and generates the user action $A_t = (a_i)_{i=1}^N$ and utterance $W_t^{usr} = (w_{t,i}^{usr})_{i=1}^{N'}$, where N and N' are the length of the user action and utterance respectively. The one-hot action vector a represents which slot to take in the ontology. The utterance is delexicalised, where specific words are replaced with special tokens in the ontology, e.g. *"I want to find a cheap hotel in the north."* is delexicalised as *"I want to find a* `[Hotel-Price]` *hotel in the* `[Hotel-Area]`*."*.

Since delexicalised utterances, the one-hot action vectors, and the binary representation of the user goal are ontology-dependent, transferring this user simulator to an unseen ontology requires additional manual effort to create lexicalisation rules and modify the feature representations of user goals and actions. In addition, the user intrinsic status is not included.

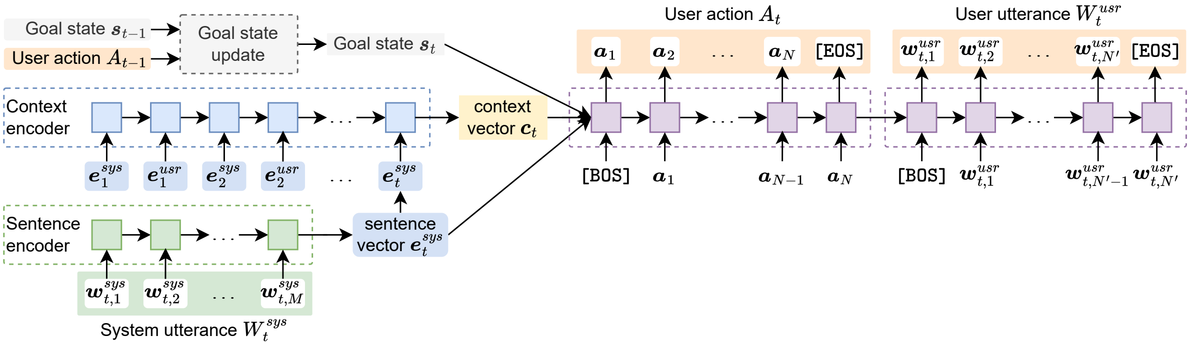


FIGURE 3.5: The end-to-end user simulator proposed by Tseng et al. (2021).

3.4 Simulating user intrinsic behaviour for task-oriented dialogues

The behaviour of human beings is influenced by their emotions (Gross, 1998). Therefore, it is important to simulate the user emotion or satisfaction level in order to capture user behaviours triggered by different emotions.

In comparison to generating responses with given emotions (Colombo et al., 2019; Mao et al., 2022; Song et al., 2019) or recognising user satisfaction after receiving user utterances (Bodigutla et al., 2020; Engelbrecht et al., 2009; Hara et al., 2010; Higashinaka et al., 2010; Schmitt and Ultes, 2015; Song et al., 2022), user satisfaction modelling should predict user intrinsic states before generating actions or utterances based on the dialogue context and user intrinsic status. Sun et al. (2021) and Deng et al. (2022) investigate how user satisfaction impacts user behaviour on the semantic level. Pan et al. (2022) transfer the emotion from chit-chat to task-oriented dialogues utilising data augmentation.

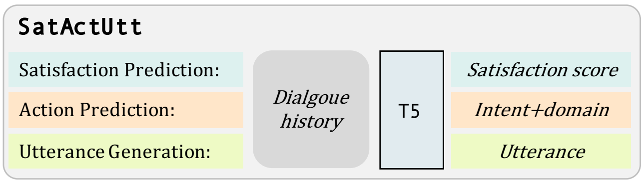


FIGURE 3.6: The model structure of SatActUtt (Kim and Lipani, 2022).

Kim and Lipani (2022) introduced a user simulator called SatActUtt based on the T5 model (Raffel et al., 2020), which utilises multi-task learning to generate users' satisfaction, action (consisting of intent and domain), and utterance based on the dialogue history with different prompts, as depicted in Figure 3.6. While SatActUtt demonstrates adequate prediction of user satisfaction scores based on dialogue history, it does not include user goals as its inputs. A dialogue system however cannot be trained in interaction with the user simulator if that user simulator does not model the user goal. Additionally, SatActUtt primarily focuses on satisfaction and dissatisfaction, disregarding other important factors such as diverse emotion elicitors or distinct user personas.

3.5 Conclusions

Rule-based user simulators are explainable and efficient for training dialogue systems, but maintaining substantial rules on more realistic scenarios is impractical and the richness of human behaviour is difficult to capture solely by certain rules. On the other hand, data-driven user simulators can learn user behaviour from the data, but collecting and labelling new datasets for unseen domains is also labour-intensive. Therefore, a user simulator should be ontology-agnostic to make it more feasible to build dialogue systems on a new domain.

In addition, the user simulator for task-oriented dialogue systems should focus not only on extrinsic behaviour, e.g. semantic actions and user utterances but also on intrinsic status, e.g. user persona or emotions. By considering the user intrinsic status, user simulators could capture diverse user behaviour triggered by different emotions and can provide more fine-grained feedback beyond task success during interaction with dialogue systems.

Chapter 4

Domain-independent User Simulation with Transformers for Task-oriented Dialogue Systems

This chapter summarises our work on domain-independent user simulation with transformers for task-oriented dialogues systems and gives a verbatim copy of our paper (Lin et al., 2021):

Hsien-chin Lin et al. (July 2021). “Domain-independent User Simulation with Transformers for Task-oriented Dialogue Systems”. In: *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Singapore and Online: Association for Computational Linguistics, pp. 445–456. URL: `https://aclanthology.org/2021.sigdial-1.47`

4.1 Summary

To remove the need to maintain intractable handcrafted rules of rule-based user simulators and redesign or retrain data-driven user simulators on new domains, we propose a domain-independent transformer-based user simulator (TUS). Firstly, we leverage the transformer-based structure, as a result, our model can capture relationships over long distances between different slots in different turns. Secondly, the feature representation of the input, e.g. the dialogue context, and the output, e.g. the value of slots in the user goal, is domain-independent. For example, we represent the value of a slot as where it comes from, e.g. it is from the user goal or mentioned by the system, instead of a one-hot representation over all possible values.

With this domain-agnostic architecture, TUS can generalise to unseen domains and learn cross-domain user behaviour from data. We compare TUS with a rule-based user simulator, agenda-based user simulator (ABUS) (Schatzmann et al., 2007), and a data-driven user simulator, variational hierarchical Seq2Seq user simulator (VHUS) (Gur et al., 2018), through both automated evaluations and human trails. TUS demonstrates competitive performance with rule-based simulators on pre-defined domains, and it can generalise to unseen domains in a zero-shot fashion

4.2 Personal contributions

All writing, implementation, and technical results are my contribution. Co-authors assisted in writing and proofreading.

Domain-independent User Simulation with Transformers for Task-oriented Dialogue Systems

Hsien-chin Lin[1], Nurul Lubis[1], Songbo Hu[2], Carel van Niekerk[1],
Christian Geishauser[1], Michael Heck[1], Shutong Feng[1], and Milica Gašić[1]

[1]Heinrich Heine University Dusseldorf, Germany
[2]Department of Computer Science and Technology, University of Cambridge, UK
[1]{linh,lubis,niekerk,geishaus,heckmi,shutong.feng,gasic}@hhu.de
[2]sh2091@cam.ac.uk

Abstract

Dialogue policy optimisation via reinforcement learning requires a large number of training interactions, which makes learning with real users time consuming and expensive. Many set-ups therefore rely on a user simulator instead of humans. These user simulators have their own problems. While hand-coded, rule-based user simulators have been shown to be sufficient in small, simple domains, for complex domains the number of rules quickly becomes intractable. State-of-the-art data-driven user simulators, on the other hand, are still domain-dependent. This means that adaptation to each new domain requires redesigning and retraining. In this work, we propose a domain-independent transformer-based user simulator (TUS). The structure of our TUS is not tied to a specific domain, enabling domain generalisation and learning of cross-domain user behaviour from data. We compare TUS with the state of the art using automatic as well as human evaluations. TUS can compete with rule-based user simulators on pre-defined domains and is able to generalise to unseen domains in a zero-shot fashion.

1 Introduction

Task-oriented dialogue systems are designed to help users accomplish specific goals within a particular task such as hotel booking or finding a flight. Solving this problem typically requires tracking and planning (Young, 2002). In tracking, the system keeps track of information about the user goal from the beginning of the dialogue until the current dialogue turn. In planning, the dialogue policy makes decisions at each turn to maximise future rewards at the end of the dialogue (Levin and Pieraccini, 1997). The system typically needs thousands of interactions to train a usable policy (Schatzmann et al., 2007; Pietquin et al., 2011; Li et al., 2016; Shi et al., 2019). The amount of interactions required makes learning from real users time-consuming and costly. It is therefore appealing to automatically generate a large number of dialogues with a user simulator (US)[1](Eckert et al., 1997).

Rule-based USs are interpretable and have shown success when applied in small, simple domains. However, expert knowledge is required to design their rules and the number of rules needed for complex domains quickly becomes intractable (Schatzmann et al., 2007). In addition, handcrafted rules are unable to capture human behaviour to its fullest extent, leading to suboptimal performance when interacting with real users (Schatzmann et al., 2006).

Data-driven USs on the other hand can learn user behaviour directly from a corpus. However, they are still domain-dependent. This means that in order to accommodate an unseen domain one needs to collect and annotate a new dataset, and retrain or even re-engineer the simulator.

We propose a transformer-based domain-independent user simulator (TUS). Unlike existing data-driven simulators, we design the feature representation to be domain-independent, allowing the simulator to easily generalise to new domains without modifying or retraining the model. We utilise a transformer architecture (Vaswani et al., 2017) so that the input sequence can have a variable length and dynamic order. The dynamic order takes into account the user's priorities and the varying input length enables the US to incorporate system actions in a seamless manner. TUS predicts the value of each slot and the domains of the current turn, allowing the model to optimise its performance in multiple granularities. By disentangling the user behaviour from the domains, TUS can learn a more general user policy to train the dialogue policy.

[1]There are approaches that attempt to learn a dialogue policy from direct interaction with humans (Gašić et al., 2011). Even then, USs are essential for development and evaluation.

Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 445–456
July 29–31, 2021.

We compare policies trained with our TUS to policies trained with other USs through indirect and direct evaluation as well as human evaluation. The results show that policies trained with TUS outperform those that are trained with another data-driven US and are on par with policies trained with the agenda-based US (ABUS). Moreover, the policy generalises better when evaluated with a different US. Automatic and human evaluations on our zero-shot study show that leave-one-domain-out TUS is able to generalise to unseen domains while maintaining a comparable performance to ABUS and TUS trained on the full training data.

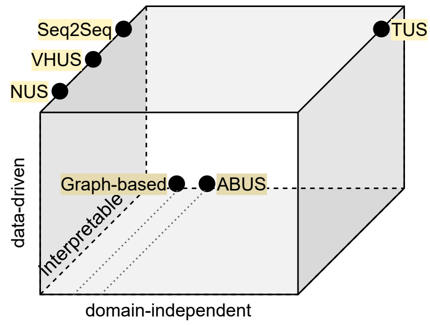


Figure 1: The difference between USs. We compare to which extent a model is data-driven, domain-independent and interpretable.

2 Related Work

The quality of a US has a significant impact on the performance of a reinforcement-learning based task-oriented dialogue system (Schatzmann et al., 2005). One of the early models include an N-gram user simulator proposed by Eckert et al. (1997). It uses a 2-gram model $P(a_u|a_m)$ to predict the user action a_u according to the system action a_m. Since it only has access to the latest system action, its behaviour can be illogical if the goal changes. Therefore, models which can take into account a given user goal were introduced (Georgila et al., 2006; Eshky et al., 2012). The Bayesian model of Daubigney et al. (2012) predicts the user action based on the user goal, and hidden Markov models are used to model the user and the system behaviour (Cuayáhuitl et al., 2005). The graph-based US of Scheffler and Young (2002) combines all possible dialogue paths in a graph. It can generate reasonable and consistent behaviour, but is impractical to implement, since extensive domain knowledge is required.

The agenda-based user simulator (ABUS) (Schatzmann et al., 2007) models the user state as a stack-like agenda, ordered according to the priority of the user actions. The probabilities of updating the agenda and choosing user actions are set manually or learned from data (Keizer et al., 2010). Still, the stacking and popping rules are domain-dependent and need to be designed carefully.

To build a data-driven model, the sequence-to-sequence (Seq2Seq) model structure is widely used. El Asri et al. (2016) propose a Seq2Seq semantic level US with an encoder-decoder structure. Each turn is fed into the encoder recurrent neural network (RNN) and embedded as a context vector. Then this context vector is passed to the decoder RNN to generate user actions. To add new domains, it is necessary to modify the domain-dependent feature representation and retrain the model.

Instead of generating semantic level output, the neural user simulator (NUS) by Kreyssig et al. (2018) generates responses in natural language, thus requiring less labeling, at the expense of interpretability. However, its feature representation is still domain-dependent.

A variational hierarchical Seq2Seq user simulator (VHUS) is proposed by Gür et al. (2018). Instead of designing dialogue history features, the model encodes the user goal and system actions with a vector using an RNN, which alleviates the need of heavy feature engineering. However, the inputs are represented as one-hot encodings, which are also dependent on the ontology. In addition, the output generator is not constrained by the ontology in any way, so it can generate impossible actions.

As shown in Fig. 1, ABUS and graph-based models are domain-dependent and require significant design efforts. Data-driven models such as Seq2Seq, NUS, and VHUS can learn from data, but are constrained by the underlying domain. NUS generates natural language responses, which requires less labeling, but comes with reduced interpretability.

Shi et al. (2019) compared different ways to build a US and indicated that the data-driven models suffer from bias in the corpus. If some actions are rare in the corpus, the model cannot capture them. Thus, the dialogue policy cannot explore all possible paths during training with the data-driven USs. It is important to learn more general human behaviour to reduce the impact of the corpus bias.

3 Problem Description

Task-oriented dialogue systems are defined by a given *ontology*, which specifies the concepts that the system can handle. The ontology can include multiple *domains*. In each domain, there are *informable slots*, which are the attributes that users can assign *values* to, and *requestable slots*, which are the attributes that users can query. For example, in Fig. 2 the user goal has two domains, "hotel" and "restaurant". The slot `Area` is an informable slot with the value `North` in domain "hotel" and `Addr` is a requestable slot in domain "restaurant". The *system state* records the slots and values mentioned in the dialogue history. A US for task-oriented dialogue systems needs to provide coherent responses according to a given user goal $G = \{domain_1 : [(slot_1, value_1), (slot_2, value_2), \dots], \dots\}$. The domains, slots and values are selected from the ontology.

The *user action* is composed of user intents, domains, slots, and values. We consider user intents that appear in the MultiWOZ dataset (Budzianowski et al., 2018). It is of course possible to consider arbitrary intents within the same model architecture, as long as they are defined a priori[2]. The two possible user intents we consider are *Inform* and *Request*. With *Inform*, the user can provide information, correct the system or confirm the system's recommendations. When a user goal cannot be fulfilled, the user can also randomly select a value from the ontology and change the goal. With *Request*, the user can request information about certain slots.

The *system action* is similar to the user action, but there exist more (system) intents. For example, the system can provide suggestions to users with the intent *Recommendation* and make reservations for users with the intent *Book*. More system intents can be found in Appendix A.

We view user simulation in a task-oriented dialogue as a sequence-to-sequence problem. For each turn t, we extract the input feature vectors V^t of the input list of slots $S^t = [s_1, s_2, \dots]$, which is composed of the slots from the user goal and the system action. The output sequence $O^t = [o_1^t, o_2^t, \dots]$ is then generated by the model, where o_i^t shows how the value for slot s_i is obtained. The input feature representation and the output target should be domain-independent in order to generalise to unseen domains without redesigning and retraining. More details can be found in Sec. 4.

By working on the semantic level during training, we retain interpretability. To interact with real users during human evaluation, we rely on template-based natural language generation to convert the semantic-level actions into utterances, as language generation is out of the scope of this work.

[2]We note that intents are not normally dependent on the domain but rather on the kind of dialogue that is being modeled, e.g. task-oriented or chit-chat.

User Goal
`Info:` Hotel-Area=North, Rest-Area=North
`Reqt:` Hotel-Name, Rest-Addr
Conversation
Turn 0
`USR:` I want to find a hotel in the north and a nearby restaurant.
`Inform(Hotel-Area=North, Rest-Area=North)`
`SYS:` There are some good hotels in the south. Which price range do you prefer? Would you mind providing more information?
`Recom(Hotel-Area=South), Request(Hotel-Price), general-reqmore()`
Turn 1
`USR:` No, I want one in the north and I don't care about the price range.
`Inform(Hotel-Area=North, Hotel-Price=dontcare)`

Figure 2: An example dialogue with a multi-domain goal.

4 Transformer-based Domain-independent User Simulator

The TUS model structure is shown in Fig. 3. For each turn t, the list of input feature vectors $V^t = [v_1^t, v_2^t \dots, v_{n_t}^t]$ is generated based on the system actions and the user goal, where v_i^t is the feature vector of slot s_i and n_t is the length of the input list in turn t, V^t. We explain the feature representation in detail in Sec. 4.1. Inspired by ABUS, which models the user state as a stack-like agenda, the length of input list n_t at each turn t varies by taking into account slots mentioned in the system's action. For example, in Fig. 3 the input list V^0 only contains the slots in the user goal at the first turn. Then the system mentions a slot not in the user goal, `Hotel-Price`. So in turn 1 the length of input list V^1 is $n_1 = n_0 + 1$ because one slot is inserted into the input list V^1. The whole input sequence to the model is $V_{input} = [v_{CLS}, v_1^t, \dots, v_{SEP}, v_1^{t-1}, \dots, v_{SEP}]$, where v_{CLS} is the representation of `[CLS]` and v_{SEP} is the representation of `[SEP]`.

The user policy network is a transformer (Vaswani et al., 2017; Devlin et al., 2019). We choose this structure because transformers are able to handle input sequences of arbitrary lengths and to capture the relationship between slots thanks

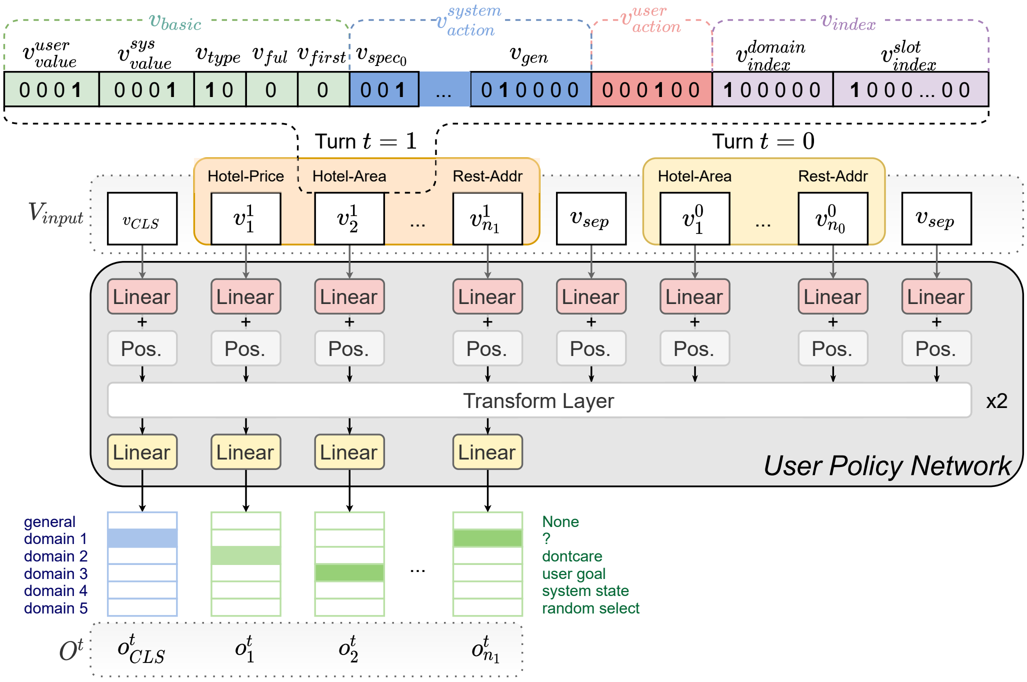


Figure 3: The TUS model structure. The input list starts with a special token, `[CLS]`, and comprises slot lists from previous turns. The slot lists from each turn are separated by a token, `[SEP]`. The model predicts an output vector for each slot in the last turn. Note that the order of slots in each turn is independent from each other. The output for `[CLS]` represents which domains should be selected in the current turn. The user goal and dialogue history are shown in Fig. 2 and here we give the example of the input feature v_i for slot `Hotel-Area`.

to self-attention. The model structure includes a linear layer and position encoding for inputs, two transformer layers, and one linear layer for outputs.

The output list $O^t = [o_1^t, \ldots, o_{n_t}^t]$ consists of one-hot vectors o_i^t which determine the values of the slots s_i at turn t. The dimensions of $o_i^t \in \{0,1\}^6$ correspond to "none", "don't care", "?", "from the user goal", "from the system state", or "randomly selected". More precisely, "none" means that this slot is not mentioned in this turn, "don't care" signifies that the US does not care about this slot, "?" means the US wants to request information about this slot, "from user goal" implies that the value is the same as in the user goal, "from system state" means that the value is as mentioned by the system, and lastly "randomly selected" indicates that the US wants to change its goal by randomly selecting a value from the ontology.

The loss function for slots measures the difference between the predicted output O^t and the target Y^t at each turn t from the dataset as computed by cross entropy (CE), i.e.,

$$loss_{slots} = \frac{1}{n_t} \sum_{i=1}^{n_t} \mathrm{CE}(o_i^t, y_i^t), \qquad (1)$$

where n_t is the number of slots in the input list, o_i^t is the output, and y_i^t is the target of slot s_i in turn t.

4.1 Domain-independent Input Features

We design the input feature representation v_i^t of each slot s_i in turn t consisting of a set of sub-vectors, all of which are domain-independent. For better readability, we drop the slot index i and the turn index t, i.e. we write v for v_i^t.

4.1.1 Basic Information Features

Inspired by the feature representation proposed in El Asri et al. (2016), we use a feature vector v_{basic} that is composed of binary sub-vectors to represent the basic information for each slot. Each slot has two value vectors: v_{value}^{sys} represents the value in the system state, and v_{value}^{user} represents the value in the user goal. Each value vector is a 4-dimensional one-hot vector, with coordinates encoding "none", "?", "don't care" or "other values", in this order. For example, in turn 1 in Fig. 2, for slot `Hotel-Price` $v_{value}^{user} = [1, 0, 0, 0]$, i.e., "none", because it is not in the user goal, and $v_{value}^{sys} = [0, 1, 0, 0]$, i.e., "?", because the system requests it.

The slot type vector v_{type} is a 2-dimensional vector which represents whether a slot is in the user goal as a constraint or a request. For example, in Fig. 2 for Hotel-Area $v_{type} = [1, 0]$ (constraint), while for Hotel-Name $v_{type} = [0, 1]$ (request). A value of $[0, 0]$ means that the slot is not included in the user goal.

The state vector v_{ful} encodes whether or not a constraint or informable slot has been fulfilled. The value is set to 1 if the constraint has been fulfilled, and to 0 otherwise. The vector v_{first} similarly encodes whether a slot is mentioned for the first time.

The basic information feature vector v_{basic} is the concatenation of these vectors, i.e.,

$$v_{basic} = v_{value}^{user} \oplus v_{value}^{sys} \oplus v_{type} \oplus v_{ful} \oplus v_{first} \quad (2)$$

4.1.2 System Action Features

The system action feature vector v_{action}^{system} encodes system actions in each turn. There are two kinds of system actions, general actions and domain-specific actions. The general actions are composed only with general intents, such as "reqmore" and "bye". For example, general-reqmore(). The feature vector of general actions v_{gen} is a multi-hot encoding of whether or not a general intent appears in the dialogue. With a total number of n_{gen} general intents, for each $k \in \{1, \dots, n_{gen}\}$, the k-th entry of v_{gen} is set to 1 if the k-th general intent is part of the system act.

On the other hand, domain-specific actions are composed with domains, slots, values, and domain-specific intents such as "recommend" and "select". For example, Recom(Hotel-Area=South). Each domain-specific action vector v_{spec_j} with the domain-specific j-th intent, $j \in \{1, \dots, n_{spec}\}$, where n_{spec} is the total number of domain-specific intents, is represented by a 3-dimensional one-hot encoding that describes whether the value is "none", "?" or "other values".

The final action representation v_{action}^{system} is formed by concatenating n_{spec} domain-specific action representations together with the general action representation, i.e.,

$$v_{action}^{system} = v_{spec_0} \oplus \dots \oplus v_{spec_{n_{spec}}} \oplus v_{gen}. \quad (3)$$

For the slot Hotel-Area in Fig. 3, we have a vector for each intent. For the intent "recommend" $v_{spec_0} = [0, 0, 1]$, which means that "other values" (in this case South) are mentioned. For all other domain-specific intents, the vectors are $[1, 0, 0]$ since no value is mentioned. In terms of the general intents, only "reqmore" is mentioned, so $v_{gen}[1] = 1$, as "reqmore" is the first general intent.

4.1.3 User Action Features

The output vector from the previous turn O^{t-1} is also included in the input features of the next turn t to take into account what has been mentioned by the US itself, i.e. for slot s_i in turn t, the user action feature $v_{action}^{user} = o_i^{t-1}$.

4.1.4 Domain and Slot Index Features

In some cases, multiple slots may share the same basic feature v_{basic}, system action feature v_{action}^{system} and user action feature v_{action}^{user}. This similarity in features of different slots makes it difficult for the model to distinguish one slot from another, despite the positional encoding. In particular, it is challenging for the model to learn the relationship between turns for a given slot because the number and the order of slots vary from one turn to the next. This may lead to over-generation: the model selects all slots with the same feature vector.

To counteract this issue, we introduce the index feature v_{index}, which consists of the domain index feature $v_{index}^{domain} \in \{0, 1\}^{l_d}$ and the slot index feature $v_{index}^{slot} \in \{0, 1\}^{l_s}$, where l_d is the maximum number of domains in a user goal and l_s is the maximum number of slots in any given domain[3].

To make the index feature ontology-independent, for a particular slot, v_{index} remains consistent throughout a dialogue, but varies between dialogues. The order of the index in each dialogue is determined by the order in the user goal. For example, the "hotel" domain can be the first domain in one user goal of the first dialogue, and the second domain in the next.

Then for each slot in each turn the input feature vector v is formed by concatenating all sub-vectors:

$$v = v_{basic} \oplus v_{action}^{system} \oplus v_{action}^{user} \oplus v_{index}. \quad (4)$$

An example of v for slot Hotel-Area is shown in Fig. 3 based on the dialogue history in Fig. 2. Examples of how the feature representation is constructed can be seen in Appendix D.

4.2 Domain Prediction

Inspired by solving downstream tasks using BERT (Devlin et al., 2019), we utilise the output of [CLS], o_{CLS}, to predict which domains are considered in turn t as a multi-label classification

[3] This does not need to be dependent on the number of domains or slots, it can simply be a random identifier assigned to each slot during one dialogue.

problem. The domain loss $loss_{domain}$ measures the difference between the output o_{CLS} and the target y_{CLS} for each turn by binary cross entropy (BCE). The final loss function is defined as

$$loss = loss_{slots} + loss_{domain}. \qquad (5)$$

5 Experimental Setup

5.1 Supervised Training for TUS

Our model is implemented in PyTorch (Paszke et al., 2019) and optimised using the Adam optimiser (Kingma and Ba, 2015) with learning rate 5×10^{-4}. The dimension of the input linear layer is 100, the number of the transformer layers is 2, and the dimension of the output linear layer is 6. The maximum number of domains l_d is 6 and the maximum number of slots in one domain l_s is 10. During training, the dropout rate is 0.1.

We train our model[4] on the MultiWOZ 2.1 dataset (Eric et al., 2020), consisting of dialogues between two humans, one posing as a user and the other as an operator. The dialogues in the dataset are complex because there may be more than one domain involved in one dialogue, even in the same turn. During training and testing with the dataset, the order of slots in the input list is derived from the data, which means slot s_i is before slot s_{i+1} if the user mentioned slot s_i first. For inference without the dataset, such as when using TUS to train a dialogue policy, the order of slots is randomly generated.

We measure how well a US can fit the dataset by precision, recall, F1 score, and turn accuracy. The turn accuracy measures how many model predictions per turn are identical to the corpus, based on the oracle dialogue history.

5.2 Training Policies with USs

User simulators are designed to train dialogue systems, thus a better user simulator should result in a better dialogue system. We train different dialogue policies by proximal policy optimization (PPO) (Schulman et al., 2017), a simple and stable reinforcement learning algorithm, with ABUS, VHUS, and TUS as USs in the ConvLab-2 framework (Zhu et al., 2020). The policies are trained for 200 epochs, each of which consists of 1000 dialogues. The reward function gives a reward of 80 for a successful dialogue and of -1 for each dialogue turn, with the maximum number of dialogue turns set to 40. For failed dialogues, an additional penalty is set to -40. Each dialogue policy is trained on 5 random seeds. The dialogue policies are then evaluated using all USs by cross-model evaluation (Schatztnann et al., 2005) to demonstrate the generalisation ability of the policy trained with a particular US when evaluated with a different US.

5.3 Leave-one-domain-out Training

To evaluate the ability of TUS in handling unseen domains, we remove one domain during supervised learning of TUS. The leave-one-domain-out TUSs are used to train dialogue policies with all possible domains. For example, TUS-noHotel is trained on the dataset without the "hotel" domain. During policy training, the user goal is generated randomly from all possible domains.

Some domains in MultiWOZ may share the same slots, such as "restaurant" and "hotel" domains which contain property-related slots, e.g. "area," "name," and "price range." However, the corpus also includes domains that are quite different from the rest, For example, the "train" domain which contains many time-related slots such as "arrival time" or "departure time", as well as unique slots such as "price" and "duration." The different properties of the domains will allow us to study the zero-shot transfer capability of the model.

5.4 Human Evaluation

Following the setting in Kreyssig et al. (2018), we select 2 of the 5 trained versions of each dialogue policy for evaluation in a human trial: the version performing best on ABUS, and the version performing best in interaction with TUS. The results of the two versions are averaged. For each version we collect 200 dialogues, which means there are 400 dialogues for each policy in total. Dialogue policies trained with VHUS significantly underperform, so we only consider policies trained with ABUS or TUS for the human trial (see Table 1). The best and the worst policies in the leave-one-domain-out experiment are also included to see the upper and lower bound of the zero-shot domain generalisation performance.

Human evaluation is performed via DialCrowd (Lee et al., 2018) connected to Amazon Mechanical Turk[5]. Users are provided with a randomly generated user goal and are required to interact with our systems in natural language.

[4] https://gitlab.cs.uni-duesseldorf.de/general/dsml/tus_public

[5] https://www.mturk.com/

US for training	US for evaluation ABUS	VHUS	TUS	avg.
ABUS	0.93	0.09	0.58	0.53
VHUS	0.62	0.11	0.37	0.36
TUS	0.79	0.10	0.69	0.53

Table 1: The success rates of policies trained on ABUS, VHUS, and TUS when tested on various USs.

6 Experimental Results

6.1 Cross-model Evaluation

The results of our experiments are shown in Table 1. The policy trained with TUS performs well when evaluated with ABUS, with 10% absolute improvement in the success rate over its performance on TUS. On the other hand, while a policy trained with ABUS performs almost perfectly when evaluated with ABUS, the performance drops significantly, by 35% absolute, when this policy interacts with TUS. This signals that, in the case of ABUS, the policy overfits to the US used for training, and is not able to generalise well to the behaviour of other USs. We found that VHUS is neither able to train nor to evaluate a multi-domain policy adequately. This was also observed in the experiments by Takanobu et al. (2019). We suspect that this is due to the fact that VHUS was designed to operate on a single domain and does not generalise well to the multi-domain scenario. To the best of our knowledge, no other data-driven US has been developed for the multi-domain scenario.

The success rates of policies trained with ABUS and TUS during training, evaluated with both US, are shown in Fig. 4. Each of the systems is trained 5 times on different random seeds. We report the average success rate as well as the standard deviation. Although the policy trained with TUS is more unstable when evaluated on ABUS, it still shows an improvement from the initial policy, converging at around 79%. On the other hand, the policy trained with ABUS and evaluated with TUS barely show any improvements.

6.2 Impact of features and loss functions

We conduct an ablation study to investigate the usefulness of the proposed features and loss functions. The result is shown in Table. 2. First, we measure the performance of the basic model which uses only the basic information feature v_{basic}, the system action feature v^{system}_{action}, and the user action feature v^{user}_{action} as the input. While this model can have a high recall rate, the precision and the turn accuracy are fairly low. We deduce that without the index features the model cannot distinguish the difference between slots and therefore tends to select slots of the same slot type in one turn. For example, it provides all constraints in the first turn, which leads to high recall and over-generation.

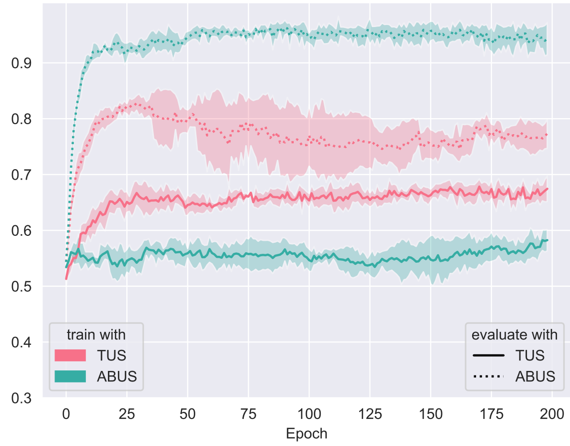


Figure 4: The success rates of policies during training with TUS and ABUS.

method	P	R	F1	ACC	LEN
basic model	0.11	0.71	0.19	0.11	4.51
+ index feature	0.17	0.51	0.26	0.44	1.29
+ domain loss	0.17	0.54	0.26	0.46	1.22

Table 2: The TUS ablation experiments. We analyse the impact of different settings by measuring precision P, recall R, F1 score, turn accuracy ACC, and the average slots mentioned in the first turn user action LEN. Humans, on average, mention 1.5 slots in the first turn.

Analysis of the generated user actions shows that the basic model tends to mention four or more slots in the first turn. This is unnatural, since human users tend to only mention one or two slots at the beginning of a dialogue. More details about the average slots per turn can be found in Appendix B.

After adding the index feature v_{index}, the recall rate is decreased by 17% absolute, but the turn accuracy is increased by 35% absolute, along with improvements on the precision and the F1 score. Furthermore, the average number of slots per turn is closer to that of a real user. Although the recall rate with respect to the target in the data is decreased, this is not necessarily a concern since in dialogue there are many different plausible actions for a given context. For example, when searching for a restaurant, we may provide the information of the area first, or the food type. The order of

US for training	removed data(%)	ABUS Attr.	Hotel	Rest.	Taxi	Train	all	TUS Attr.	Hotel	Rest.	Taxi	Train	all	mean
TUS-noAttr	32.20	**0.69**	0.64	**0.81**	0.65	**0.75**	**0.77**	0.71	0.58	0.66	0.61	**0.69**	0.69	**0.73**
TUS-noTaxi	19.60	0.63	0.61	0.81	0.61	0.70	0.74	0.69	0.60	**0.69**	0.64	0.68	**0.69**	0.72
TUS-noRest	45.21	0.62	**0.66**	0.80	0.56	0.75	0.76	**0.71**	**0.60**	0.64	**0.65**	0.64	0.68	0.72
TUS-noTrain	36.95	0.64	0.65	0.78	**0.67**	0.62	0.73	0.67	0.54	**0.63**	0.64	0.58	**0.64**	0.68
TUS-noHotel	40.15	0.59	0.59	0.76	**0.61**	0.54	0.69	0.64	0.52	0.61	0.61	0.55	0.62	0.66
TUS	0	0.69	0.68	0.81	0.66	0.77	0.79	0.73	0.59	0.66	0.68	0.64	0.69	0.74

Table 3: The success rates of dialogue policies trained with leave-one-domain-out TUSs. For example, the TUS-noAttr model is trained without the "attraction" domain. The sum of all removed data is more than 100% because some dialogues have multiple domains. We report results on all domains.

communicating these constraints may vary.

When we include the domain loss $loss_{domain}$ during training, both the recall rate and the turn accuracy improve while a similar average slot length per turn is maintained. These results indicate that the proposed ontology-independent index features can help the model to distinguish one slot from the other, which solves the over-generation problem of the basic model. The domain loss allows for more accurate prediction of the domain at turn level and the value for each slot at the same time.

6.3 Zero-shot Transfer

We test the capability of the model to handle unseen domains in a zero-shot experiment. In a leave-one-domain-out fashion we remove dialogues involving one particular domain when training the US. The share of each domain in the total dialogue data ranges from 19.60% to 45.21%. During dialogue policy training we sample the user goal from all domains. As presented in Table 3, removing one domain from the training data when training the US does not dramatically influence the policy on the corresponding domain. The final performance of the policies trained with leave-one-domain-out TUSs is still reasonably comparable to the policy trained with the full TUS. This is especially noteworthy considering the substantial amount of data removed during US training and the difference between each domain.

We observe that the model is able to learn about the removed domain from the other domains, although the removed domain is different from the remaining ones. For example, the "train" domain is very different from "attraction", "restaurant", and "hotel", and it is more complex than "taxi", but TUS-noTrain still performs reasonably well on the "train" domain. This signals that the model can do zero-shot transfer by leveraging other domain information. The worst performance on the "train" domain happens instead when the "hotel" domain is removed, i.e. the domain with the most substantial amount of data.

US for training	success Attr.	Hotel	all	overall
ABUS	0.76	0.70	0.83	3.90
TUS	0.73	0.69	0.83	4.03
TUS-noAttr	0.75	0.54	0.81	4.01
TUS-noHotel	0.73	0.55	0.76	3.86

Table 4: The human evaluation results include success rate and overall rating as judged by users.

Our results also show that that some domains are more sensitive to data removal than others, irrespective of which domain is removed. This indicates that some domains are more involved and simply require more training data. This result demonstrates that TUS has the capability to handle new unseen domains without modifying the feature representation or retraining the model. It also shows that our model is sample-efficient.

6.4 Human Evaluation

The result of the human evaluation is shown in Table 4. In total, 156 users participated in the human evaluation. The number of interactions per user ranges from 10 to 80. The success rate measures whether the given goal is fulfilled by the system and the overall rating grades the system's performance from 1 star (poor) to 5 stars (excellent). TUS is able to achieve a comparable success rate as ABUS, without domain-specific information, and even scores slightly better in terms of overall rating. We were not able to observe any statistically significant differences between ABUS and TUS in the human evaluation. For leave-one-domain-out models, the performance of TUS-noAttr is similar to that one of ABUS and TUS without a statistically significant difference. We do however observe a statistically significant decrease in the success rate of TUS-noHotel when compared to TUS and ABUS ($p < 0.05$). This is unsurprising as the hotel domain accounts for 40.15% of the training data. For both TUS-noAttr and TUS-noHotel, the success rate on the domain "attraction" is comparable to TUS and ABUS, but the success rate on the domain "hotel" is relatively low. As observed in the simulation, removing a domain does not decrease the success rate in the corresponding domain as the feature representation is domain agnostic. Instead, it impacts domains which need plenty of data to learn.

7 Conclusion

We propose a domain-independent user simulator with transformers, TUS. We design ontology-independent input and output feature representations. TUS outperforms the data-driven VHUS and it has a comparable performance to the rule-based ABUS in cross-model evaluation. Human evaluation confirms that TUS can compete with ABUS even though ABUS is based on carefully designed domain-dependent rules. Our ablation study shows that the proposed features and loss functions are essential to model natural user behavior from data. Lastly, our zero-shot study shows that TUS can handle new domains without feature modification or model retraining, even with substantially fewer training samples.

In future work, we would like to learn the order of slots and add output language generation to make the behaviour of TUS more human-like. Applying reinforcement learning to this model would also be of interest.

Acknowledgments

We would like to thank Ting-Rui Chiang and Dr. Maxine Eskenazi from Carnegie Mellon University for their help with the human trial. This work is a part of DYMO project which has received funding from the European Research Council (ERC) provided under the Horizon 2020 research and innovation programme (Grant agreement No. STG2018 804636). N. Lubis, C. van Niekerk, M. Heck and S. Feng are funded by an Alexander von Humboldt Sofja Kovalevskaja Award endowed by the German Federal Ministry of Education and Research. Computational infrastructure and support were provided by the Centre for Information and Media Technology at Heinrich Heine University Düsseldorf. Computing resources were provided by Google Cloud.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. 2005. Human-computer dialogue simulation using hidden markov models. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 290–295. IEEE.

Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. 2012. A comprehensive reinforcement learning framework for dialogue management optimization. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):891–902.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Wieland Eckert, Esther Levin, and Roberto Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 80–87. IEEE.

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems. *Interspeech 2016*, pages 1151–1155.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428.

Aciel Eshky, Ben Allison, and Mark Steedman. 2012. Generative goal-driven user simulation for dialog management. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 71–81, Jeju Island, Korea. Association for Computational Linguistics.

Milica Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 312–317. IEEE.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Ninth International Conference on Spoken Language Processing*.

Izzeddin Gür, Dilek Hakkani-Tür, Gokhan Tür, and Pararth Shah. 2018. User modeling for task oriented dialogues. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 900–906. IEEE.

Simon Keizer, Milica Gašić, Filip Jurcicek, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Parameter estimation for agenda-based user simulation. In *Proceedings of the SIGDIAL 2010 Conference*, pages 116–123.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Florian Kreyssig, Iñigo Casanueva, Paweł Budzianowski, and Milica Gašić. 2018. Neural user simulation for corpus-based policy optimisation of spoken dialogue systems. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 60–69, Melbourne, Australia. Association for Computational Linguistics.

Kyusong Lee, Tiancheng Zhao, Alan W. Black, and Maxine Eskenazi. 2018. DialCrowd: A toolkit for easy dialog system assessment. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 245–248, Melbourne, Australia. Association for Computational Linguistics.

Esther Levin and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *Fifth European Conference on Speech Communication and Technology*.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. 2011. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):1–21.

Jost Schatzmann, Kallirroi Georgila, and Steve Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, pages 45–54.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152, Rochester, New York. Association for Computational Linguistics.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126.

Jost Schatztnann, Matthew N Stuttle, Karl Weilhammer, and Steve Young. 2005. Effects of the user model on simulation-based learning of dialogue strategies. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 220–225. IEEE.

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the second international conference on Human Language Technology Research*, pages 12–19. Citeseer.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Weiyan Shi, Kun Qian, Xuewei Wang, and Zhou Yu. 2019. How to build user simulators to train rl-based dialog systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1990–2000.

Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. 2019. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. In

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 100–110, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Steve Young. 2002. Talking to machines (statistically speaking). In *Seventh International Conference on Spoken Language Processing*.

Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. ConvLab-2: An Open-Source Toolkit for Building, Evaluating, and Diagnosing Dialogue Systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

A All System Intents

All system intents in the MultiWOZ 2.1 dataset are listed in Table 5, including 5 general intents and 9 domain-specific intents.

type	intents
general	welcome, reqmore, bye, thank, greet
domain-specific	recommend, inform, request, select, book, nobook, offerbook, offerbooked, nooffer

Table 5: All system intents in the MultiWOZ 2.1

B Average Action Length in Each Turn

The average number of slots mentioned by TUS in each turn when interacting with the rule-based dialogue system is shown in Fig. 5. When the index feature v_{index} and the domain loss $loss_{domain}$ are added, TUS can deal with the over-generation problem and behave more similarly to what is observed in the corpus.

C Success Rates of Leave-one-domain-out Training

The training success rates of dialogue policies trained with leave-one-domain-out TUSs, which are evaluated on TUS, are shown in Fig. 6. In comparison to the full TUS, the leave-one-domain-out TUSs are more unstable, but they can achieve a comparable success rate at the end.

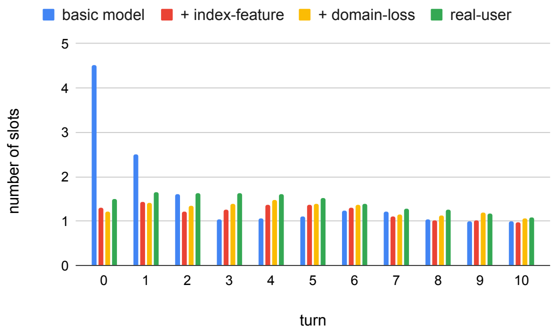


Figure 5: The average user action length per turn when interacting with the rule-based dialogue system. The average action length of real users in the corpus is also presented.

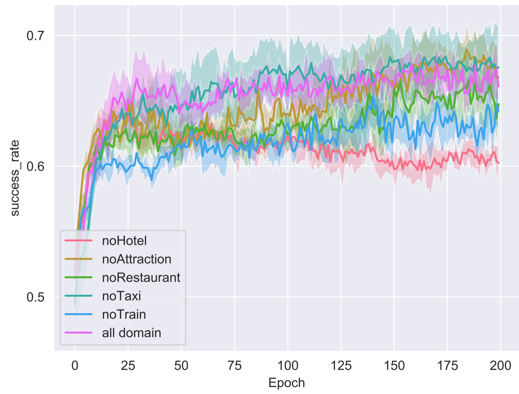


Figure 6: The success rates of dialogue policies trained with leave-one-domain-out TUSs during training, when evaluated on TUS.

D An example for the input feature representation

The list of input feature vectors and output sequence are presented on Fig. 7 based on Fig. 2.

For turn 0, V^0 only includes 4 vectors from the user goal. For turn 1, the system mentions slot `Hotel-Price`, which is not in the user goal, so the feature vector of slot `Hotel-Price` is inserted into V^1, where the 1-st dimension of v_{slot}^{domain} is 1 because domain *Hotel* is the first domain in this conversation and the 3-rd dimension of v_{index}^{slot} is 1 because it is the third slot in domain *Hotel*.

In comparison between the feature vectors of slot `Hotel-Area` in turn 0, v_1^0, and turn 1, v_1^0, the v_{value}^{sys} and v_{spec_0} are different because of the system's domain-specific action `Recom(Hotel-Area=South)`. The system also mentioned a general action, `general-reqmore()`, thus v_{gen} is changed. In

		v_{basic}					v_{action}^{system}			v_{action}^{user}	v_{index}			
		v_{value}^{user}	v_{value}^{sys}	v_{type}	v_{ful}	v_{first}	v_{spec_0}		v_{gen}		v_{index}^{domain}	v_{index}^{slot}		
Turn 0	v_1^0 **(Hotel-Area)**	0 0 0 1	1 0 0 0	1 0	0	0	1 0 0	...	0 0 0 0 0 0	0 0 0 0 0 0	1 0 0 0 0 0	1 0 0 0 ... 0 0	o_1^0	0 0 0 1 0 0
	v_2^0 (Rest-Area)	0 0 0 1	1 0 0 0	1 0	0	0	1 0 0	...	0 0 0 0 0 0	0 0 0 0 0 0	0 1 0 0 0 0	1 0 0 0 ... 0 0	o_2^0	0 0 0 1 0 0
	v_3^0 (Hotel-Name)	0 1 0 0	1 0 0 0	0 1	0	0	1 0 0	...	0 0 0 0 0 0	0 0 0 0 0 0	1 0 0 0 0 0	0 1 0 0 ... 0 0	o_3^0	1 0 0 0 0 0
	v_4^0 (Rest-Addr)	0 1 0 0	1 0 0 0	0 1	0	0	1 0 0	...	0 0 0 0 0 0	0 0 0 0 0 0	0 1 0 0 0 0	0 1 0 0 ... 0 0	o_4^0	1 0 0 0 0 0
Turn 1	v_1^1 **(Hotel-Price)**	1 0 0 0	0 1 0 0	0 0	0	1	1 0 0	...	0 1 0 0 0 0	0 0 0 0 0 0	1 0 0 0 0 0	0 0 1 0 ... 0 0	o_1^1	0 0 1 0 0 0
	v_2^1 **(Hotel-Area)**	0 0 0 1	0 0 0 1	1 0	0	1	0 0 1	...	0 1 0 0 0 0	0 0 0 1 0 0	1 0 0 0 0 0	1 0 0 0 ... 0 0	o_2^1	0 0 0 1 0 0
	v_3^1 (Rest-Area)	0 0 0 1	1 0 0 0	1 0	0	1	1 0 0	...	0 1 0 0 0 0	0 0 0 1 0 0	0 1 0 0 0 0	1 0 0 0 ... 0 0	o_3^1	1 0 0 0 0 0
	v_4^1 (Hotel-Name)	0 1 0 0	1 0 0 0	0 1	0	0	1 0 0	...	0 1 0 0 0 0	1 0 0 0 0 0	1 0 0 0 0 0	0 1 0 0 ... 0 0	o_4^1	1 0 0 0 0 0
	v_5^1 (Rest-Addr)	0 1 0 0	1 0 0 0	0 1	0	0	1 0 0	...	0 1 0 0 0 0	1 0 0 0 0 0	0 1 0 0 0 0	0 1 0 0 ... 0 0	o_5^1	1 0 0 0 0 0

Figure 7: The input and feature representation according to Fig. 2. v_{CLS} and v_{sep} are ignored in this graph.

addition, this slot is first mentioned at turn 0, so v_{first} is changed from 0 to 1. Similarly, v_{action}^{user} is also modified according to the user action. On the other hand, v_{value}^{user} is the same because the user does not update its goal, v_{type} is not changed because the slot is still a constraint, and v_{ful} is 0 because it has not been fulfilled yet. v_{index}^{domain} and v_{index}^{slot} are also the same through the whole conversation.

E Example Dialogue Generated by TUS

An example dialogue with a multi-domain user goal is shown in Fig. 8. It shows that TUS is able to switch between different domains (from turn 2 to 6), respond to the system's requests, and generate multi-domain actions (in turn 5).

```
Turn 0
USR: Inform(Hotel-Area=north, Hotel-Stars=0,
              Hotel-Parking=yes)
SYS: Inform(Hotel-Parking=yes)

Turn 1
USR: Request(Hotel-Type)
SYS: Inform(Hotel-Type=guesthouse)
Turn 2
USR: Inform(Hotel-Stars=0, Hotel-Parking=yes)
SYS: Inform(Hotel-Stars=0, Hotel-Type=guesthouse,
              Hotel-Area=north, Hotel-Parking=yes,
              Hotel-Price=cheap)
Turn 3
USR: Inform(Attr-Type=college)
SYS: Inform(Attr-Choice=18),
     Recom(Attr-Name=hughes hall)
Turn 4
USR: Request(Taxi-Phone, Taxi-Car)
SYS: Request(Taxi-Leave)
Turn 5
USR: Inform(Taxi-Leave=dontcare),
     Request(Taxi-Phone, Taxi-Car, Attr-Fee)
SYS: Inform(Taxi-Car=audi, Taxi-Phone=44162528555,
              Taxi-Car=honda, Taxi-Phone=46793705737,
              Attr-Fee=free)
Turn 6
USR: Request(Attr-Post)
SYS: Inform(Attr-Post=cb23bu)
Turn 7
USR: general-bye()
SYS: general-greet()
```

Figure 8: A dialogue generated by TUS when interacting with the rule-based policy.

Chapter 5

GenTUS: Simulating User Behaviour and Language in Task-oriented Dialogues with Generative Transformers

This chapter summarises our work on simulating user behaviour and language in task-oriented dialogues with generative transformers and gives a verbatim copy of our paper (Lin et al., 2022):

Hsien-chin Lin et al. (Sept. 2022). "GenTUS: Simulating User Behaviour and Language in Task-oriented Dialogues with Generative Transformers". In: *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Edinburgh, UK: Association for Computational Linguistics, pp. 270–282. URL: https://aclanthology.org/2022.sigdial-1.28

5.1 Summary

Although it is efficient to train dialogue systems with semantic-level user simulators, neglecting the diversity in the natural language used in real-world deployment will lead to suboptimal performance.

Therefore, we propose a generative transformer-based user simulator (GenTUS). GenTUS has an encoder-decoder architecture, jointly optimising the user policy and natural language generation. By generating semantic actions and natural language utterances, GenTUS maintains interpretability while enhancing linguistic variation. Furthermore, GenTUS represents inputs and outputs as word sequences, thus it can generalise to unseen domains, slots, and values without modifying the feature representation, and leverage pre-trained language models, e.g. BART (Lewis et al., 2020), to generate user utterances in natural language.

We evaluate the performance of GenTUS using both automatic metrics and human evaluation. Our results demonstrate that GenTUS generates more natural language and can transfer to unseen ontologies in a zero-shot fashion. In addition, the behaviour of GenTUS can be further refined through reinforcement learning, allowing for the training of specialised user simulators.

5.2 Personal contributions

All writing, implementation, and technical results are my contribution. Co-authors assisted in writing and proofreading.

GenTUS: Simulating User Behaviour and Language in Task-oriented Dialogues with Generative Transformers

Hsien-Chin Lin, Christian Geishauser, Shutong Feng, Nurul Lubis, Carel van Niekerk, Michael Heck and **Milica Gašić**
Heinrich Heine University Düsseldorf, Germany
{linh,geishaus,shutong.feng,lubis,niekerk,heckmi,gasic}@hhu.de

Abstract

User simulators (USs) are commonly used to train task-oriented dialogue systems (DSs) via reinforcement learning. The interactions often take place on semantic level for efficiency, but there is still a gap from semantic actions to natural language, which causes a mismatch between training and deployment environment. Incorporating a natural language generation (NLG) module with USs during training can partly deal with this problem. However, since the policy and NLG of USs are optimised separately, these simulated user utterances may not be natural enough in a given context. In this work, we propose a generative transformer-based user simulator (GenTUS). GenTUS consists of an encoder-decoder structure, which means it can optimise both the user policy and natural language generation jointly. GenTUS generates both semantic actions and natural language utterances, preserving interpretability and enhancing language variation. In addition, by representing the inputs and outputs as word sequences and by using a large pre-trained language model we can achieve generalisability in feature representation. We evaluate GenTUS with automatic metrics and human evaluation. Our results show that GenTUS generates more natural language and is able to transfer to an unseen ontology in a zero-shot fashion. In addition, its behaviour can be further shaped with reinforcement learning opening the door to training specialised user simulators.

1 Introduction

Task-oriented dialogue systems (DSs) assist their users in accomplishing a goal, such as booking a flight ticket or making a payment. This should be done through natural language interactions between the system and the user, whilst the system interacts with various external databases and API calls in the background. The core component of such a DS is the dialogue policy module, which decides what should be said to the user next. This module can be trained via interaction with users, through reinforcement learning (RL). However, this creates a conflict between the high cost of interacting with real users and the large amount of interactions required for RL. As a result, user simulators (USs) are often utilised instead to train dialogue policies, as they make it possible for the system to learn from a large number of interactions in a controlled environment at a fraction of the cost.

Rule-based USs are widely used both in research and industry because they are interpretable and can be built without a labelled dataset. However, designing the rules demands expert knowledge and creating these rules becomes intractable on complex domains, making them only suitable for small and simple domains. In addition, human behaviour is too complex and diverse to be manually described by rules, leading to sub-optimal performance of DSs in deployment scenarios (Schatzmann et al., 2006).

On the other hand, data-driven USs can be built with less expert involvement. However, these models are either ontology-dependent (El Asri et al., 2016; Gür et al., 2018; Kreyssig et al., 2018), which means adapting to a new domain requires re-engineering the feature representation or re-training the model, or they do not model the language of the user (Lin et al., 2021). Both shortcomings are serious. The user simulator needs to support zero-shot transfer across ontologies, as it is difficult to collect enough labelled data for each new domain. The ability to produce natural language output is also critical as it makes the training and testing environment more challenging and similar to the real user scenario. Therefore, models that can attain both properties are much needed.

In this work, we propose a model that has both desired properties. More specifically, our contributions are as follows:

- We, propose a **gen**erative **t**ransformer-based

Proceedings of the SIGdial 2022 Conference, pages 270–282
Heriot-Watt University, Edinburgh, UK. 07-09, September, 2022

user **s**imulator that we call *GenTUS*[1]. The response of *GenTUS* includes both semantic actions and natural language utterances, which retains interpretability and induces linguistic variation.

- By optimising the user policy and natural language jointly, GenTUS generates more natural language in the given context.
- GenTUS can adapt to an unseen ontology in a zero-shot fashion and have its behaviour further shaped by reinforcement learning (RL).

The rest of the paper is organised as follows. In Section 2, we review the related work. Section 3 describes in detail the proposed simulation framework. In Section 4, we present the experimental set-up, followed by the experimental results in Section 5. We conclude with Section 6.

2 Related Work

The performance of a task-oriented dialogue policy trained by RL is significantly affected by the quality of the US used to generate the interactions (Schatzmann et al., 2005). An N-gram user simulator proposed by Eckert et al. (1997) is one of the earliest data-driven models. This model predicts the user action a_u according to the system action a_m based on a bi-gram model $P(a_u|a_m)$. Its behaviour is often unreasonable since it only takes the latest system action as input without any information about the user goal. Therefore, models which can act on a given user goal were introduced (Georgila et al., 2006; Eshky et al., 2012). A Bayesian user simulation model which predicts the user action based on the user goal is proposed by Daubigney et al. (2012). In Cuayáhuitl et al. (2005), the user and the system behaviour are modelled by hidden Markov models. A graph-based US, which constructs a graph from all possible dialogue paths, is proposed by Scheffler and Young (2002). This simulator can act reasonably and consistently, but it is not practical to implement in a complex scenario, as it requires extensive domain knowledge.

The agenda-based user simulator (ABUS) (Schatzmann et al., 2007) is widely used to train tourist-information DSs. Its behaviour is based on hand-crafted stacking and popping of rules with a stack-like agenda user goal, ordered by the priority of the user actions. It is difficult to transfer this model to a new ontology because the rules need to be redesigned. Moreover, it only provides semantic-level dialogue acts.

To reduce the involvement of experts, further data-driven user simulator approaches have been proposed. The sequence-to-sequence (Seq2Seq) model structure is the most common framework. A semantic level Seq2Seq user simulator with an encoder-decoder structure is proposed by El Asri et al. (2016). This model embeds the dialogue history into a context vector via a recurrent neural network (RNN) encoder. Its decoder then generates user actions based on the context embedding vector.

Instead of generating dialogue acts, the neural user simulator (NUS) of Kreyssig et al. (2018) can generate responses in natural language. However, this model has limited interpretability because it does not provide semantic-level outputs and its input representation is domain-dependent.

The variational hierarchical Seq2Seq user simulator (VHUS) proposed by Gür et al. (2018) encodes the system actions and the user goal by RNNs instead of complex dialogue history features and generates semantic user actions. Its features are still domain-dependent as system actions and user goals are represented by domain-dependent one-hot encodings. As VHUS has no constraints in the decoding process, it often generates impossible actions under the given ontology.

A domain-independent transformer-based user simulator (TUS) is proposed by Lin et al. (2021). With domain-independent input and output feature representations, TUS can adapt to an unseen domain in a zero-shot fashion. However, it does not model natural language output. Moreover, all intents are part of the model, which makes transfer to an unrelated ontology, i.e. the one with a different sets of intents, difficult.

To convert the dialogue acts from the semantic level to natural language, a user simulator commonly includes an NLG module connected to the semantic level user policy. Although template-based NLGs are widely used in research, creating templates for every dialogue act is labour-intensive and lacks language variation. Data-driven NLG models, such as SC-LSTM (Wen et al., 2015) and SC-GPT (Peng et al., 2020) can generate natural language utterances conditioned on given semantic actions. However, taking only semantic actions as input, their results may not be sufficiently natural in a given context. In addition, the user policy and

[1] `https://gitlab.cs.uni-duesseldorf.de/general/dsml/gentus-public.git`

NLG model cannot be optimised jointly within the modular architecture.

An end-to-end US which generates both dialogue acts and utterances is proposed by Tseng et al. (2021), although in their evaluation they train a DS using only the semantic actions from the US. The NLG of this US is based on a simple delexicalised LSTM model. The user goal is represented as a binary vector, with each dimension representing a domain-slot pair in the ontology. This creates several obstacles for transfer to an unseen ontology: such a transfer would require further hand-coded lexicalisation rules for the NLG component, modifications of the feature representations and further fine-tuning of the US policy.

3 Generative Transformer-based User Simulation

Task-oriented DSs are expected to handle the requests of real users in natural language. Therefore, when designing USs, it is important to endow them with the ability to converse with the system via natural language as well. In this way, we can study, for example, the robustness of the systems towards misunderstandings that may occur when conversing with real users. On the other hand, users rarely misunderstand the DS response. It is hence reasonable to assume that the input to the US may be on the semantic level. This is also practical in such cases as when DSs need to execute API calls, such as playing a song or turning off the light.

Task-oriented DSs are built upon an *ontology* which includes all possible *intents* that the user or the system can exhibit in their actions and *domains*, which describes the entities the user or the system can talk about. Domains are further characterised by a number of *slots* and each slot can take a number of *values*. In task-oriented DS we assume that the user has a particular goal they want to achieve. We define goal as the following set $G = \{domain_1 : [(slot_1, value_1), (slot_2, value_2), \ldots], domain_2 : [(slot_3, value_3), \ldots], \ldots\}$, where domains, slots and values are selected from the ontology.

The semantic *user action* and *system action* are composed of several tuples of the following structure: $(intent, domain, slot, value)$. Users and systems may have different intents, e.g., systems can *recommend* an option and users can *negate* the recommended offer. A semantic action can be converted into a natural language utterance, which we denote with $text_{usr}$ in the case of a user action.

User simulation in a task-oriented dialogue can be modelled as a sequence-to-sequence problem. For each turn, GenTUS takes the context information as an input sequence, including the system action, the user history, the user goal, and turn information, and generates the semantic action and the natural language response as the output sequence. In following sections, we provide more details.

3.1 Model Structure

The backbone of the proposed GenTUS user simulation model is an encoder-decoder structure as shown in Fig. 1. In turn t, the user goal is updated by the user action from the previous turn and the current system action. If the system informs that the user's request is not possible or fails, the value of constraint slots will be replaced by a random value. The encoder takes the system action $action^t_{sys}$, user actions from previous 3 turns $action^{t-1:t-3}_{usr}$, the user goal $goal$, and the turn number t as input. Then the decoder generates both the user semantic action $action^t_{usr}$ conditioned on the output of the encoder and the associated natural language response $text_{usr}$. We initialise GenTUS by BART (Lewis et al., 2020), which is a transformer-based natural language generator with a bidirectional encoder and a left-to-right decoder. BART achieves convincing results on text generation and comprehension tasks after fine-tuning.

3.2 Input and Output Representation

The *system action* and *user action* are semantic level dialogue acts and are represented by a list of tuples $(intent, domain, slot, value)$. Note that the output of this user simulator is a semantic as well as a natural language representation of the user action. The natural language action is sent to the system, while the semantic action is retained by the user simulator for the next turn. The user goal $goal$ is represented by a list of tuples,

$$[(domain_1, type_1, slot_1, value_1, status_1), \\ (domain_2, type_2, slot_2, value_2, status_2), \ldots] \tag{1}$$

Following the setting in Lin et al. (2021), the tuples are ordered by the user preference, which means one tuple is in front of the others if the user prefer to mention it earlier. The *intent*, *domain*, *slot*, and *value* are sampled from the ontology. The *type* represents whether a slot in the goal is a constraint *info*, a request *reqt*, or a booking informa-

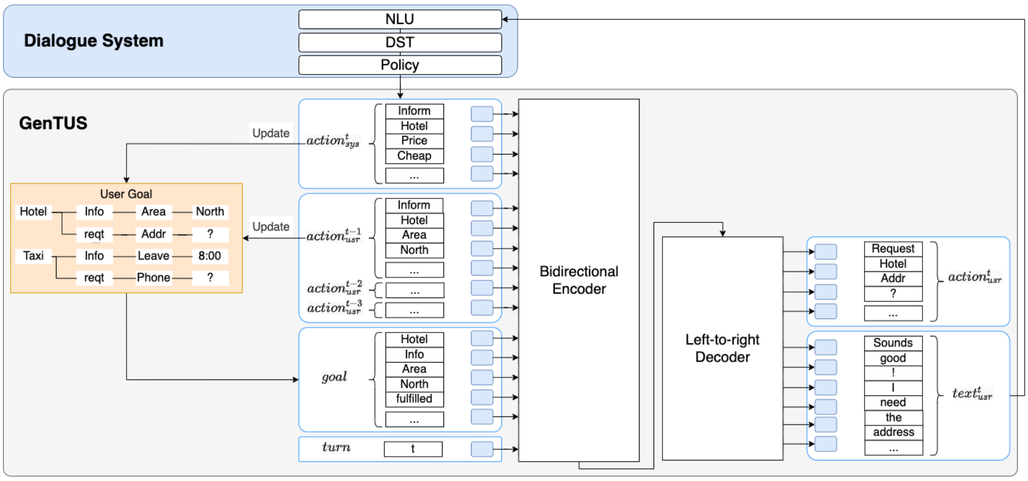


Figure 1: The model structure of GenTUS. Both input and output are JSON-formatted word sequences.

tion *book*. The *status* represents the condition of each domain-slot pair. It can be *fulfilled*, *in conflict*, *requested*, or *not mentioned*. The turn information is the number of the current dialogue turn. We represent the input to GenTUS as a JSON-formatted string: "{"system": $action^t_{sys}$, "user": $action^{t-1:t-3}_{usr}$, "goal": $goal$, "turn": t}".

The output of GenTUS is a set of semantic-level user sub-actions and the corresponding utterance in natural language. The output is also easily represented as a JSON-formatted string: "{"action": $action^t_{usr}$, "text": $text^t_{usr}$}".

As ultimately both input and output contain only words, we can train GenTUS as a sequence-to-sequence model. By using a pre-trained language model for initialisation, we can harness the generalisation capabilities of these powerful models when adapting to a new ontology.

3.3 Constrained Semantic Decoding Space

The downside of using a large pre-trained language model as a generator is that it may suffer from generating hallucinations. This means that we should place constraints on the output to prevent generating illegal semantic actions, which is particularly problematic for DSs.

In order to only produce valid actions, every semantic action $(intent, domain, slot, value)$ is created by following a path in a graph that defines the valid actions, where the graph is constructed as follows. The possible *intents* in the diagram are derived from the ontology. For example, the MultiWOZ dataset (Budzianowski et al., 2018) contains general intents like *greeting* and *bye*, and domain-specific intents like *inform* and *request*. The possible *domains*, *slots* and *values* are derived from the user goal, and system actions are used to update the nodes. The possible paths following intent, domain, slot and value are constrained by the ontology, which defines what valid actions are comprised of. Fig. 2 depicts an example, where GenTUS selected the action `[(Inform, Hotel, Area, North)]` in turn 0 and `[(Request, Hotel, Addr, ?), (Inform, Taxi, Leave, 8:00)]` in turn 1 by following the two paths in the diagram. The graph derived from the user goal is depicted on the left of Fig. 2 and updated after the system asked about a cheap hotel. After every decoded action the model can decide whether to continue or stop the decoding process. It is important to highlight that while we use the ontology to constrain the generation process, no part of the ontology is ever part of the model, but the model uses the ontology as one additional input. In that way it can be transferred to a new ontology in a purely zero-shot manner.

4 Experimental Setup

The objective of our experiments is four-fold. First, we want to show that when trained and tested on the same ontology, the user simulator can adequately capture the semantics represented in the real user data. At the same time, we also want to examine its zero-shot capability by conducting the evaluation on another unseen ontology. Second, as natural language output is an important component of the proposed model, we evaluate it separately using both automatic measures and a human preference test. Third, we jointly evaluate the GenTUS dia-

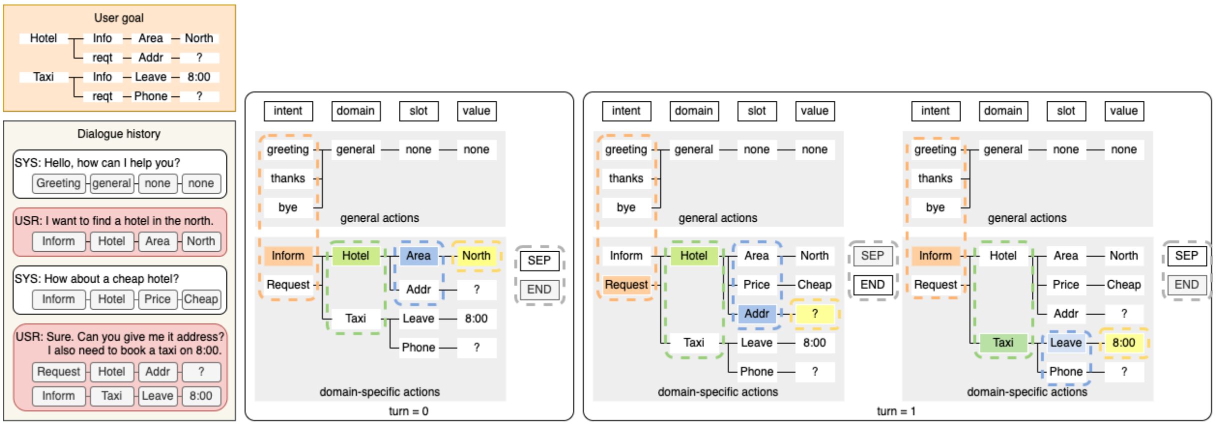


Figure 2: An example of a constrained semantic decoding space. The intents come from the ontology whereas domains, slots and values come from the user goal. In addition, system actions can insert new nodes. The user semantic actions can only contain nodes from the graph. More details are mentioned in section 3.3.

logue policy and its natural language output using a human trial and compare it to the state of the art. This aims to show the value of optimising the user simulator behaviour and language at the same time. Finally, we show how the behaviour of GenTUS can be further shaped by RL in interaction with a DS, with the aim of demonstrating that this model can yield a number of specialised user simulators.

4.1 Datasets

We conduct our experiments on two corpora, the Multi-Domain Wizard-of-Oz (MultiWOZ) (Budzianowski et al., 2018) and Schema-Guided Dialogue (SGD) (Lee et al., 2022) datasets. MultiWOZ is a human-to-human conversation dataset including around 10k dialogues, one person posing as a user and the other as an operator. In this dataset, more than one domain may be involved in one dialogue, even in the same turn. SGD consists of more than 20k dialogues between humans and a virtual assistant. The ontology of MultiWOZ includes 5 intents (3 general intents, e.g., *greeting* and *bye*, and 2 domain-specific intent, i.e., *inform* and *request*) and 7 different domains, e.g. *hotel* and *attraction*. On the other hand, the ontology of SGD includes 11 intents (2 general intents, i.e., *thank-you* and *goodbye*, and 9 domain-specific intents, e.g. *inform*, *request*, and *confirm*) and 20 different domains, e.g., *bank* and *music*. More details of these two datasets are listed in Appendix A.

4.2 Supervised Learning for GenTUS

Our model is inherited from Huggingface's transformers (Wolf et al., 2020) and trained on both MultiWOZ and SGD. To measure how well GenTUS can transfer to a new ontology, the model trained on MultiWOZ is not only tested on the MultiWOZ test set but also evaluated on the SGD test set without any further fine-tuning, and vice versa. To the best of our knowledge, no other data-driven US has been tested in such a rigorous zero-shot transfer set-up.

We evaluate NLG performance by automatic metrics, including slot error rate (SER), sacre-BLEU score (Post, 2018) and self-BLEU score (Zhu et al., 2018), and a human preference test. SER evaluates the exact matching of semantic actions in the candidate utterance. $SER = (m + h)/N$, where N is the total number of slots in semantic actions, m and h stand for the number of missing and hallucinated slots, respectively. The self-BLEU is a diversity evaluation metric. For every data point we generate a sentence. Given such a sentence, we calculate a BLEU score where the reference sentences are all other generated sentences. Then we can get the self-BLEU score by averaging all these results. The lower self-BLEU score implies the higher diversity. We conduct the human preference test on the Amazon Mechanical Turk[2] platform. Following the setting of Peng et al. (2021), the workers are requested to rate each utterance from 1 (bad) to 3 (good) in terms of informativeness and naturalness. *Informativeness* measures whether the given utterance contains all the information specified in the semantic actions. *Naturalness* evaluates whether the given utterance is human-like. A screenshot of this questionnaire can be found in Appendix C.

In addition, we measure how well GenTUS can

[2] https://www.mturk.com/

fit or transfer to a dataset using precision, recall, F1 score, as well as turn accuracy on the semantic level and sacre-BLEU on the language level.

4.3 Training the Dialogue System with User Simulators

USs are designed to simulate the real-world scenario for training DSs, thus USs should respond in natural language as real users' utterances. In this section, we investigate the ability of the proposed model to train a dialogue policy by interacting on the natural language level.

The policies of different DSs are trained by proximal policy optimization (PPO) (Schulman et al., 2017), a simple and stable RL algorithm, with different USs, including the agenda-based US (ABUS) with template-based NLG (ABUS-T), ABUS with SC-GPT (ABUS-S), and GenTUS which generates language. Note that we do not include NLG modules in evaluation which are based on delexicalisation, such as Tseng et al. (2021), as their performance strongly depends on the amount of hand-coding invested in defining the delexicalisation rules. The downsides of delexicalisation already became clear in early neural network dialogue state trackers (Mrkšić et al., 2017) and are further exacerbated in natural language generation (Peng et al., 2020). We do however include a rule-based user simulator (Schatzmann et al., 2007) with a template-based NLG, noted as ABUS-T in our experiments, as the rule-based user simulator has achieved competitive results in human evaluations (Kreyssig et al., 2018; Lin et al., 2021). Also, TUS (Lin et al., 2021) did not significantly outperform ABUS in the human trial, so we exclude it from the evaluation here.

To deal with the user response in natural language, a natural language understanding module composed with BERT (Devlin et al., 2019) (BERTNLU) is included and a rule-based dialogue state tracker (RuleDST) is used to track the users' states for each DS. These modules, e.g., BERTNLU, RuleDST, ABUS, a template-based NLG, and SC-GPT, are provided in the ConvLab-2 framework (Zhu et al., 2020).

We train policies for 200 epochs, each of which consists of 1000 dialogues. The reward function gives a reward of 80 for a successful dialogue and -1 for each dialogue turn, with the maximum number of dialogue turns set to 40. For failed dialogues, an additional penalty is set to -40. Each dialogue policy is trained on 5 random seeds.

We apply the cross-model evaluation (Schatztmann et al., 2005) to evaluate these DSs. Different USs are used to evaluate a DS which is trained with a particular US to estimate the generalisation ability. We also conduct an interactive human trial. For evaluation, we select the DS policy performing best on the US it was trained on. For each DS we collected 300 dialogues. The human trial is implemented with DialCrowd (Lee et al., 2018; Huynh et al., 2022) connected to the Amazon Mechanical Turk platform. Users are provided with randomly generated user goals based on the ontology of MultiWOZ and are required to interact with DSs in natural language.

4.4 Fine-tuning GenTUS with RL

Simulators purely trained using supervised learning will learn behaviour that best fits the data and most likely will result in general behaviour. As behaviour can be very different from one user to another, it is important to be able to model different user behaviours, which will in turn result in more robust policies. To this end, we further fine-tune GenTUS using RL and shape its behaviour by deploying different reward functions. In order to achieve that, for a given user action $\{(intent_i, domain_i, slot_i, value_i)\}_{i=1}^{m}$, we define the turn level reward $r := -\rho_{eff} + \rho_{act} \cdot m$, where ρ_{eff} and ρ_{act} are hyperparameters. In addition, as for the system reward, we give a reward of 80 for a successful dialogue and -40 for a failed dialogue at the very end of the dialogue. We let GenTUS interact with the rule-based dialogue system both on semantic level and optimise its behaviour using PPO. We test two different reward settings that are distinguished by the turn level reward: $r_1 := -5 \cdot m$ (turn level penalty and low action reward) and $r_2 := -10 + 20 \cdot m$ (turn level penalty and high action reward). The corresponding average returns and trained user simulators associated with the rewards are abbreviated with R_1, R_2 and $User_1, User_2$ respectively. We train each model on 4 different seeds. We then take for every seed the model with highest average return on its respective reward and evaluate on the other reward functions to obtain a cross-reward evaluation.

5 Experimental Results

Our experimental results can be divided into five parts. In Section 5.1, we analysis the impact of different features with an ablation study. In Section 5.2, we conduct the *direct* evaluation by measuring automatic metrics (SER, sacre-BLEU, and self-BLEU) and human ratings (informativeness and naturalness) from the preference test. In Section 5.3, we focus on the generalisability of GenTUS by a zero-shot ontology transfer experiment, measured by semantic level and language level metrics on two different corpora. The *indirect* evaluation is in Section 5.4. We compare DSs trained by different USs with cross-model evaluation. The result from the interactive human trial is discussed in Section 5.5. In Section 5.6, we show that it is possible to further configure the behaviour of GenTUS via RL.

5.1 Impact of different features

We conduct an ablation study to investigate the usefulness of our proposed feature representation. The result is shown in Table 1. First, we measure the performance of the model which takes the turn information, the system action $action_{sys}^{t}$ and the user action $action_{usr}^{t-1}$ from previous turn. Without context information, the model can only achieve 0.21 turn accuracy and 0.35 F1-score. After including the user goal $goal$, the F1-score is improved by 0.30 and the turn accuracy is also improved by 0.30 absolutely. After adding more user history $action_{usr}^{t-1:t-3}$, the F1 score is also improved slightly with the same turn accuracy.

This result indicates that the context information can improve the performance especially including the user goal in the input sequence.

Model	**P**	**R**	**F1**	**ACC**
System and user action only	0.42	0.30	0.35	0.21
+ user goal	0.66	0.64	0.65	0.51
+ history	0.68	0.66	0.66	0.51

Table 1: The GenTUS ablation experiments on MultiWOZ. We analyse the impact of different input features by measuring precision (P), recall (R), F1 score (F1), and turn accuracy (ACC).

5.2 Natural Language Evaluation

The NLG performance of different models on MultiWOZ is shown in Table 2. TemplateNLG, SC-GPT, and GenTUS-golden generate natural language responses from golden semantic actions and their SER is calculated based on these golden semantic actions. On the other hand, the language of GenTUS is generated based on semantic actions

Model	**SER ↓**	**sacre-BLEU ↑**	**self-BLEU ↓**
Human	3.92%	-	0.77
TemplateNLG	1.67%	10.46	0.89
SC-GPT	5.33%	10.51	**0.79**
GenTUS-golden	5.73%	**19.61**	0.93
GenTUS	**3.97%**	-	0.95

Table 2: The NLG performance on MultiWOZ. GenTUS-golden is generated based on the golden semantic actions and GenTUS is using its own semantic action prediction. The arrow direction means which trend is better.

Model	**Informativeness**	**Naturalness**
SC-GPT	2.50	2.45
GenTUS	2.55	**2.58**

Table 3: Human preference test for NLG on MultiWOZ. The naturalness score is statistically significantly different ($p_v < 0.05$).

predicted by itself, which means we can directly measure the agreement between the semantic action the simulator indented to produce and the final natural language content produced by the simulated user. The sacre-BLEU is calculated with golden utterances.

Although data-driven NLG models have higher SER than template-based NLG, these models have better scores in BLEU. GenTUS-golden outperforms SC-GPT by 9.10 points in BLEU because our model not only takes semantic actions as input but also context information, e.g., the user goal. Moreover, there is no statistically significant difference in SER between SC-GPT and GenTUS-golden. The human preference test in Table 3 also shows that GenTUS is more natural than SC-GPT with similar informativeness. The diversity of the proposed model is the worst, which is not surprising as we didn't include beam-search or sampling to keep the computational complexity as low as possible. An investigation of a method which balances the two we leave for future work.

Without golden dialogue acts in the input, the SER of GenTUS drops by 1.77% absolute when GenTUS generates utterances from its prediction dialogue acts instead of from golden dialogue acts, which means the language-level and semantic-level outputs of GenTUS are in agreement. In other words, with the context information and its predicted semantic actions, GenTUS can generate more natural language and have fewer missing and redundant pieces of information.

5.3 Zero-shot Ontology Transfer

The results of zero-shot ontology transfer are shown in Table 4. For the semantic level evaluation, GenTUS has higher precision, recall, F1 score and turn accuracy on MultiWOZ than SGD when training and testing on the same corpus. The reason is the ontology of SGD is more complicated than MultiWOZ, i.e., contains more intents, domains, slots and values as shown in Section 4.1.

The performance of GenTUS trained on MultiWOZ dropped by 0.39 on F1 score and 0.35 on turn accuracy when testing on SGD. On the other hand, GenTUS trained on SGD can still achieve 0.49 on F1 score and 0.34 turn accuracy when testing on MultiWOZ without fine-tuning on the unseen MultiWOZ ontology. In other words, GenTUS trained on SGD can get a comparable F1 score and turn accuracy on both known and unknown ontology.

When testing and training on the same corpus, the BLEU score of GenTUS is 17.84 on MultiWOZ and 18.30 on SGD. However, when transferring to another corpus, the BLEU score drops because users in MultiWOZ and SGD have different vocabulary and language styles.

Train data	**Test data**	**Semantic** P	R	F1	ACC	**Language** sacreBLEU
M	M	0.68	0.66	0.66	0.51	17.84
S	S	0.60	0.58	0.58	0.47	18.30
S	M	0.51	0.51	0.49	0.34	2.70
M	S	0.30	0.26	0.27	0.16	1.86

Table 4: The cross-dataset evaluation of GenTUS based on two different corpora, MultiWOZ 2.1 (M) and Schema-Guided Dialogue dataset (S). The semantic actions and language responses generated by GenTUS are evaluated by semantic level metrics, i.e., precision (P), recall (R), F1 score (F1) and turn accuracy (ACC), and language level metric, i.e., sacre-BLEU.

5.4 Cross-model Evaluation

The results of cross-model evaluation are presented in Table 5. The DS trained with GenTUS has the best performance when interacting with ABUS-T in a 15% absolute improvement in success rate over its performance on GenTUS. On the other hand, although the DS trained with ABUS-T achieves 78% success rate, its performance drops by 28% absolute when evaluated by GenTUS. The DS trained with ABUS-S also performs best when interacting with ABUS-T, with 17% absolute improvement in success rate interacting with ABUS-S. All three DSs achieve their best performance when evaluated by ABUS-T, which means ABUS is the easiest setting. This indicates that it may not be sufficient to simulate real world scenario with only a handcrafted policy and a template-based NLG.

On the other hand, the USs with data-driven NLG are more difficult for the DS to handle. The DS trained by ABUS-T performs better than the DS trained by ABUS-S because they learn from the same policy and SC-GPT has higher SER, making the DS hard to be fully optimised.

US for training	**US for testing** ABUS-T	ABUS-S	GenTUS
ABUS-T	**0.78**	**0.63**	0.50
ABUS-S	0.74	0.57	0.45
GenTUS	0.68	0.43	**0.53**

Table 5: The success rates of policies trained on GenTUS, ABUS with template NLG (ABUS-T), and ABUS with SC-GPT (ABUS-S) when tested on various USs. Each pair is evaluated by 400 dialogues on 5 seeds, which is 2K dialogues in total.

5.5 Interactive Human Trial

US for training	**Success**	**Overall**
ABUS-T	0.75	3.71
ABUS-S	0.79	3.83
GenTUS	**0.86**	**4.08**

Table 6: The interactive human trial results include success rate and overall rating as judged by users. Each system is evaluated by 300 dialogues. The success rate and overall score of GenTUS are statistically significantly different from ABUS-S and ABUS-T ($p_v < 0.05$)

The result of the interactive human trial is shown in Table 6. 155 users were involved in this trial. The number of interactions per user varies from 1 to 48. A dialogue is rated as successful if the system fulfils the user's given goal. The overall rating ranges from 1 (very poor) to 5 (excellent).

The DS trained by GenTUS outperforms the DS trained by ABUS-T and the DS trained by ABUS-S both on success rate and overall rating, which shows that is beneficial to train a DS with a jointly optimised user policy and NLG. However, we cannot observe statistically significant differences between ABUS-T and ABUS-S on success and overall rating, which means including a data-driven NLG module with the rule-based US is not sufficient to train an optimal DS.

Models	Success	Avg Acts	Turns	$\mathbf{R_1}$	$\mathbf{R_2}$
User 1	0.84 ± 0.03	1.33 ± 0.03	7.01 ± 0.27	33.5 ± 3.5	34.2 ± 3.7
User 2	0.78 ± 0.04	1.81 ± 0.04	7.24 ± 0.33	4.3 ± 6.2	119.1 ± 15.5
Supervised	0.76 ± 0.08	1.39 ± 0.04	7.38 ± 0.32	30.9 ± 8.2	38.6 ± 10.0

Table 7: Results after fine-tuning GenTUS using RL on three different reward functions. Results show mean and 95% confidence intervals.

5.6 Fine-tuning GenTUS with RL

The results of RL training are depicted in Table 7. We can observe that both users obtain the highest return on the respective reward function. The success rate of both user 1 and user 2 are higher than supervised model because of the success reward signal in RL. User 1, which tries to lower its number of actions, has a similar average number of actions compared to supervised model, suggesting that paid users from the corpus do not want to say more than is necessary to achieve a successful dialogue. User 2, which is rewarded for taking many actions in a turn, shows a much higher average number of actions compared to the other users, reflecting a different user behaviour – a chatty user.

6 Conclusion

We propose a generative transformer-based user simulator (GenTUS), which achieves high interpretability and linguistic variation by generating both semantic actions and natural language utterances. Moreover, it produces generalisable feature representation by treating the inputs and outputs as word sequences and leveraging a large pre-trained language model. Our results show that GenTUS generates more natural language than SC-GPT in a given context and it can transfer to an unseen ontology in a zero-shot fashion. We consolidate our findings by a number of automatic as well as human evaluations. In addition, the GenTUS behaviour can be further configured by RL with different reward functions, providing an opportunity to build specialised USs. In future work, we hope to modify also the NLG of GenTUS via RL in order to model user sentiment or personality.

Acknowledgements

This work is a part of DYMO project which has received funding from the European Research Council (ERC) provided under the Horizon 2020 research and innovation programme (Grant agreement No. STG2018 804636). N. Lubis, C. van Niekerk, M. Heck and S. Feng are funded by an Alexander von Humboldt Sofja Kovalevskaja Award endowed by the German Federal Ministry of Education and Research. Computing resources were provided by Google Cloud and HHU ZIM.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. 2005. Human-computer dialogue simulation using hidden markov models. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 290–295. IEEE.

Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. 2012. A comprehensive reinforcement learning framework for dialogue management optimization. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):891–902.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Wieland Eckert, Esther Levin, and Roberto Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 80–87. IEEE.

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems. *Interspeech 2016*, pages 1151–1155.

Aciel Eshky, Ben Allison, and Mark Steedman. 2012. Generative goal-driven user simulation for dialog management. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language*

Learning, pages 71–81, Jeju Island, Korea. Association for Computational Linguistics.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Ninth International Conference on Spoken Language Processing*.

Izzeddin Gür, Dilek Hakkani-Tür, Gokhan Tür, and Pararth Shah. 2018. User modeling for task oriented dialogues. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 900–906. IEEE.

Jessica Huynh, Ting-Rui Chiang, Jeffrey Bigham, and Maxine Eskenazi. 2022. Dialcrowd 2.0: A quality-focused dialog system crowdsourcing toolkit. In *Proceedings of the Language Resources and Evaluation Conference*, pages 1256–1263, Marseille, France. European Language Resources Association.

Florian Kreyssig, Iñigo Casanueva, Paweł Budzianowski, and Milica Gašić. 2018. Neural user simulation for corpus-based policy optimisation of spoken dialogue systems. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 60–69, Melbourne, Australia. Association for Computational Linguistics.

Harrison Lee, Raghav Gupta, Abhinav Rastogi, Yuan Cao, Bin Zhang, and Yonghui Wu. 2022. Sgd-x: A benchmark for robust generalization in schema-guided dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Kyusong Lee, Tiancheng Zhao, Alan W Black, and Maxine Eskenazi. 2018. Dialcrowd: A toolkit for easy dialog system assessment. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 245–248.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Hsien-chin Lin, Nurul Lubis, Songbo Hu, Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, and Milica Gasic. 2021. Domain-independent user simulation with transformers for task-oriented dialogue systems. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 445–456, Singapore and Online. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, Vancouver, Canada. Association for Computational Linguistics.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2021. Soloist: Building task bots at scale with transfer learning and machine teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 172–182.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Jost Schatzmann, Kallirroi Georgila, and Steve Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, pages 45–54.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152, Rochester, New York. Association for Computational Linguistics.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126.

Jost Schatztnann, Matthew N Stuttle, Karl Weilhammer, and Steve Young. 2005. Effects of the user model on simulation-based learning of dialogue strategies. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 220–225. IEEE.

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the second international conference on Human Language Technology Research*, pages 12–19. Citeseer.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable dialogue systems and user simulators. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 152–166, Online. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. ConvLab-2: An Open-Source Toolkit for Building, Evaluating, and Diagnosing Dialogue Systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100.

A Intents and domains in MultiWOZ and SGD

type	system	user
general	welcome, reqmore, bye, thank, greet	bye, thank, greet
domain-specific	recommend, inform, request, select, book, nobook, offerbook, offerbooked, nooffer	inform, request

Table 8: Al intents in the MultiWOZ dataset.

All intents in the MultiWOZ dataset are listed in Table 8 and all intents in SGD dataset are listed in Table 9. The domains in SGD follow the form of <domain_name>_<number> and the number is used to disambiguate services from the same domain (Lee et al., 2022). We normalize them to domain name only. All domains in MultiWOZ and SGD and listed in Table 10.

type	system	user
general	goodbye, req_more	thank_you, goodbye
domain-specific	inform, notify_success, request, notify_failure, confirm, offer_intent, offer, inform_count	inform_intent, inform negate_intent, negate affirm_intent, affirm, request_alts, request, select

Table 9: All intents in the SGD dataset.

dataset	domains
MultiWOZ	attraction, hospital, hotel, police, restaurant, taxi, train
SGD	alarm, banks, bus, calendar, events, flights, homes, hotels, media, messaging, movies, music, payment, rental_cars, restaurants, ride_sharing, servies, train, travel, weather

Table 10: All domains in MultiWOZ and SGD. The names of domains in SGD are normalized.

B A generated dialogue example

Here we show a dialogue example of GenTUS interacting with a rule-based DS in Table 11. The system's semantic actions are converted to natural language by a template-based NLG.

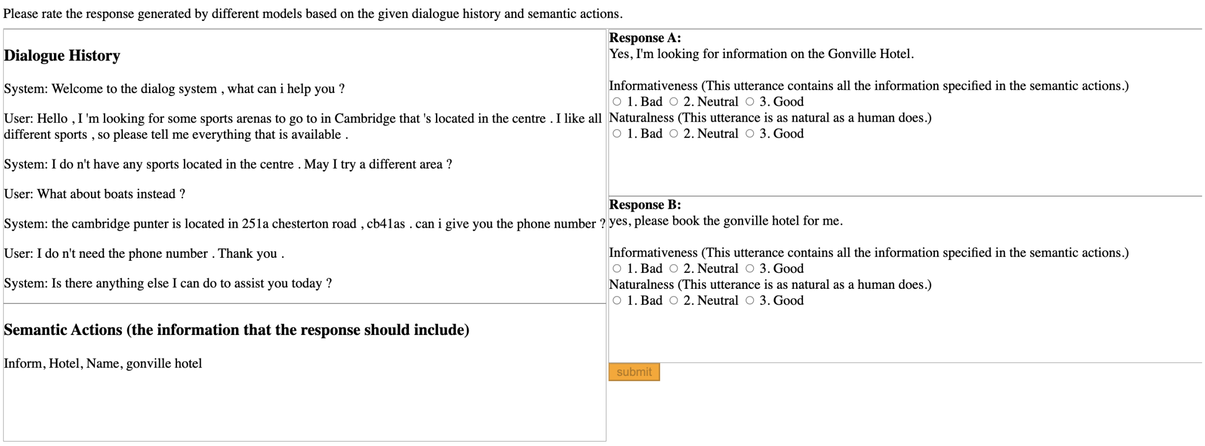
We can notice that GenTUS provides a natural response in turn 5. This utterance is difficult for the natural language understanding model and the system is not capable to handle it. As a result, GenTUS mentions the departure and destination explicitly in turn 8.

C An example of human preference test

The dialogue history, semantic actions, and utterances generated by two different systems are presented to workers on AMT platform. They are required to rate the informativeness and naturalness of these utterances respectively.

User goal	
attraction: {info: {type: college}, reqt: {postcode: ?, entrance fee: ?}}, hotel: {info: {area: north, star:0}, reqt: {parking:?}, taxi: {arriveby:13:00}, reqt:{phone:?, car type:?}}	
Dialogue	
0-sys	Welcome to the dialogue system. How can I help you?
0-usr	I'm looking for a college to visit in town.
1-sys	There 's 18 college in the centre , anything in mind ? What area would you like this in ?
1-usr	No, but I would like the postcode and entrance fee.
2-sys	We have christ's college. Its postcode is cb23bu . Would you be interested in visiting a college? The entry fee is free. The postcode is cb23bu. Did you need any further assistance today?
2-usr	Yes, I'm also looking for a place to stay in the north.
3-sys	I have 13 moderate guesthouse that fit your criteria. Want one of those? I 'd be happy to help. May I ask what price range and area of town you are looking for?
3-usr	I would like a 0 star rating.
4-sys	I have 1 option for you. It's listed as cheap. It is a guesthouse. Would city centre north b and b work for you?
4-usr	Does it have free parking?
5-sys	City centre north b and b is available would you like to try that ? There is wifi available at the hotel . The parking is free . The hotel is 0 stars . It is a guesthouse .
5-usr	**Great. I also need a taxi to commute between the two places.**
6-sys	Okay, I can help with that. What are your departure site and destination? Also, when would you like to leave or arrive by?
6-usr	I need to arrive by 13:00.
7-sys	There are 18. Would you like me to recommend one for you?
7-usr	I 'd like the car type and contact number please.
8-sys	There are 18 entries found for "college" in the centre of town. Would you like the first listing? How about the christ's college? Glad I could be of assistance! If you need anything else, let me know.
8-usr	**Yes, I need a taxi to get me from the college to the hotel by 13:00.**
...	

Table 11: A dialogue example of GenTUS interacting with rule-based dialogue system. GenTUS responds in a natural way utilising anaphora in turn 5. After it finds out the system is not capable to understand, GenTUS provides the departure and destination explicitly in turn 8.


Please rate the response generated by different models based on the given dialogue history and semantic actions.

Dialogue History

System: Welcome to the dialog system , what can i help you ?

User: Hello , I 'm looking for some sports arenas to go to in Cambridge that 's located in the centre . I like all different sports , so please tell me everything that is available .

System: I do n't have any sports located in the centre . May I try a different area ?

User: What about boats instead ?

System: the cambridge punter is located in 251a chesterton road , cb41as . can i give you the phone number ?

User: I do n't need the phone number . Thank you .

System: Is there anything else I can do to assist you today ?

Semantic Actions (the information that the response should include)

Inform, Hotel, Name, gonville hotel

Response A:
Yes, I'm looking for information on the Gonville Hotel.

Informativeness (This utterance contains all the information specified in the semantic actions.)
○ 1. Bad ○ 2. Neutral ○ 3. Good
Naturalness (This utterance is as natural as a human does.)
○ 1. Bad ○ 2. Neutral ○ 3. Good

Response B:
yes, please book the gonville hotel for me.

Informativeness (This utterance contains all the information specified in the semantic actions.)
○ 1. Bad ○ 2. Neutral ○ 3. Good
Naturalness (This utterance is as natural as a human does.)
○ 1. Bad ○ 2. Neutral ○ 3. Good

submit

Figure 3: An example of human preference test.

Chapter 6

EmoUS: Simulating User Emotions in Task-Oriented Dialogues

This chapter summarises our work on simulating user emotions in task-oriented dialogues and gives a verbatim copy of our paper (Lin et al., 2023):

Hsien-Chin Lin et al. (2023). "EmoUS: Simulating User Emotions in Task-Oriented Dialogues". In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '23. Taipei, Taiwan: Association for Computing Machinery, pp. 25262531. ISBN: 9781450394086. DOI: 10.1145/3539618.3592092. URL: https://doi.org/10.1145/3539618.3592092

6.1 Summary

State-of-the-art user simulators in task-oriented dialogues only simulate extrinsic user behaviour, e.g. user semantic actions or user utterances in natural language. Without modelling user personas or emotions, user simulators cannot capture the diverse range of user behaviour influenced by user intrinsic states, resulting in emotionless natural language responses with limited linguistic diversity. In addition, dialogue systems optimised on interactions with these neutral user simulators tend to focus only on transactional aspects driven by task success, neglecting the user's emotional state. This limitation may lead to a high drop-off rate when deployed to real customers.

To effectively address this issue, we introduce EmoUS, a user simulator for task-oriented dialogue systems that learns to simulate user emotions with user behaviour. EmoUS generates user emotions based on the dialogue context, e.g. user goals, system actions from the previous turn, the user history, and the user persona. After generating the user emotion, EmoUS generates semantic actions and utterances in natural language. In other words, the behaviour of EmoUS is not only conditioned on the dialogue context but also on the user emotion and persona.

Our experimental results demonstrate that EmoUS can properly predict user emotions and generate more diverse natural language responses. Additionally, we analyse how different system behaviour elicit specific user emotions, showing that our user simulator can be used to evaluate dialogue systems in addition to task success and probe their impact on the user's emotional state. The development of such methods is crucial to the increasing usage of large language model chatbots and the growing ethical concerns surrounding them.

6.2 Personal contributions

All writing, implementation, and technical results are my contribution. Co-authors assisted in writing and proofreading.

EmoUS: Simulating User Emotions in Task-Oriented Dialogues

Hsien-Chin Lin
Shutong Feng
Christian Geishauser
linh@hhu.de
Heinrich Heine University Düsseldorf
Düsseldorf, Germany

Nurul Lubis
Carel van Niekerk
Michael Heck
Heinrich Heine University Düsseldorf
Düsseldorf, Germany

Benjamin Ruppik
Renato Vukovic
Milica Gašić
gasic@hhu.de
Heinrich Heine University Düsseldorf
Düsseldorf, Germany

ABSTRACT

Existing user simulators (USs) for task-oriented dialogue systems only model user behaviour on semantic and natural language levels without considering the user persona and emotions. Optimising dialogue systems with generic user policies, which cannot model diverse user behaviour driven by different emotional states, may result in a high drop-off rate when deployed in the real world. Thus, we present EmoUS, a user simulator that learns to simulate user emotions alongside user behaviour. EmoUS generates user emotions, semantic actions, and natural language responses based on the user goal, the dialogue history, and the user persona. By analysing what kind of system behaviour elicits what kind of user emotions, we show that EmoUS can be used as a probe to evaluate a variety of dialogue systems and in particular their effect on the user's emotional state. Developing such methods is important in the age of large language model chat-bots and rising ethical concerns.

CCS CONCEPTS

• **Human-centered computing → User models**; • **Computing methodologies → Discourse, dialogue and pragmatics**.

KEYWORDS

dialogue system, user simulation, emotion simulation

ACM Reference Format:

Hsien-Chin Lin, Shutong Feng, Christian Geishauser, Nurul Lubis, Carel van Niekerk, Michael Heck, Benjamin Ruppik, Renato Vukovic, and Milica Gašić. 2023. EmoUS: Simulating User Emotions in Task-Oriented Dialogues. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3539618.3592092

1 INTRODUCTION

Task-oriented dialogue systems (DSs) help users accomplish their goals, such as searching for nearby restaurants or booking a hotel. Proficient DSs are often trained via reinforcement learning (RL), which demands a large number of interactions between the system and users, making training with real users expensive and time-consuming. Therefore, utilizing user simulators (USs) to build a controlled interactive environment becomes attractive [6].

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07.
https://doi.org/10.1145/3539618.3592092

Despite recent USs in task-oriented dialogues properly modelling user extrinsic behaviour in terms of semantic actions and natural language [17, 36], a crucial aspect is still lacking: the user intrinsic state such as user persona and the emotional state. A generic user policy may lead to limited linguistic diversity and fails to capture diverse actions driven by varying user emotions. Adjusting the probability distribution of user actions in rule-based USs is a popular method to address diversity [13], but real users differ in more ways than just action preferences. Training USs by supervised learning with different initialisation [35] or by RL with varying reward functions can also form various user policies [17], but that can only provide diverse extrinsic behaviour, e.g. the action length in each turn or the semantic content.

In this work, we propose a user simulator that models the user emotional state conditioned on the dialogue context and the user persona. More specifically, our contributions are as follows:

- We propose an **emo**tional **u**ser **s**imulator that we call *EmoUS*[1]. The *EmoUS* response includes the user emotion, semantic actions, and natural language utterances. To the best of our knowledge, this is the first user simulator with user emotion for task-oriented dialogue systems.
- EmoUS exhibits an increased linguistic diversity for the same context by modelling the user policy and emotion jointly,
- The user emotion of EmoUS provides valuable insights for evaluating DSs, offering a more subtle and detailed understanding beyond a simple measure of task success.

2 RELATED WORK

The effectiveness of a task-oriented dialogue policy trained by RL with a US is greatly affected by the quality of the US [27]. Rule-based USs are commonly used to train DSs, such as the agenda-based US (ABUS) [28]. ABUS models the user goal as a stack-like agenda, ordered by the priority of the user actions updated by hand-crafted stacking and popping rules. While its action probability distribution can be manipulated to simulate different user behaviour [13], it only generates semantic actions without natural language generation or emotion prediction. Moreover, designing rules for complex scenarios is labour-intensive and transferring these rules to new domains can be challenging. To address these limitations, data-driven USs have been developed, which learn user policy directly from data. The sequence-to-sequence (Seq2Seq) model structure is the most common framework. The input sequence may include the dialogue history and user goal as a list of features or plain text. The output sequence can be semantic actions or natural language utterances [7, 10, 15, 17, 18, 37, 38]. Tang et al. [35] train USs by supervised

[1]https://gitlab.cs.uni-duesseldorf.de/general/dsml/emous-public

learning with different initialisation to create different user policies. Lin et al. [17] proposed GenTUS, an ontology-independent US which generates natural language utterances as well as the underlying semantic actions for interpretability. Its behaviour can be configured by RL with different reward functions. These USs can simulate extrinsic user behaviour, e.g. actions and utterances, but intrinsic user states are neglected, e.g. satisfaction level and emotional status [24].

In comparison to generating responses with given emotions [3, 21, 33] or recognising user satisfaction classification after receiving user utterances [1, 8, 11, 12, 30, 32], the user satisfaction modelling should predict intrinsic user states first then generates actions or utterances. Sun et al. [34] and Deng et al. [4] investigate how user satisfaction impacts user behaviour on the semantic level. Pan et al. [23] transfer the emotion from chit-chat to task-oriented dialogues utilising data augmentation. Kim and Lipani [14] proposed SatActUtt, which generates users' satisfaction, action (only with intent and domain), and utterance based on dialogue history as multi-task learning. We consider SatActUtt as our baseline as it is the first US modelling both intrinsic and extrinsic user behaviour. While SatActUtt can predict user satisfaction scores adequately based on dialogue history, it does not include the user goal. This makes it difficult to train a dialogue system. In addition, it only considers satisfaction and dissatisfaction, disregarding aspects such as different emotion elicitors or user personas [19, 22]. Feng et al. [9] annotated a task-oriented dialogue dataset with 7 user emotions inspired by Ortony, Clore and Collins (OCC) emotion model [22]. It considers user conduct and emotion elicitors for human-human and human-machine task-oriented dialogues, representing a more fine-grained user intrinsic state for task-oriented dialogues.

3 SIMULATING USER EMOTION IN TASK-ORIENTED DIALOGUES

Task-oriented DSs are underpinned by an *ontology* which is typically composed of *intents*, *domains*, *slots*, and *values*. *Intents* define user or system global intentions of their respective actions in each turn. Users and systems may have different intents, e.g., systems can *confirm* user's request and users can *negate* system's proposal. *Domains* are the topics that can be discussed in the conversation. They can be further specified by *slots* and each can take a number of *values*. We assume that the users of task-oriented dialogues have a *goal* they want to achieve, which is defined as $G = \{d_1 : [(s_1, v_1), (s_2, v_2), \ldots], d_2 : [(s_3, v_3), \ldots], \ldots\}$, where domain d_i, slot s_i and value v_i are selected from the ontology.

Semantic *user actions* and *system actions* are composed of tuples, $(intent, domain, slot, value)$. Semantic actions can be transformed into natural language utterances. User *emotion* in task-oriented dialogues may be triggered by different elicitors, or related to different user personas. For example, the system not responding adequately may lead to the user being dissatisfied [9]. A user *persona* represents users' attitudes and feelings towards certain events, such as feeling fearful after a robbery [20] or includes users' conduct, i.e. how users behave when communicating, e.g. are they polite or impolite. For example, the persona of a polite user who is feeling excited to visit a museum is $persona$ = {user: polite, attraction: excited}. The user persona can be derived from dialogue history during training and sampled from a distribution for inference.

User simulation with emotion can be viewed as a Seq2Seq problem. For each turn, EmoUS predicts the user emotion based on the context information, e.g. the dialogue history, the user goal, and the user persona, and generates semantic actions and natural language responses based as follows.

3.1 Model structure

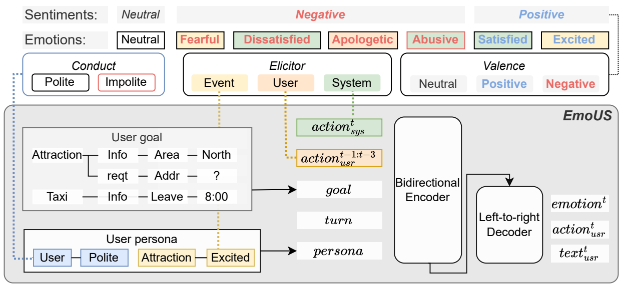


Figure 1: The model structure of EmoUS

EmoUS builds upon GenTUS and additionally incorporates user persona and emotion prediction. More specifically, EmoUS takes the system action $action_{sys}^t$, user history $action_{usr}^{t-1:t-3}$, user goal $goal$, turn information $turn$ and the user persona $persona$ as input and generates user emotion $emotion$, semantic actions $action_{usr}^t$, and an utterance $text_{usr}^t$ as output at turn t (as shown in Fig. 1). By introducing different user personas and emotions, more diverse user behaviours on both semantic and linguistic aspects can be simulated even in the same context.

EmoUS considers the three aspects of user emotions in task-oriented dialogues according to EmoWOZ [9], namely *elicitor*, *conduct*, and *valence*. The emotion elicitor can be an event, the system, or the user. Their respective information can be captured from the event $persona$ attribute, system $action_{sys}^t$, and user $action_{usr}^{t-1:t-3}$. The user conduct, whether polite or impolite, is recorded as a user persona. The valence aspect, or the sentiment polarity of each emotion, is informed intrinsically in the emotion prediction.

Following the setting in Lin et al. [17], the input and output sequences are represented as JSON-formatted strings, composed of natural language tokens. In this way, EmoUS achieves ontology independence and can transfer to unseen domains.[2] Then we train EmoUS as a Seq2Seq model and leverage BART [16], a transformer-based natural language generator with a bidirectional encoder and a left-to-right decoder. BART demonstrates impressive performance in a range of language-related tasks.

4 EXPERIMENTAL SETUP

The aim of our experiments is to demonstrate that EmoUS is able to generate user emotion, semantic actions, and utterances based on the context of the conversation and the user persona. Furthermore, we show that the emotion prediction of EmoUS is a valuable tool

[2]As this property is directly inherited from GenTUS, we do not examine it in our experiments.

for evaluating DSs. We conduct our experiments on EmoWOZ [9]. It contains user emotion annotations for human-human dialogues from MultiWOZ [2] and $1k$ human-machine dialogues between volunteers and an RNN-based dialogue policy trained on MultiWOZ.

4.1 Supervised learning for emotion simulation

Our model is inherited from Huggingface's transformers [39] and trained on EmoWOZ. To measure the emotion prediction performance, we calculate the macro-F1 score of sentiments and emotions. We compare sentiment prediction against SatActUtt [14], a user model which predicts sentiment, user action (composed with intent and domain only), and utterances based on the dialogue history.

Following the setting of Lin et al. [17], we evaluate the performance of modelling user semantic actions by F1 score and turn accuracy and the natural language generation (NLG) performance by slot error rate (SER), sacre-BLEU score [25] and self-BLEU score [41]. SER measures the agreement between the semantic actions and the corresponding utterance. $SER = (m + h)/N$, where N is the total number of slots in semantic actions, m and h stand for the number of missing and hallucinated slots. The self-BLEU evaluates the diversity of generated utterances in the following way. After generating a sentence for every data point, we calculate a BLEU score by treating all other generated sentences as references. By averaging these scores, we get the self-BLEU score where the lower score implies a higher diversity.

4.2 Interacting with DS

We estimate the generalisation ability of a US by cross-model evaluation, where a DS trained with a particular US is evaluated by different USs [29]. Policies of different DSs are trained with various USs, including the agenda-based US (ABUS) with T5 [26] natural language generator (ABUS-T5), GenTUS, and EmoUS, by proximal policy optimisation (PPO) [31], a simple and stable RL algorithm, for 200 epochs, each of which consists of 1000 dialogue turns. Each policy is trained on 5 random seeds and the performance is averaged. The DSs also include a natural language understanding module composed with BERT [5] for understanding users' utterances and a rule-based dialogue state tracker for tracking the users' states under the ConvLab-3 framework [40].

We also analyse how different system behaviour elicit user emotions. For this purpose, we used $1k$ dialogues between EmoUS and a DS trained by EmoUS. We categorised various system behaviour in the following groups: *confirm* - the system repeats the slots and values informed by the user, *no_confirm* - the system does not repeat this information, *miss_info* - the system requests the information just mentioned by the user, *neglect* - the system does not respond to the user request, *reply* - the system responds to the user request, and *loop* - the system takes identical actions for two turns in a row.

5 EXPERIMENTAL RESULTS

5.1 User emotion modelling

As shown in Table 1, EmoUS outperforms SatActUtt on sentiment prediction by 0.314 on macro-F1 score. This is not unexpected as EmoUS includes the user goal in inputs and the user sentiment in task-oriented dialogues is centred around the user goal [9]. In addition, the performance of sentiment prediction between EmoUS and EmoUS$_{noPersona}$ is similar, but the emotion prediction improves by 0.202 on the macro-F1 score when including the user persona. This indicates that considering the user persona improves the performance of user emotions triggered by different elicitors.

Table 1: Performance for emotion and sentiment prediction of different models by measuring macro-F1 score.

model	sentiment	emotion
SatActUtt	0.379	-
EmoUS$_{noPersona}$	0.673	0.299
EmoUS	**0.693**	**0.501**

We demonstrate that user emotion simulation can be further configured by multiplying different weights w on the probability of *neutral*, i.e. *neutral* is more likely to be selected with a higher weight. As shown in Fig. 2, EmoUS is purely neutral without any emotion as $w = 1.5$. As the weight decreases, EmoUS achieves the best performance on *fearful* as $w = 0.95$, best on *dissatisfied* as $w = 0.9$, and best on *apologetic* as $w = 0.85$. Thus, we can change the probability distribution of emotions in the user response, inducing different user behaviour, by modifying the weight of emotions.

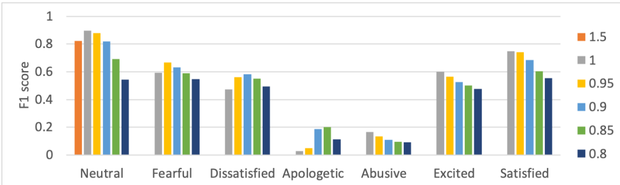


Figure 2: Different weights of the neutral emotion will have different F1-score on each user emotion.

5.2 User action prediction

The results of user action prediction are shown in Table 2, where EmoUS$_{emo}$ generates semantic actions based on golden emotions. EmoUS is superior to SatActUtt because EmoUS can generate semantic actions following the user goal, whereas SatActUtt does not consider the user goal. Additionally, EmoUS is still comparable to GenTUS despite it models a more complex task, simulating user emotions and semantic actions jointly. Moreover, EmoUS$_{emo}$ surpasses GenTUS since EmoUS$_{emo}$ generates semantic actions utilising more information then GenTUS, such as the user persona and golden emotions.

5.3 Natural language evaluation

The NLG results are shown in Table 3, where GenTUS$_{act}$ generates utterances based on golden semantic actions and EmoUS$_{emo+act}$ is based on golden emotion and semantic actions. On the other hand, GenTUS and EmoUS are generated based on their prediction. The Sacre-BLEU is calculated with golden utterances.

Although SatActUtt generates the most diverse utterances with the lowest Self-BLEU score, it also has the lowest Sacre-BLEU score,

Table 2: Performance for user action prediction.

model	Intents+domains F1	Intents+domains ACC	Full action F1	Full action ACC
GenTUS	0.890	0.854	0.762	0.600
SatActUtt	0.317	0.221	-	-
EmoUS	0.892	0.857	0.764	0.603
EmoUS$_{emo}$	**0.904**	**0.867**	**0.775**	**0.611**

which means it by and large generates random responses irrelevant to the user goal. On the other hand, EmoUS$_{emo+act}$ has a comparable Sacre-BLEU and SER with GenTUS$_{act}$ and a much lower Self-BLEU score, which means EmoUS is able to generate more diverse responses than GenTUS but still follows the user goal and maintains the agreement between the semantics and the language.

Table 3: The NLG performance on EmoWOZ of different models. The arrow directions represent which trend is better.

model	SER↓	Sacre-BLEU↑	Self-BLEU↓
Human	0.054	-	0.770
GenTUS	0.116	-	0.950
GenTUS$_{act}$	**0.092**	**19.61**	0.930
SatActUtt	-	2.90	**0.433**
EmoUS	0.118	-	0.715
EmoUS$_{emo+act}$	0.096	16.91	0.708

5.4 Cross-model evaluation

As shown in Table 4, the DS trained with EmoUS performs comparably to the DS trained with ABUS-T5 when evaluating with ABUS-T5 (0.62 vs 0.63 success rate), and similarly to the DS trained with GenTUS when evaluating with GenTUS (both at 0.53 success rate). However, the DS trained with EmoUS outperforms the DS trained with ABUS-T5 by 7% absolute and the DS trained with GenTUS 5% absolute on success rate when evaluating with EmoUS (success rates of 0.52 vs 0.45 and 0.47 respectively). This indicates that EmoUS provides a better learning environment and makes DSs trained with it perform well when evaluated on diverse USs.

Table 4: The success rates of policies trained on EmoUS, GenTUS, and ABUS with T5 NLG (ABUS-T5) when tested on various USs. Each pair is evaluated by 400 dialogues on 5 seeds, which is 2K dialogues in total.

US for training	US for evaluation: ABUS-T5	GenTUS	EmoUS
ABUS-T5	0.63	0.48	0.45
GenTUS	0.60	0.53	0.47
EmoUS	0.62	0.53	**0.52**

5.5 System behaviour with the user emotions

In $1k$ dialogues between EmoUS and a DS trained by it, the system behaviour $confirm$, $no\_confirm$, and $miss\_info$ elicit neutral emotion. As systems respond properly, e.g. *reply*, users are likely to feel satisfied, but when systems behave unprofessionally, e.g. *neglect* and *loop*, users may feel dissatisfied (see Table 5). This observation is in line with the user study conducted by Sun et al. [34].

Furthermore, we plot the average user sentiment per turn where positive = +1, neutral = 0, and negative = −1, for each dialogue outcome. As expected, users are more positive in successful dialogues and more negative in failed dialogues on average (see Fig. 3). In addition, we also notice a drop in sentiment around turn 6, which suggests the user may feel impatience after that.

Table 5: Proportion of neutral and system-eliciting emotions triggered by various system behaviour.

System behaviour	User emotion: Neutral	Dissatisfied	Satisfied
confirm	86.00%	2.20%	11.80%
no_confirm	71.80%	16.60%	11.60%
miss_info	79.20%	11.10%	9.70%
neglect	27.10%	65.00%	7.90%
reply	51.50%	4.10%	44.40%
loop	28.60%	65.90%	5.50%

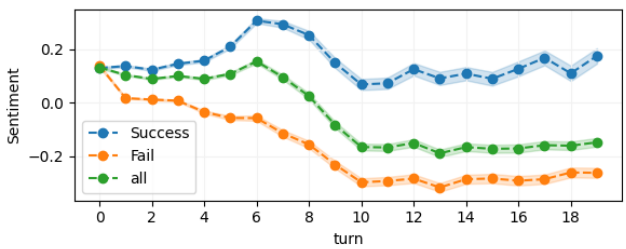


Figure 3: The average user sentiment in different turns.

6 CONCLUSION

We present EmoUS, a simulated user that generates emotional and thus more diverse output in task-oriented dialogues. It can be further configured by manipulating different weights for each emotion or different user personas. Our results show that EmoUS is useful to examine the impact of dialogue systems on the user's emotional state. Developing such probes is of particular importance with the increasing usage of dialogue systems and the rising ethical concerns of large language model chat-bots.

In future, the correlations between personas and emotions should be investigated, e.g. whether polite users show more satisfaction even though system responses are inadequate. Human evaluation should also be conducted to address the validity of our simulation. Furthermore, we plan to utilise EmoUS for the development of emotion-sensitive DSs.

REFERENCES

[1] Praveen Kumar Bodigutla, Lazaros Polymenakos, and Spyros Matsoukas. 2019. Multi-domain Conversation Quality Evaluation via User Satisfaction Estimation. arXiv:1911.08567 [cs.LG]

[2] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 5016–5026. https://doi.org/10.18653/v1/D18-1547

[3] Pierre Colombo, Wojciech Witon, Ashutosh Modi, James Kennedy, and Mubbasir Kapadia. 2019. Affect-Driven Dialog Generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 3734–3743. https://doi.org/10.18653/v1/N19-1374

[4] Yang Deng, Wenxuan Zhang, Wai Lam, Hong Cheng, and Helen Meng. 2022. User Satisfaction Estimation with Sequential Dialogue Act Modeling in Goal-Oriented Conversational Systems. In *Proceedings of the ACM Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 2998–3008. https://doi.org/10.1145/3485447.3512020

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[6] W. Eckert, E. Levin, and R. Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*. 80–87. https://doi.org/10.1109/ASRU.1997.658991

[7] Layla El Asri, Jing He, and Kaheer Suleman. 2016. A Sequence-to-Sequence Model for User Simulation in Spoken Dialogue Systems. *Interspeech 2016* (2016), 1151–1155.

[8] Klaus-Peter Engelbrecht, Florian Gödde, Felix Hartard, Hamed Ketabdar, and Sebastian Möller. 2009. Modeling User Satisfaction with Hidden Markov Models. In *Proceedings of the SIGDIAL 2009 Conference*. Association for Computational Linguistics, London, UK, 170–177. https://aclanthology.org/W09-3926

[9] Shutong Feng, Nurul Lubis, Christian Geishauser, Hsien-chin Lin, Michael Heck, Carel van Niekerk, and Milica Gasic. 2022. EmoWOZ: A Large-Scale Corpus and Labelling Scheme for Emotion Recognition in Task-Oriented Dialogue Systems. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, 4096–4113. https://aclanthology.org/2022.lrec-1.436

[10] Izzeddin Gür, Dilek Hakkani-Tür, Gokhan Tür, and Pararth Shah. 2018. User modeling for task oriented dialogues. In *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 900–906.

[11] Sunao Hara, Norihide Kitaoka, and Kazuya Takeda. 2010. Estimation Method of User Satisfaction Using N-gram-based Dialog History Model for Spoken Dialog System. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA), Valletta, Malta. http://www.lrec-conf.org/proceedings/lrec2010/pdf/579_Paper.pdf

[12] Ryuichiro Higashinaka, Yasuhiro Minami, Kohji Dohsaka, and Toyomi Meguro. 2010. Modeling User Satisfaction Transitions in Dialogues from Overall Ratings. In *Proceedings of the SIGDIAL 2010 Conference*. Association for Computational Linguistics, Tokyo, Japan, 18–27. https://aclanthology.org/W10-4304

[13] Simon Keizer, Milica Gašić, Filip Jurčíček, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Parameter estimation for agenda-based user simulation. In *Proceedings of the SIGDIAL 2010 Conference*. Association for Computational Linguistics, Tokyo, Japan, 116–123. https://aclanthology.org/W10-4323

[14] To Eun Kim and Aldo Lipani. 2022. A multi-task based neural model to simulate users in goal oriented dialogue systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2115–2119.

[15] Florian Kreyssig, Iñigo Casanueva, Paweł Budzianowski, and Milica Gašić. 2018. Neural User Simulation for Corpus-based Policy Optimisation of Spoken Dialogue Systems. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. Association for Computational Linguistics, Melbourne, Australia, 60–69. https://doi.org/10.18653/v1/W18-5007

[16] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7871–7880.

[17] Hsien-chin Lin, Christian Geishauser, Shutong Feng, Nurul Lubis, Carel van Niekerk, Michael Heck, and Milica Gasic. 2022. GenTUS: Simulating User Behaviour and Language in Task-oriented Dialogues with Generative Transformers. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Edinburgh, UK, 270–282. https://aclanthology.org/2022.sigdial-1.28

[18] Hsien-chin Lin, Nurul Lubis, Songbo Hu, Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, and Milica Gasic. 2021. Domain-independent User Simulation with Transformers for Task-oriented Dialogue Systems. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Singapore and Online, 445–456. https://aclanthology.org/2021.sigdial-1.47

[19] Nurul Lubis, Sakriani Sakti, Graham Neubig, Tomoki Toda, Ayu Purwarianti, and Satoshi Nakamura. 2016. Emotion and its triggers in human spoken dialogue: Recognition and analysis. *Situated Dialog in Speech-Based Human-Computer Interaction* (2016), 103–110.

[20] François Mairesse and Marilyn Walker. 2006. Automatic Recognition of Personality in Conversation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. Association for Computational Linguistics, New York City, USA, 85–88. https://aclanthology.org/N06-2022

[21] Yanying Mao, Fei Cai, Yupu Guo, and Honghui Chen. 2022. Incorporating Emotion for Response Generation in Multi-Turn Dialogues. *Applied Intelligence* 52, 7 (may 2022), 7218–7229. https://doi.org/10.1007/s10489-021-02819-z

[22] Andrew Ortony, Gerald L. Clore, and Allan Collins. 1988. *The Cognitive Structure of Emotions*. Cambridge University Press. https://doi.org/10.1017/CBO9780511571299

[23] Yan Pan, Mingyang Ma, Bernhard Pflugfelder, and Georg Groh. 2022. User Satisfaction Modeling with Domain Adaptation in Task-oriented Dialogue Systems. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Edinburgh, UK, 630–636. https://aclanthology.org/2022.sigdial-1.59

[24] Soujanya Poria, Navonil Majumder, Rada Mihalcea, and Eduard Hovy. 2019. Emotion recognition in conversation: Research challenges, datasets, and recent advances. *IEEE Access* 7 (2019), 100943–100953.

[25] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics, Belgium, Brussels, 186–191. https://www.aclweb.org/anthology/W18-6319

[26] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[27] Jost Schatzmann, Kallirroi Georgila, and Steve Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*. 45–54.

[28] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Association for Computational Linguistics, Rochester, New York, 149–152. https://www.aclweb.org/anthology/N07-2038

[29] Jost Schatztnann, Matthew N Stuttle, Karl Weilhammer, and Steve Young. 2005. Effects of the user model on simulation-based learning of dialogue strategies. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005*. IEEE, 220–225.

[30] Alexander Schmitt and Stefan Ultes. 2015. Interaction quality: assessing the quality of ongoing spoken dialog interaction by experts—and how it relates to user satisfaction. *Speech Communication* 74 (2015), 12–36.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[32] Xiaohui Song, Liangjun Zang, Rong Zhang, Songlin Hu, and Longtao Huang. 2022. Emotionflow: Capture the Dialogue Level Emotion Transitions. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 8542–8546. https://doi.org/10.1109/ICASSP43922.2022.9746464

[33] Zhenqiao Song, Xiaoqing Zheng, Lu Liu, Mu Xu, and Xuanjing Huang. 2019. Generating Responses with a Specific Emotion in Dialog. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 3685–3695. https://doi.org/10.18653/v1/P19-1359

[34] Weiwei Sun, Shuo Zhang, Krisztian Balog, Zhaochun Ren, Pengjie Ren, Zhumin Chen, and Maarten de Rijke. 2021. Simulating user satisfaction for the evaluation of task-oriented dialogue systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2499–2506.

[35] Zhiwen Tang, Hrishikesh Kulkarni, and Grace Hui Yang. 2021. High-Quality Dialogue Diversification by Intermittent Short Extension Ensembles. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 1861–1872. https://doi.org/10.18653/v1/2021.findings-acl.163

[36] Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable Dialogue Systems and User Simulators. In *Proceedings of the 59th Annual Meeting*

of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 152–166.

[37] Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable Dialogue Systems and User Simulators. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 152–166. https://doi.org/10.18653/v1/2021.acl-long.13

[38] Dazhen Wan, Zheng Zhang, Qi Zhu, Lizi Liao, and Minlie Huang. 2022. A Unified Dialogue User Simulator for Few-shot Data Augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 3788–3799. https://aclanthology.org/2022.findings-emnlp.277

[39] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

[40] Qi Zhu, Christian Geishauser, Hsien-chin Lin, Carel van Niekerk, Baolin Peng, Zheng Zhang, Michael Heck, Nurul Lubis, Dazhen Wan, Xiaochen Zhu, et al. 2022. ConvLab-3: A Flexible Dialogue System Toolkit Based on a Unified Data Format. *arXiv preprint arXiv:2211.17148* (2022).

[41] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1097–1100.

Chapter 7

Conclusions and Future Work

In this chapter, an overview of the key contributions made within this thesis is provided and potential research directions are pointed out. The research presented in this thesis aimed to improve user modelling in task-oriented dialogues, generating user responses in natural language and incorporating the simulation of user emotions with a domain-independent structure.

7.1 Results

Previous user simulators are domain-dependent, which means adapting a user model to an unseen domain is challenging. In addition, the user policy and natural language generation are optimised separately, which causes a suboptimal performance. Furthermore, simulating user extrinsic behaviour only, e.g. user actions and utterances, without user intrinsic status, e.g. the satisfaction level and user emotions, limits the diversity of the user behaviour.

In Chapter 4, a domain-independent transformer-based user simulator (TUS) is proposed to address the challenges of adapting user simulators for new domains. Unlike rule-based user simulators that demand manual rule rewriting, and domain-dependent simulators that necessitate feature representation redesign or complete model retraining, TUS employs a domain-independent feature representation, allowing it to adapt to a new domain in a zero-shot fashion. However, TUS generates user responses in semantic actions, not able to generate utterances in natural language.

In Chapter 5, a generative transformer-based user simulator (GenTUS) is proposed, jointly optimising the user policy and natural language generation. GenTUS maintains its interpretability by generating user semantic actions and enriches linguistic diversity through utterance generation. Furthermore, the inputs and outputs of GenTUS are represented as word sequences, enabling it to adapt to a new ontology without feature modification and model fine-tuning. In addition, GenTUS generates user utterances conditioned not only on semantic actions but also on the dialogue context, resulting in more natural responses in comparison to a separated natural language generation module. Moreover, the behaviour of GenTUS can be further fine-tuned through reinforcement learning with different reward functions, allowing for the development of specific user simulators. Although GenTUS can generate natural user utterances, its diversity is limited and it only considers user extrinsic behaviour, e.g. semantic actions and utterances, without the user intrinsic states, e.g. user emotions.

In Chapter 6, a user simulator that learns to simulate user emotions with user extrinsic behaviour, EmoUS, is introduced. EmoUS predicts user emotions according to the dialogue context and the user persona, then it generates semantic actions and utterances based on the emotion, persona, and dialogue context, which means EmoUS considers the dialogue context as well as the user intrinsic status for modelling the extrinsic behaviour. The experimental results demonstrate that EmoUS can properly predict user emotions and produce more diverse natural language responses. Furthermore, according to how different system behaviour can trigger varying user emotions, EmoUS does not only evaluate dialogue system task success but also probes their influence on the user's emotional experience, which is important due to the growing demand for large language model chatbots and the ethical concerns associated with their usage.

In this thesis, the user simulation of task-oriented dialogues can capture a broader spectrum of human behaviour, including not only extrinsic user behaviour but also intrinsic user status, ranging from neutral semantic actions to emotional natural language utterances. A more human-like user modelling should encourage researchers to leverage user simulators for developing dialogue systems, improving the coverage rate of training and evaluation, and resolving ethical considerations.

7.2 Future work

Developing an appropriate user simulation for interactive systems is still an unsolved question. Here are some possible research topics:

Safety of interactive systems. With the increasing integration of dialogue systems into our daily lives, the concern of ethical issues is on the rise. If the development of dialogue systems only focuses on technical aspects while overlooking ethical considerations, people might avoid using these systems, or even start protesting against them (Kimon Kieslich and Došenović, 2023). However, detecting problematic system behaviours, e.g. abusive language, misleading suggestion, and biased information, by human annotators is not trivial, since offensive words may harm people (Jay, 2009). The results obtained in this thesis open the door to investigating the use of a user simulator to assert safety of a dialogue system. Namely, a user simulation, which models user intrinsic behaviour, would be capable of detecting and evaluating unfavourable system responses, indicating when and why people experience dissatisfaction or frustration while interacting with the system. As a result, enhancing the modelling of negative emotions such as "abusive" and "dissatisfied" in user simulation is essential to build a safe and well-behaved ethical system.

What influence does the internal state have on external behaviour? In Chapter 6, different system behaviour elicit diverse user emotions. However, how user intrinsic status impacts external behaviour is not clear, e.g. will patient users still be satisfied even when the system behaves unprofessionally? It is also useful to integrate user emotions in addition to reward signals or conditions for utterance generation, since a desirable user experience is not only about the task success but also about the language style and the system conduct, e.g. the politeness of task-oriented dialogue systems is emphasised to prevent users from getting annoyed, especially when systems frequently request users' personal information to complete tasks (Kurz et al., 2021; Mishra et al., 2022).

How to incorporate various sources of data? In this thesis, user simulators are trained on dialogue corpora, e.g. MultiWOZ (Budzianowski et al., 2018) and schema-guided dialogues (Rastogi et al., 2020). These datasets come with detailed labels, such as system actions, user actions, and dialogue states, where manual annotation of these datasets is a time-consuming and labour-intensive process. Conversely, there exists a vast amount of unstructured data on the internet, e.g. restaurant reviews or travel blogs, which are rich in content but unstructured and lack annotations. Therefore, it is appealing to investigate how to learn user behaviour not only from well-annotated datasets, which are structured but limited and potentially outdated, but also from diverse sources of data that are noisy, unstructured, yet more current and comprehensive. In addition, considering multi-modal user behaviour is important, e.g. clicks during browsing, speech with emotional expressions, and facial expressions, for modelling a more human-like behaviour.

Large language model in user simulation. Large language models (LLMs) achieve impressive results on various natural language processing tasks (Brown et al., 2020). However, there are several issues when including LLMs in user simulation, which means LLMs are not a solution but rather an opportunity (Heck et al., 2023). The challenges are as follows:

- **Training and serving an LLM are extremely costly.** The considerable computational demands arising from the extensive scale of LLMs can lead to substantial latency and various related issues due to the numerous parameters involved (Ma et al., 2023). Therefore, it is challenging to fine-tune an LLM as a user simulator and to train a dialogue system with LLM-based user simulators by reinforcement learning.
- **Hallucination in response.** Although LLMs produce impressive responses, they still generate unexpected content, referred to as a hallucination (Bang et al., 2023). In other words, it is not trivial to properly control an LLM to follow given user goals.
- **Diverstion from natural language.** While LLMs are valuable for data augmentation when resources are scarce or unavailable (Bayer et al., 2022; Huang et al., 2022; Yoo et al., 2021), training LLMs on machine-generated text can introduce challenges, e.g. *model collapse*, referring to the model begins to forget the true underlying distribution of real-world data and becomes poisoned by machine-generated synthetic data (Shumailov et al., 2023). In addition, although learning from machine-to-machine interaction has demonstrated impressive breakthroughs, it may cause un-humanlike behaviour. For example, AlphaGo, optimised by self-play, has established strategies which professional players would never adopt (Bory, 2019). These moves are creative and intriguing, but it is crucial to prevent dialogue systems from generating machine-style language that deviates from natural language during training with an LLM-based user simulator.

As a result, it is essential to investigate how and when to use LLMs in user simulation, find a balance between performance, latency, and cost, and set up a mechanism to check the response of user simulators is correct and natural, without hallucination and machine-style utterances.

7.2.1 Bridging the gap between humans and machines

User simulation can do more than train dialogue systems or provide quantitative evaluation such as task success rate or satisfaction scores. By leveraging LLMs, user simulators may generate qualitative feedback, helping refine the user journey and improve the design of the system. With proper prompting or fine-tuning on specific datasets, user simulation may represent various users' backgrounds, e.g. people who are not familiar with technology or from different cultures, for addressing diverse user needs and developing user-centred systems. In addition, we can introduce different guidelines or regulations in user simulation, e.g. General Data Protection Regulation[1], to avoid any violation in the system since the early development process.

User simulation is not only essential in the development of dialogue systems but can also play an important role during serving. User simulators can be viewed as an *experienced user* since they have interacted with systems countless times. Therefore, they can help cold-start users get familiar with the system or simplify the user's complex requests. Furthermore, individuals can have their personalised simulator, which learns from their daily routines and can proactively request dialogue systems for them. Through future research in user simulation, human-computer interaction can be reformulated, shifting from a two-agent paradigm, consisting of just the system and user, to a multi-agent framework. The gap between humans and machines can be bridged in more than one way but with numerous possibilities and opportunities.

[1] https://gdpr-info.eu/

Appendix A

Gradient Vanishing

Gradient Vanishing is one challenge for optimising deep neural networks. Given a loss function $\mathcal{L}$, a deep neural network with L layers and the lth layer f_l can be formulated as:

$$\boldsymbol{h}_l = f_l(\boldsymbol{h}_{l-1}) = \phi(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l), \tag{A.1}$$

where ϕ is the activation function, $\boldsymbol{z}_l = \boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l$ is the linear sum of the layer input, $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ are the weights and bias, the gradient of the first layer weight can be calculated by chain rule:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_1} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_L} \frac{\partial \boldsymbol{h}_L}{\partial \boldsymbol{z}_L} \frac{\partial \boldsymbol{z}_L}{\partial \boldsymbol{h}_{L-1}} \frac{\partial \boldsymbol{h}_{L-1}}{\partial \boldsymbol{z}_{L-1}} \cdots \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{W}_1} \tag{A.2}$$

For simplicity, the bias $\boldsymbol{b}$ for each layer is set to $\boldsymbol{0}$. The gradient includes three parts, the derivative of the loss function $\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_L}$, the derivatives of the activation functions, e.g. $\phi' = \frac{\partial \boldsymbol{h}_L}{\partial \boldsymbol{z}_L}$, and the weights of layers, e.g. $\boldsymbol{W}_L = \frac{\partial \boldsymbol{z}_L}{\partial \boldsymbol{h}_{L-1}}$. The gradient vanishing problem is caused by these repeated multiplications, which means if these values are very small, the gradient will quickly reach 0. For example, the derivative of the Sigmoid function (Equation 2.2) is

$$\phi'_{\texttt{Sigmoid}}(x) = \phi_{\texttt{Sigmoid}}(x_i)\,(1 - \phi_{\texttt{Sigmoid}}(x_i)) \tag{A.3}$$

As shown in Figure A.1, the Sigmoid function suffers from gradient vanishing problem because of $\phi'_{\texttt{Sigmoid}}(x) < 1$.

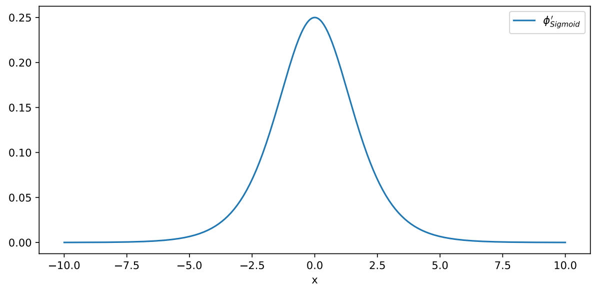


FIGURE A.1: The derivative of the Sigmoid function.

Appendix B

Supplementary Proofs

B.1 Positional Encoding

Theorem 1. *In the position j, the d-dimensional positional encoding $\boldsymbol{p}_j \in \mathbb{R}^d$ is defined as*

$$p_{j,i} = \begin{cases} \sin(\omega_k \cdot j), & \text{if } i = 2k \\ \cos(\omega_k \cdot j), & \text{if } i = 2k+1, \end{cases} \tag{B.1}$$

where $\omega_k = \frac{1}{10000^{2k/d}}$ and $k = 0, 1, \dots, \lceil \frac{d}{2} \rceil - 1$. For any fixed offset $\delta \in \mathbb{N}$, a linear transformation $\boldsymbol{T}^\delta \in \mathcal{R}^{d \times d}$ exists, such that

$$\boldsymbol{T}^\delta \boldsymbol{p}_j = \boldsymbol{p}_{j+\delta}. \tag{B.2}$$

Proof. According to the addition theorems of sine and cosine functions, for every ω_k

$$\begin{aligned} \sin(\omega_k \cdot (j+\delta)) &= \cos(\omega_k \cdot \delta)\sin(\omega_k \cdot j) + \sin(\omega_k \cdot \delta)\cos(\omega_k \cdot j) \\ \cos(\omega_k \cdot (j+\delta)) &= -\sin(\omega_k \cdot \delta)\sin(\omega_k \cdot j) + \cos(\omega_k \cdot \delta)\cos(\omega_k \cdot j), \end{aligned} \tag{B.3}$$

which can be reformulated as

$$\begin{bmatrix} \cos(\omega_k \cdot \delta) & \sin(\omega_k \cdot \delta) \\ -\sin(\omega_k \cdot \delta) & \cos(\omega_k \cdot \delta) \end{bmatrix} \begin{bmatrix} \sin(\omega_k \cdot j) \\ \cos(\omega_k \cdot j) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k \cdot (j+\delta)) \\ \cos(\omega_k \cdot (j+\delta)) \end{bmatrix} \tag{B.4}$$

If we define a projection matrix $\Phi_k^\delta \in \mathcal{R}^{2 \times 2}$ as

$$\Phi_k^\delta = \begin{bmatrix} \cos(\omega_k \cdot \delta) & \sin(\omega_k \cdot \delta) \\ -\sin(\omega_k \cdot \delta) & \cos(\omega_k \cdot \delta) \end{bmatrix} \tag{B.5}$$

where the projection matrix does not depend on any position index j, then the Equation B.4 can be reformulated as

$$\Phi_k^\delta \begin{bmatrix} p_{j,2k} \\ p_{j,2k+1} \end{bmatrix} = \begin{bmatrix} p_{j+\delta,2k} \\ p_{j+\delta,2k+1} \end{bmatrix} \tag{B.6}$$

which means the linear transformation $\boldsymbol{T}^\delta$ in Equation B.2 is

$$\boldsymbol{T}^\delta = \begin{bmatrix} \Phi_0^\delta & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \Phi_1^\delta & \dots & \mathbf{0} \\ \dots & \dots & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \dots & \Phi^\delta_{\lceil \frac{d}{2} \rceil - 1} \end{bmatrix} \tag{B.7}$$

□

Theorem 2. *In the position j, the norm of the d-dimensional positional encoding $\boldsymbol{p}_j \in \mathbb{R}^d$ is*

$$\|\boldsymbol{p}_j\|_2 = \sqrt{\frac{d}{2}} \tag{B.8}$$

Proof. According to $\sin^2(x) + \cos^2(x) = 1$, the norm of the positional encoding is

$$\begin{aligned} \|\boldsymbol{p}_j\|_2 &= \sqrt{\sin^2(\omega_0 \cdot j) + \cos^2(\omega_0 \cdot j) + \cdots + \sin^2(\omega_{\lceil \frac{d}{2} \rceil - 1} \cdot j) + \cos^2(\omega_{\lceil \frac{d}{2} \rceil - 1} \cdot j)} \\ &= \sqrt{\frac{d}{2}} \end{aligned} \tag{B.9}$$

□

Theorem 3. *The distance between two positional encodings is symmetric. For any fixed offset $\delta \in \mathbb{N}$,*

$$\|\boldsymbol{p}_j - \boldsymbol{p}_{j+\delta}\|_2 = \|\boldsymbol{p}_j - \boldsymbol{p}_{j-\delta}\|_2 \tag{B.10}$$

Proof. According to the addition theorem $\cos(x - y) = \cos(x)\cos(y) + \sin(x)\sin(y)$ and $\cos(x) = \cos(-x)$, we can find out

$$\begin{aligned} &\|\boldsymbol{p}_j - \boldsymbol{p}_{j+\delta}\|_2^2 \\ &= \|\boldsymbol{p}_j\|_2^2 + \|\boldsymbol{p}_{j+\delta}\|_2^2 - 2 \sum_{k=0}^{\lceil \frac{d}{2} \rceil - 1} (\cos(\omega_k \cdot j)\cos(\omega_k \cdot (j+\delta)) + \sin(\omega_k \cdot j)\sin(\omega_k \cdot (j+\delta))) \\ &= d - 2 \sum_{k=0}^{\lceil \frac{d}{2} \rceil - 1} \cos(\omega_k \cdot j - \omega_k \cdot (j+\delta)) \\ &= d - 2 \sum_{k=0}^{\lceil \frac{d}{2} \rceil - 1} \cos(-\omega_k \cdot j + \omega_k \cdot (j+\delta)) \\ &= d - 2 \sum_{k=0}^{\lceil \frac{d}{2} \rceil - 1} \cos(\omega_k \cdot j - \omega_k \cdot (j-\delta)) \\ &= \|\boldsymbol{p}_j\|_2^2 + \|\boldsymbol{p}_{j-\delta}\|_2^2 - 2 \sum_{k=0}^{\lceil \frac{d}{2} \rceil - 1} (\cos(\omega_k \cdot j)\cos(\omega_k \cdot (j-\delta)) + \sin(\omega_k \cdot j)\sin(\omega_k \cdot (j-\delta))) \\ &= \|\boldsymbol{p}_j - \boldsymbol{p}_{j-\delta}\|_2^2 \end{aligned} \tag{B.11}$$

□

B.2 Bias Correction for adaptive moment estimation

The moving average of gradients $\boldsymbol{m}$ and the squared gradients $\boldsymbol{v}$ in the adaptive moment estimation (Adam) is initialised to 0, i.e. $\boldsymbol{m}_0 = \boldsymbol{0}, \boldsymbol{v}_0 = \boldsymbol{0}$, which makes $\boldsymbol{m}$ and $\boldsymbol{v}$ bias on zero during the initial epochs. Therefore, the bias correction is introduced. The bias correction of $\boldsymbol{v}$ is explained as an example.

Let $\boldsymbol{g}$ be the gradient of a loss function $\mathcal{L}$, i.e. $\boldsymbol{g} = \nabla\mathcal{L}$, and $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k$ are the gradients in epoch 1 to k, each of them is from distribution $p(\boldsymbol{g}_k)$, i.e. $\boldsymbol{g}_k \sim p(\boldsymbol{g}_k)$. The moving average of the squared gradients $\boldsymbol{v}_k = \beta_2 \cdot \boldsymbol{v}_{t-1} + (1-\beta_2) \cdot \boldsymbol{g}_k^2$, where $\beta_2 \in [0, 1)$ is the decay rate and $\boldsymbol{g}_k^2 = \boldsymbol{g} \odot \boldsymbol{g}$ is the elementwise square, can be formulated as:

$$\boldsymbol{v}_k = (1-\beta_2) \sum_{i=1}^{k} \beta_2^{k-i} \cdot \boldsymbol{g}_i^2 \tag{B.12}$$

where β_2^k is the β_2 in the power of k. The relationship of the expected value of $\boldsymbol{v}_k$ and the expected value of $\boldsymbol{g}_k$ at epoch k is

$$\begin{aligned} E[\boldsymbol{v}_k] &= E\left[(1-\beta_2) \sum_{i=1}^{k} \beta_2^{k-i} \cdot \boldsymbol{g}_i^2\right] \\ &= E\left[\boldsymbol{g}_i^2\right] \cdot (1-\beta_2) \sum_{i=1}^{k} \beta_2^{k-i} + \zeta \\ &= E\left[\boldsymbol{g}_i^2\right] \cdot (1-\beta_2^k) + \zeta \end{aligned} \tag{B.13}$$

where $\zeta = 0$ if $E\left[\boldsymbol{g}_i^2\right]$ is stationary, or the β_2 can be chosen small to keep ζ small. Then the normalised moving average of the squared gradients is

$$\hat{\boldsymbol{v}}_k = \frac{\boldsymbol{v}_k}{1-\beta_2^k} \tag{B.14}$$

Bibliography

Arrieta, Alejandro Barredo et al. (2020). “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. In: *Information fusion* 58, pp. 82–115.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). “Layer normalization”. In: *arXiv preprint arXiv:1607.06450*.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1409.0473.

Bang, Yejin et al. (2023). *A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity*. arXiv: 2302.04023 [cs.CL].

Bayer, Markus, Marc-André Kaufhold, and Christian Reuter (Dec. 2022). “A Survey on Data Augmentation for Text Classification”. In: *ACM Comput. Surv.* 55.7. ISSN: 0360-0300. DOI: 10.1145/3544558. URL: https://doi.org/10.1145/3544558.

Bick, Daniel and MA Wiering (2021). “Towards Delivering a Coherent Self-Contained Explanation of Proximal Policy Optimization”. PhD thesis. Masters thesis, 2021.[Online]. Available: https://fse.studenttheses.ub.rug.nl/25709/.

Bodigutla, Praveen Kumar, Aditya Tiwari, Spyros Matsoukas, Josep Valls-Vargas, and Lazaros Polymenakos (Nov. 2020). “Joint Turn and Dialogue level User Satisfaction Estimation on Multi-Domain Conversations”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 3897–3909. DOI: 10.18653/v1/2020.findings-emnlp.347. URL: https://aclanthology.org/2020.findings-emnlp.347.

Bordes, Antoine, Y-Lan Boureau, and Jason Weston (2017). “Learning End-to-End Goal-Oriented Dialog”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=S1Bb3D5gg.

Bory, Paolo (2019). “Deep new: The shifting narratives of artificial intelligence from Deep Blue to AlphaGo”. In: *Convergence* 25.4, pp. 627–642. DOI: 10.1177/1354856519829679. eprint: https://doi.org/10.1177/1354856519829679. URL: https://doi.org/10.1177/1354856519829679.

Brown, Tom et al. (2020). “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Budzianowski, Paweł, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić (Oct. 2018). “MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium:

Association for Computational Linguistics, pp. 5016–5026. DOI: 10.18653/v1/D18-1547. URL: https://www.aclweb.org/anthology/D18-1547.

Castelvecchi, Davide (2016). "Can we open the black box of AI?" In: *Nature News* 538.7623, p. 20.

Chiang, Cheng-Han, Yung-Sung Chuang, and Hung-yi Lee (Nov. 2022). "Recent Advances in Pre-trained Language Models: Why Do They Work and How Do They Work". In: *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: Tutorial Abstracts*. Taipei: Association for Computational Linguistics, pp. 8–15. URL: https://aclanthology.org/2022.aacl-tutorials.2.

Chowdhery, Aakanksha et al. (2022). "Palm: Scaling language modeling with pathways". In: *arXiv preprint arXiv:2204.02311*.

Colombo, Pierre, Wojciech Witon, Ashutosh Modi, James Kennedy, and Mubbasir Kapadia (June 2019). "Affect-Driven Dialog Generation". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 3734–3743. DOI: 10.18653/v1/N19-1374. URL: https://aclanthology.org/N19-1374.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4, pp. 303–314.

Deng, Yang, Wenxuan Zhang, Wai Lam, Hong Cheng, and Helen Meng (2022). "User Satisfaction Estimation with Sequential Dialogue Act Modeling in Goal-Oriented Conversational Systems". In: *Proceedings of the ACM Web Conference 2022*. WWW '22. Virtual Event, Lyon, France: Association for Computing Machinery, pp. 29983008. ISBN: 9781450390965. DOI: 10.1145/3485447.3512020. URL: https://doi.org/10.1145/3485447.3512020.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

Dey, Rahul and Fathi M Salem (2017). "Gate-variants of gated recurrent unit (GRU) neural networks". In: *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE, pp. 1597–1600.

Dong, Qingxiu et al. (2023). *A Survey on In-context Learning*. arXiv: 2301.00234 [cs.CL].

Dong, Yihe, Jean-Baptiste Cordonnier, and Andreas Loukas (2021). "Attention is not all you need: Pure attention loses rank doubly exponentially with depth". In: *International Conference on Machine Learning*. PMLR, pp. 2793–2803.

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7.

El Asri, Layla, Jing He, and Kaheer Suleman (2016). "A Sequence-to-Sequence Model for User Simulation in Spoken Dialogue Systems". In: *Interspeech 2016*, pp. 1151–1155.

Elman, Jeffrey L (1990). "Finding structure in time". In: *Cognitive science* 14.2, pp. 179–211.

Engelbrecht, Klaus-Peter, Florian Gödde, Felix Hartard, Hamed Ketabdar, and Sebastian Möller (Sept. 2009). "Modeling User Satisfaction with Hidden Markov Models". In: *Proceedings of the SIGDIAL 2009 Conference*. London, UK: Association for Computational Linguistics, pp. 170–177. URL: https://aclanthology.org/W09-3926.

Fischer, Gerhard (2001). "User Modeling in Human-Computer Interaction". In: *User Model. User Adapt. Interact.* 11.1-2, pp. 65–86. DOI: 10.1023/A:1011145532042. URL: https://doi.org/10.1023/A:1011145532042.

Fukushima, Kunihiko (1969). "Visual feature extraction by a multilayered network of analog threshold elements". In: *IEEE Transactions on Systems Science and Cybernetics* 5.4, pp. 322–333.

Geishauser, Christian, Carel van Niekerk, Hsien-chin Lin, Nurul Lubis, Michael Heck, Shutong Feng, and Milica Gašić (Oct. 2022). "Dynamic Dialogue Policy for Continual Reinforcement Learning". In: *Proceedings of the 29th International Conference on Computational Linguistics*. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, pp. 266–284. URL: https://aclanthology.org/2022.coling-1.21.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (Apr. 2011). "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, pp. 315–323. URL: https://proceedings.mlr.press/v15/glorot11a.html.

Gross, James J (1998). "The emerging field of emotion regulation: An integrative review". In: *Review of general psychology* 2.3, pp. 271–299.

Gur, Izzeddin, Dilek Hakkani-Tür, Gökhan Tür, and Pararth Shah (2018). "User Modeling for Task Oriented Dialogues". In: *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018*. IEEE, pp. 900–906. DOI: 10.1109/SLT.2018.8639652. URL: https://doi.org/10.1109/SLT.2018.8639652.

Hara, Sunao, Norihide Kitaoka, and Kazuya Takeda (May 2010). "Estimation Method of User Satisfaction Using N-gram-based Dialog History Model for Spoken Dialog System". In: *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA). URL: http://www.lrec-conf.org/proceedings/lrec2010/pdf/579_Paper.pdf.

He, Junxian, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig (2022). "Towards a Unified View of Parameter-Efficient Transfer Learning". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. URL: https://openreview.net/forum?id=0RDcd5Axok.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

Heck, Michael, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, Shutong Feng, Christian Geishauser, Hsien-chin Lin, Carel van Niekerk, and Milica Gašić (July 2023). "ChatGPT for Zero-shot Dialogue State Tracking: A Solution or an Opportunity?" In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Toronto, Canada: Association for Computational Linguistics, pp. 936–950. DOI: 10.18653/v1/2023.acl-short.81. URL: https://aclanthology.org/2023.acl-short.81.

Henderson, Matthew, Blaise Thomson, and Steve Young (Aug. 2013). "Deep Neural Network Approach for the Dialog State Tracking Challenge". In: *Proceedings of the SIGDIAL 2013 Conference*. Metz, France: Association for Computational Linguistics, pp. 467–471. URL: https://aclanthology.org/W13-4073.

Higashinaka, Ryuichiro, Yasuhiro Minami, Kohji Dohsaka, and Toyomi Meguro (Sept. 2010). "Modeling User Satisfaction Transitions in Dialogues from Overall Ratings". In: *Proceedings of the SIGDIAL 2010 Conference*. Tokyo, Japan: Association for Computational Linguistics, pp. 18–27. URL: https://aclanthology.org/W10-4304.

Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky (2012). "Overview of mini-batch gradient descent". In: *Neural Networks for Machine Learning Lecture 6a* 575.8. URL: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hoffmann, Jordan et al. (2022). "Training compute-optimal large language models". In: *arXiv preprint arXiv:2203.15556*.

Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen (2022). "LoRA: Low-Rank Adaptation of Large Language Models". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

Huang, Jiaxin, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han (2022). "Large Language Models Can Self-Improve". In: *CoRR* abs/2210.11610. DOI: 10.48550/arXiv.2210.11610. arXiv: 2210.11610. URL: https://doi.org/10.48550/arXiv.2210.11610.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, pp. 448456.

Jain, Anil K, Jianchang Mao, and K Moidin Mohiuddin (1996). "Artificial neural networks: A tutorial". In: *Computer* 29.3, pp. 31–44.

Jay, Timothy (2009). "Do offensive words harm people?" In: *Psychology, public policy, and law* 15.2, p. 81.

Jordan, Michael I (1997). "Serial order: A parallel distributed processing approach". In: *Advances in psychology*. Vol. 121. Elsevier, pp. 471–495.

Kao, Wei-Tsung and Hung-yi Lee (Nov. 2021). "Is BERT a Cross-Disciplinary Knowledge Learner? A Surprising Finding of Pre-trained Models' Transferability". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 2195–2208. DOI: 10.18653/v1/2021.findings-emnlp.189. URL: https://aclanthology.org/2021.findings-emnlp.189.

Kaplan, Jared et al. (2020). "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361*.

Keizer, Simon, Milica Gašić, Filip Jurcicek, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young (2010). "Parameter estimation for agenda-based user simulation". In: *Proceedings of the SIGDIAL 2010 Conference*, pp. 116–123.

Kim, To Eun and Aldo Lipani (2022). "A Multi-Task Based Neural Model to Simulate Users in Goal Oriented Dialogue Systems". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 21152119. ISBN: 9781450387323. DOI: 10.1145/3477495.3531814. URL: https://doi.org/10.1145/3477495.3531814.

Kimon Kieslich, Marco Lünich and Pero Došenović (2023). "Ever Heard of Ethical AI? Investigating the Salience of Ethical AI Issues among the German Population". In: *International Journal of HumanComputer Interaction* 0.0, pp. 1–14. DOI: 10.1080/10447318.2023.2178612. eprint: https://doi.org/10.1080/10447318.2023.2178612. URL: ttps://doi.org/10.1080/10447318.2023.2178612.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980.

Kiran, B. Ravi, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Kumar Yogamani, and Patrick Pérez (2022). "Deep Reinforcement Learning for Autonomous Driving: A Survey". In: *IEEE Trans. Intell. Transp. Syst.* 23.6, pp. 4909–4926. DOI: 10.1109/TITS.2021.3054625. URL: https://doi.org/10.1109/TITS.2021.3054625.

Kober, Jens, J. Andrew Bagnell, and Jan Peters (2013). "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11, pp. 1238–1274. DOI: 10.1177/0278364913495721. URL: https://doi.org/10.1177/0278364913495721.

Kreyssig, Florian, Iñigo Casanueva, Paweł Budzianowski, and Milica Gašić (July 2018). "Neural User Simulation for Corpus-based Policy Optimisation of Spoken Dialogue Systems". In: *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. Melbourne, Australia: Association for Computational Linguistics, pp. 60–69. DOI: 10.18653/v1/W18-5007. URL: https://www.aclweb.org/anthology/W18-5007.

Kurz, Miriam, Birgit Brüggemeier, and Michael Breiter (2021). "Success is not Final; Failure is not Fatal – Task Success and User Experience in Interactions with Alexa, Google Assistant and Siri". In: *Human-Computer Interaction. Design and User Experience Case Studies*. Ed. by Masaaki Kurosu. Cham: Springer International Publishing, pp. 351–369. ISBN: 978-3-030-78468-3.

Lester, Brian, Rami Al-Rfou, and Noah Constant (Nov. 2021). "The Power of Scale for Parameter-Efficient Prompt Tuning". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 3045–3059. DOI: 10.18653/v1/2021.emnlp-main.243. URL: https://aclanthology.org/2021.emnlp-main.243.

Levin, Esther and Roberto Pieraccini (1997). "A stochastic model of computer-human interaction for learning dialogue strategies". In: *Fifth European Conference on Speech Communication and Technology*.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer (July 2020). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. URL: https://aclanthology.org/2020.acl-main.703.

Li, Xiujun, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz (Nov. 2017). "End-to-End Task-Completion Neural Dialogue Systems". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, pp. 733–743. URL: https://aclanthology.org/I17-1074.

Lin, Hsien-Chin, Shutong Feng, Christian Geishauser, Nurul Lubis, Carel van Niekerk, Michael Heck, Benjamin Ruppik, Renato Vukovic, and Milica Gašić (2023). "EmoUS: Simulating User Emotions in Task-Oriented Dialogues". In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '23. Taipei, Taiwan: Association for Computing Machinery, pp. 25262531. ISBN: 9781450394086. DOI: `10.1145/3539618.3592092`. URL: `https://doi.org/10.1145/3539618.3592092`.

Lin, Hsien-chin, Christian Geishauser, Shutong Feng, Nurul Lubis, Carel van Niekerk, Michael Heck, and Milica Gašić (Sept. 2022). "GenTUS: Simulating User Behaviour and Language in Task-oriented Dialogues with Generative Transformers". In: *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Edinburgh, UK: Association for Computational Linguistics, pp. 270–282. URL: `https://aclanthology.org/2022.sigdial-1.28`.

Lin, Hsien-chin, Nurul Lubis, Songbo Hu, Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, and Milica Gašić (July 2021). "Domain-independent User Simulation with Transformers for Task-oriented Dialogue Systems". In: *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Singapore and Online: Association for Computational Linguistics, pp. 445–456. URL: `https://aclanthology.org/2021.sigdial-1.47`.

Lipton, Zachary C, John Berkowitz, and Charles Elkan (2015). "A critical review of recurrent neural networks for sequence learning". In: *arXiv preprint arXiv:1506.00019*.

Liu, Yinhan et al. (2019). "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692*.

Lubis, Nurul, Christian Geishauser, Michael Heck, Hsien-chin Lin, Marco Moresi, Carel van Niekerk, and Milica Gašić (Dec. 2020). "LAVA: Latent Action Spaces via Variational Auto-encoding for Dialogue Policy Optimization". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, pp. 465–479. DOI: `10.18653/v1/2020.coling-main.41`. URL: `https://aclanthology.org/2020.coling-main.41`.

Ma, Xinyin, Gongfan Fang, and Xinchao Wang (2023). "LLM-Pruner: On the Structural Pruning of Large Language Models". In: *arXiv preprint arXiv:2305.11627*.

Madotto, Andrea, Chien-Sheng Wu, and Pascale Fung (July 2018). "Mem2Seq: Effectively Incorporating Knowledge Bases into End-to-End Task-Oriented Dialog Systems". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 1468–1478. DOI: `10.18653/v1/P18-1136`. URL: `https://aclanthology.org/P18-1136`.

Mao, Yanying, Fei Cai, Yupu Guo, and Honghui Chen (May 2022). "Incorporating Emotion for Response Generation in Multi-Turn Dialogues". In: *Applied Intelligence* 52.7, pp. 72187229. ISSN: 0924-669X. DOI: `10.1007/s10489-021-02819-z`. URL: `https://doi.org/10.1007/s10489-021-02819-z`.

Mishra, Kshitij, Mauajama Firdaus, and Asif Ekbal (2022). "Please be polite: Towards building a politeness adaptive dialogue system for goal-oriented conversations". In: *Neurocomputing* 494, pp. 242–254. ISSN: 0925-2312. DOI: `https://doi.org/10.1016/j.neucom.2022.04.029`. URL: `https://www.sciencedirect.com/science/article/pii/S0925231222003952`.

Niu, Jingcheng, Wenjie Lu, and Gerald Penn (2022). "Does BERT Rediscover a Classical NLP Pipeline?" In: *Proceedings of the 29th International Conference on Computational Linguistics,* *COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*. Ed. by Nicoletta Calzolari et al. International Committee on Computational Linguistics, pp. 3143–3153. URL: https://aclanthology.org/2022.coling-1.278.

OpenAI (2023). "GPT-4 Technical Report". In: *ArXiv* abs/2303.08774.

Ouyang, Long et al. (2022). *Training language models to follow instructions with human feedback*. arXiv: 2203.02155 [cs.CL].

Pan, Yan, Mingyang Ma, Bernhard Pflugfelder, and Georg Groh (Sept. 2022). "User Satisfaction Modeling with Domain Adaptation in Task-oriented Dialogue Systems". In: *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Edinburgh, UK: Association for Computational Linguistics, pp. 630–636. URL: https://aclanthology.org/2022.sigdial-1.59.

Pascanu, Razvan, Tomás Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1310–1318. URL: http://proceedings.mlr.press/v28/pascanu13.html.

Peng, Baolin, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao (2021). "Soloist: Building Task Bots at Scale with Transfer Learning and Machine Teaching". In: *Transactions of the Association for Computational Linguistics* 9, pp. 807–824. DOI: 10.1162/tacl_a_00399. URL: https://aclanthology.org/2021.tacl-1.49.

Pietquin, Olivier, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet (2011). "Sample-Efficient Batch Reinforcement Learning for Dialogue Management Optimization". In: *ACM Trans. Speech Lang. Process.* 7.3. ISSN: 1550-4875. DOI: 10.1145/1966407.1966412. URL: https://doi.org/10.1145/1966407.1966412.

Radford, Alec, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. (2018). "Improving Language Understanding by Generative Pre-Training". In: OpenAI. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Rae, Jack W. et al. (2022). *Scaling Language Models: Methods, Analysis & Insights from Training Gopher*. arXiv: 2112.11446 [cs.CL].

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *J. Mach. Learn. Res.* 21, 140:1–140:67. URL: http://jmlr.org/papers/v21/20-074.html.

Rastogi, Abhinav, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan (2020). "Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, pp. 8689–8696. DOI: 10.1609/aaai.v34i05.6394. URL: https://doi.org/10.1609/aaai.v34i05.6394.

Reif, Emily, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim (2019). "Visualizing and Measuring the Geometry of BERT". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/159c1ffe5b61b41b3c4d8f4c2150f6c4-Paper.pdf.

Scao, Teven Le et al. (2022). "Bloom: A 176b-parameter open-access multilingual language model". In: *arXiv preprint arXiv:2211.05100*.

Schatzmann, Jost, Kallirroi Georgila, and Steve Young (Sept. 2005). "Quantitative Evaluation of User Simulation Techniques for Spoken Dialogue Systems". In: *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*. Lisbon, Portugal: Special Interest Group on Discourse and Dialogue (SIGdial), pp. 45–54. URL: https://aclanthology.org/2005.sigdial-1.6.

Schatzmann, Jost, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young (Apr. 2007). "Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System". In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Rochester, New York: Association for Computational Linguistics, pp. 149–152. URL: https://www.aclweb.org/anthology/N07-2038.

Scheffler, Konrad and Steve Young (2002). "Automatic Learning of Dialogue Strategy Using Dialogue Simulation and Reinforcement Learning". In: *Proceedings of the Second International Conference on Human Language Technology Research*. HLT '02. San Diego, California: Morgan Kaufmann Publishers Inc., pp. 1219.

Schmitt, Alexander and Stefan Ultes (2015). "Interaction Quality: Assessing the quality of ongoing spoken dialog interaction by expertsAnd how it relates to user satisfaction". In: *Speech Communication* 74, pp. 12–36. ISSN: 0167-6393. DOI: https://doi.org/10.1016/j.specom.2015.06.003. URL: https://www.sciencedirect.com/science/article/pii/S0167639315000679.

Schulman, John, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel (2016). "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1506.02438.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347*.

Shi, Weiyan, Kun Qian, Xuewei Wang, and Zhou Yu (Nov. 2019). "How to Build User Simulators to Train RL-based Dialog Systems". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 1990–2000. DOI: 10.18653/v1/D19-1206. URL: https://aclanthology.org/D19-1206.

Shumailov, Ilia, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson (2023). "The Curse of Recursion: Training on Generated Data Makes Models Forget". In: *arXiv preprint arxiv:2305.17493*.

Silver, David et al. (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nat.* 529.7587, pp. 484–489. DOI: 10.1038/nature16961. URL: https://doi.org/10.1038/nature16961.

Solaiman, Irene et al. (2019). "Release strategies and the social impacts of language models". In: *arXiv preprint arXiv:1908.09203*.

Soltan, Saleh et al. (2022). "Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model". In: *arXiv preprint arXiv:2208.01448*.

Song, Xiaohui, Liangjun Zang, Rong Zhang, Songlin Hu, and Longtao Huang (2022). "Emotionflow: Capture the Dialogue Level Emotion Transitions". In: *ICASSP 2022 - 2022 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8542–8546. DOI: 10.1109/ICASSP43922.2022.9746464.

Song, Zhenqiao, Xiaoqing Zheng, Lu Liu, Mu Xu, and Xuanjing Huang (July 2019). "Generating Responses with a Specific Emotion in Dialog". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 3685–3695. DOI: 10.18653/v1/P19-1359. URL: https://aclanthology.org/P19-1359.

Sun, Weiwei, Shuo Zhang, Krisztian Balog, Zhaochun Ren, Pengjie Ren, Zhumin Chen, and Maarten de Rijke (2021). "Simulating User Satisfaction for the Evaluation of Task-Oriented Dialogue Systems". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, pp. 24992506. ISBN: 9781450380379. DOI: 10.1145/3404835.3463241. URL: https://doi.org/10.1145/3404835.3463241.

Thomson, Blaise and Steve Young (2010). "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems". In: *Computer Speech & Language* 24.4, pp. 562–588. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2009.07.003. URL: https://www.sciencedirect.com/science/article/pii/S0885230809000497.

Thoppilan, Romal et al. (2022). "Lamda: Language models for dialog applications". In: *arXiv preprint arXiv:2201.08239*.

Touvron, Hugo et al. (2023). "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971*.

Tseng, Bo-Hsiang, Yinpei Dai, Florian Kreyssig, and Bill Byrne (Aug. 2021). "Transferable Dialogue Systems and User Simulators". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, pp. 152–166. DOI: 10.18653/v1/2021.acl-long.13. URL: https://aclanthology.org/2021.acl-long.13.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vries, Wietse de, Andreas van Cranenburgh, and Malvina Nissim (Nov. 2020). "What's so special about BERT's layers? A closer look at the NLP pipeline in monolingual and multilingual models". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 4339–4350. DOI: 10.18653/v1/2020.findings-emnlp.389. URL: https://aclanthology.org/2020.findings-emnlp.389.

Wang, Ben and Aran Komatsuzaki (May 2021). *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*. https://github.com/kingoflolz/mesh-transformer-jax.

Wen, Tsung-Hsien, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young (Apr. 2017). "A Network-based End-to-End Trainable Task-oriented Dialogue System". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 438–449. URL: https://aclanthology.org/E17-1042.

Weninger, Felix, Hakan Erdogan, Shinji Watanabe, Emmanuel Vincent, Jonathan Le Roux, John R. Hershey, and Björn Schuller (2015). "Speech Enhancement with LSTM Recurrent Neural Networks and its Application to Noise-Robust ASR". In: *Latent Variable Analysis and Signal Separation*. Ed. by Emmanuel Vincent, Arie Yeredor, Zbyněk Koldovský, and Petr Tichavský. Cham: Springer International Publishing, pp. 91–99. ISBN: 978-3-319-22482-4.

Williams, Jason D. and Steve Young (2007). "Partially observable Markov decision processes for spoken dialog systems". In: *Computer Speech & Language* 21.2, pp. 393–422. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2006.06.008. URL: https://www.sciencedirect.com/science/article/pii/S0885230806000283.

Yoo, Kang Min, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park (Nov. 2021). "GPT3Mix: Leveraging Large-scale Language Models for Text Augmentation". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 2225–2239. DOI: 10.18653/v1/2021.findings-emnlp.192. URL: https://aclanthology.org/2021.findings-emnlp.192.

Young, Steve (2002). "Talking to machines (statistically speaking)". In: *Proc. 7th International Conference on Spoken Language Processing (ICSLP 2002)*, pp. 9–16. DOI: 10.21437/ICSLP.2002-2.

Young, Steve, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu (2010). "The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management". In: *Computer Speech & Language* 24.2, pp. 150–174. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2009.04.001. URL: https://www.sciencedirect.com/science/article/pii/S0885230809000230.