Joint Learning of Practical Dialogue Systems and User Simulators

Alistair John McLeay

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Hughes Hall College | August 2022

Dedicated to my partner Gabi, my loving parents John and Sarah, my brother Reid, and my friend and mentor LJ. Without the support of any one of you my time in Cambridge would not have been possible.

Declaration

I, Alistair McLeay of Hughes Hall College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

This report makes use of the following software:

- The code and model published with the AAAI 2021 paper "UBAR: Towards Fully End-to-End Task-Oriented Dialog System with GPT-2" which was modified to create UBAR-E and related models discussed throughout this document.
- The gpt2-user-model codebase developed by Andy Tseng was modified to develop my User Simulator model: LSP-US, that is discussed throughout this report. Andy Tseng was a PhD student at Cambridge University supervised by Professor Bill Byrne who graduated in 2022.
- The Transformer Reinforcement Learning Library for the PPO Transformer Reinforcement Learning regime discussed throughout this document.
- The official MultiWOZ 2.0 evaluation scripts were used to produce the results presented and discussed in Chapter 5.
- The PyTorch Library for training and testing all models discussed throughout this document.

Word count: 14,788

Alistair John McLeay
August 2022

Acknowledgements

I would like to thank my supervisor, Professor Bill Byrne, for his invaluable support and guidance throughout this project. It was a pleasure working with you, and your guidance made this experience an intellectually challenging and rewarding one. I would also like to thank my course mates for their support and companionship throughout this year, particularly the Bru's. It is largely thanks to them that my time at Cambridge has been so enjoyable.

Abstract

A key difficulty in training task-oriented dialogue (TOD) systems is a lack of training data. We explore the possibility of creating new dialogue data through the interaction of an end-to-end (E2E) Transformer-based TOD system, and an E2E Transformer-based User Simulator. We refer to the generation of new data through conversations between the two systems as *self-play*. Our goal is to develop such a TOD system and User Simulator using supervised learning that a) can operate on unseen dialogues, and b) are good enough to drive self-play experiments with the goal of refining their performance. We develop UBAR-E, an E2E TOD system that extends an influential model; UBAR, enabling it to work on unseen dialogues by using *Inferred Turn Domains* as opposed to ground-truth turn domains. We show this approach produces similar performance to UBAR in both model Score and a range of lexical richness metrics on the MultiWOZ dataset. We also show that UBAR-E sets a very high baseline performance. Along with UBAR-E, we develop LSP-US (Large-Scale Pre-trained User Simulator); the first E2E Transformer-based User Simulator. LSP-US uses a novel design and is able to drive conversations based on any goal that follows the MultiWOZ schema. LSP-US demonstrates comparable or greater lexical richness than UBAR-E. You can interact with UBAR-E and LSP-US using our web application at https://huggingface.co/spaces/alistairmcleay/cambridge-masters-project. We demonstrate that our two E2E Transformer-based models provide a strong foundation for future self-play experiments, achieving our primary research goal. Further to this, we perform a number of experiments using synthetic data generated from self-play between the two systems. We analyse the MultiWOZ evaluation metrics and propose a number of improvements. We develop a Reinforcement Learning (RL) regime using Proximal Policy Optimisation, PPO Transformer RL, to refine our two models, and show promising preliminary results. Finally, we suggest a framework to leverage the two models we have developed for joint optimisation in future research.

Table of contents

List of figures

List of tables

Acronyms

ABUS Agenda-based User Simulator. 12

BLEU Bilingual Evaluation Understudy Score. 41, 46

BSRM Belief State Reward Module. 35

CBE Conditional Bigram Entropy. 42, 58

DB Database. 9

DP Dialogue Policy. 9, 23, 58, 61

DST Dialogue State Tracker. 9, 23, 58, 61

E2E End-to-End. 2, 6, 21, 42, 46, 60

GPT-2 Generative Pre-trained Transformer 2. 2, 6, 11, 21, 43

ITD Inferred Turn Domain. 46

JOUST Joint Optimisation with a User SimulaTor. 3, 13

LLM Large Language Model. 2, 6, 21, 61

LSM Lexicalisation State Manager. 25

LSP-US Large-Scale Pre-trained User Simulator. 4, 14, 21, 39, 46, 60

LSTM Long Short Term Memory. 3, 13, 61

NLG Natural Language Generation. 9, 23, 58, 61

NLU Natural Language Understanding. 9, 23, 58, 61

NUS Neural User Simulator. 13

PCM Pre-trained Conversation Model. 11

PPO Proximal Policy Optimization. 14, 21, 39, 46, 61

RL Reinforcement Learning. 3, 6, 21, 39, 46, 60

RNN Recurrent Neural Network. 3, 12

SARRM System Act & Response Reward Module. 35

TOD Task-Oriented Dialogue. 1, 6, 21, 60

TRPO Trust Region Policy Optimisation. 15

TUS Transformer-based User Simulator. 13, 14

UBAR User Utterance, Belief state, system Act, system Response. 4, 11, 21, 40, 46, 60

UBAR-E UBAR-Extended. 4, 21, 39, 46, 60

US User Simulator. 3, 6

Chapter 1

Introduction

1.1 Motivation

Language is a fundamentally important part of human intelligence, and is at the heart of many of the most important economic and social activities we undertake as a species. It therefore follows that creating highly capable AI systems that can effectively model language is profoundly valuable, especially when it comes to automating processes in society. Language modelling has seen unprecedented progress over the last five years following the introduction of the Transformer architecture, introduced in the now famous *Attention is All You Need* paper by Vaswani et al. (2017).

Dialogue systems are a key area within language modelling. Language, after all, is mostly used to communicate in real-time between more than one person (or entity). There are two main research disciplines within dialogue systems; *Chat-oriented* systems and *Task-oriented* dialogue (TOD) systems. Chat-oriented systems are designed to interact with users without a specific goal or task driving their behaviour. They aim to interact in a general and fluent manner that imitates effective human "chat" (Li et al., 2016; Serban et al., 2015; Sordoni et al., 2015). On the other hand, TOD systems have a set predefined goal they use to steer the conversation. They aim to facilitate solving well-defined tasks that a user wishes to complete (Levin et al., 2000; Young et al., 2010). Example tasks could include searching for a restaurant and making a reservation (Henderson et al., 2019), or providing information about a museum exhibit (Misu and Kawahara, 2007). Figure 1.1 demonstrates an example of a TOD system interacting with a user who is trying to book a restaurant.

Over the last two years, researchers have started applying Transformers to TOD systems due to their powerful ability to learn to model complex language tasks, even when those tasks contain complex and rigid syntax (He et al., 2021; Lee, 2021; Peng et al., 2020; Su et al., 2021; Yang et al., 2020). These Transformer-based models generally use large language models

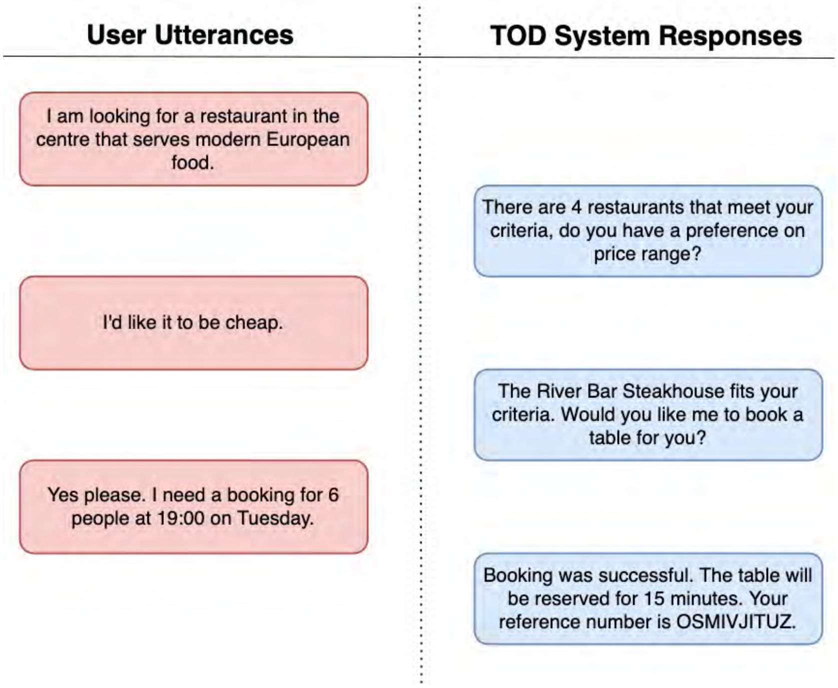


Fig. 1.1 Example dialogue between a user and a TOD system where the user has a simple one-task goal of booking a restaurant, and the TOD system needs to a) find an entity that matches the constraints outlined by the user, and b) book that entity for the user.

(LLMs) such as GPT-2 (Generative Pre-trained Transformer 2) published by Radford et al. (2019) as their "backbone". These LLMs are trained on very large datasets using unsupervised learning. The Transformer-based TOD systems take these LLMs and fine-tune them on specific TOD data, such as the MultiWOZ dataset (Budzianowski et al., 2018). This approach has quickly led to very good performance on end-to-end (E2E) TOD system benchmarks. End-to-end refers to the fact that the model operates at the natural language level, i.e., it consumes user utterances as natural language and produces system responses as natural language. This is in contrast to models that operate at the *semantic* level. In this thesis we focus on the MultiWOZ dataset. MultiWOZ is one of the most commonly used datasets to train and analyse the performance of E2E TOD systems. It contains over 10,000 complex, multi-domain dialogues.

While Transformer-based models have advanced the state-of-the-art in E2E TOD systems, their performance is still far from perfect, as shown in the MultiWOZ official results for E2E models (Budzianowski et al., 2022). One bottleneck these models face currently is a lack of high-quality data to fine-tune them. Moreover, this is not limited to Transformer-based TOD systems—it has been a challenge in the research field of TOD systems for a long time (Budzianowski and Vulić, 2019). Collecting high-quality TOD data is costly due to the time involved with correctly labelling each full dialogue, and also due to the syntactic complexity of the annotations needed for intermediary information such as belief states[1], as outlined by (Budzianowski et al., 2018).

A common approach to address the shortfall of sufficient high-quality TOD data is to introduce a User Simulator (US). A US interacts with a TOD system just as a human user would, attempting to achieve a goal made up of one or more tasks. Reinforcement Learning (RL) is then used to simulate new interactions between the two models (the TOD system and the US) to further improve the TOD system (Casanueva et al., 2018; Peng et al., 2017; Young et al., 2013), or in some cases, to improve both systems jointly (Liu and Lane, 2017; Papangelis et al., 2019; Takanobu et al., 2020). However, while there have been promising results from applying RL to jointly improve these models, no work had applied the approach to *multi-domain* E2E joint modelling until the Joint Optimisation with a User SimulaTor (JOUST) framework was published last year by Tseng et al. (2021). This approach showed very promising results, substantially improving the performance of the TOD system by jointly optimising both models in a multi-domain E2E context. Both the TOD system and User Simulator that JOUST uses are component-based pipelined TOD systems[2], built using LSTM RNNs.

While the results published by Tseng et al. (2021) were very promising, they have quickly been surpassed by Transformer-based approaches. However, none of the Transformer-based models leverage joint optimization of a US and TOD system to boost performance. Further, at the time of writing there had only been one Transformer-based User Simulator published, and it operates at the semantic level (Lin et al., 2021). An E2E Transformer-based User Simulator that operates at the natural language level could present significant value as it would allow for the development of a new version of JOUST, where the TOD system and US are built using the significantly more powerful Transformer models. Figure 1.2 shows a high-level representation of an E2E TOD system and US, each built by fine-tuning GPT-2, interacting with each other to generate unseen dialogues: a phenomenon we refer to as *self-play*.

1.2 Primary Research Goal

In this thesis we primarily focus on the following research question: Can we build a fully E2E Transformer-based TOD system *and* US, that are good enough to be used to drive joint optimisation techniques such as those outlined in Tseng et al. (2021)?

To be clear, our research goal is not to jointly optimise the two models, but rather to design and build the two E2E Transformer-based systems that can operate on new, unseen MultiWOZ

[1] Belief states, and other intermediary states used in TOD systems, are explained in Section 2.2.
[2] Pipelined TOD systems are explained in Section 2.2.

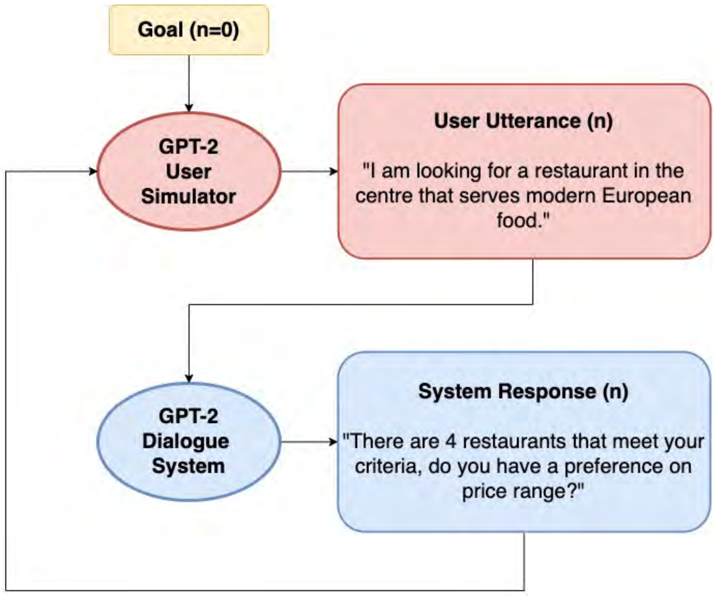


Fig. 1.2 High-level representation of self-play between an E2E TOD system and an E2E US.

dialogues. If these systems can be highly performant after individual supervised learning then they will provide a very strong foundation for future research to explore joint optimisation.

1.3 Contributions

We make two primary contributions in this thesis:

1. We present UBAR-E, an extension of UBAR that can interact with new, unseen dialogues, enabling it to be used with a User Simulator for joint optimisation. We show that UBAR-E sets a very high performance baseline for subsequent work to improve.

2. We present the design and analysis of the first E2E Transformer-based US[3]: Large-Scale Pre-trained User Simulator (LSP-US), using a novel design. LSP-US was originally designed and implemented by Andy Tseng, a recent PhD graduate supervised by Professor Bill Byrne in the Machine Intelligence Group at Cambridge University. We extend Andy's implementation and present it here. LSP-US is able to interact with UBAR-E on unseen dialogues and demonstrates comparable or greater lexical richness than UBAR-E.

With the development of these systems, we establish a strong foundation for future joint optimisation experiments. You can interact with UBAR-E and LSP-US using our web application at https://huggingface.co/spaces/alistairmcleay/cambridge-masters-project. The web

[3]We are not aware of any other E2E Transformer-based User Simulators that have been published.

application also allows you to observe the two systems interacting with each other on unseen dialogues.

Further, we make three secondary contributions in this thesis:

1. We present the results of a number of experiments using synthetic data generated from self-play between UBAR-E and LSP-US.

2. We analyse the MultiWOZ evaluation metrics and propose a number of significant improvements.

3. We develop a Reinforcement Learning regime using Proximal Policy Optimisation, PPO Transformer RL, to refine our two models, and show promising preliminary results (Ziegler et al., 2019).

Finally, in Chapter 6 we suggest a framework to leverage UBAR-E and LSP-US for joint optimisation in future research.

1.4 Thesis Outline

This thesis is organised into 6 chapters. After the Introduction, Chapter 2 provides an overview of the relevant background and puts this research in the context of previous research. Next, in Chapter 3 we present the design of our E2E Transformer-based TOD system, UBAR-E, and our E2E Transformer-based User Simulator, LSP-US. We also present the preliminary design of a proposed RL regime for fine-tuning UBAR-E: PPO Transformer RL. In Chapter 4 we describe the experimental setup for training and evaluating our models. Chapter 5 contains our results and accompanying discussions from assessing the performance and lexical richness of UBAR-E and related models. We show that UBAR-E provides a very high baseline performance for subsequent work to improve. We also show that LSP-US achieves comparable or improved lexical richness to UBAR-E. We suggest several improvements to the official evaluation criteria used for the MultiWOZ dataset, and we present an analysis of our preliminary results applying PPO Transformer RL to refine UBAR-E. Conclusions are drawn and recommendations for future work are outlined in Chapter 6.

Chapter 2

Background

The opening chapter introduced the rapidly-growing role LLMs have played in the research field of dialogue systems. We described TOD systems and the limitations they face due to a lack of annotated data, as well as the use of USs and RL to address this problem. We covered the four key contributions this thesis makes, and provided an outline of the document. In this chapter we review the relevant literature and cover the high-level technical background relevant to this thesis. We first review LLMs (Section 2.1). We then review E2E TOD systems (Section 2.2) and User Simulators (Section 2.3), both of which are key to this work. We then proceed to briefly review RL for TOD systems (Section 2.4), and finish by outlining the MultiWOZ dataset (Budzianowski et al., 2018) (Section 2.5).

2.1 Large Language Models

Since Vaswani et al. (2017) published the now famous *Attention is All You Need* paper, Transformer-based LLMs have become very influential due to their ability to scale effectively and handle long temporal contexts. These models are generally trained on very large corpora using unsupervised learning (Brown et al., 2020; Devlin et al., 2019; Radford et al., 2019). While incredibly powerful on their own, they usually need to be fine-tuned on domain-specific labelled data to gain strong performance in a particular domain. This is especially true for "smaller" models like GPT-2[1], which we focus on in this thesis.

[1]The full sized GPT-2 model still has 1.5 billion parameters. However, this is "small" compared to other more recent models like GPT-3 with 175 billion parameters (Brown et al., 2020).

2.1.1 Transformers

The research literature on Transformers is abundant (Dai et al., 2019; Devlin et al., 2019; Raffel et al., 2019), so we keep this section very high-level. The Transformer architecture is shown in Figure 2.1.

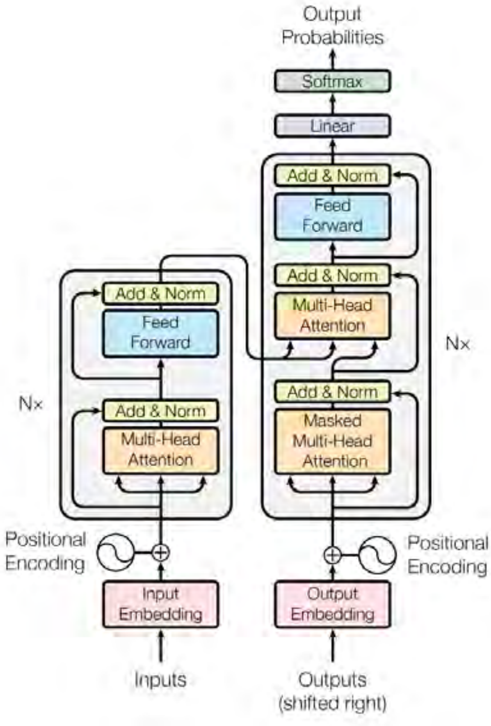


Fig. 2.1 Architecture of the original Transformer from Vaswani et al. (2017). The left side of the diagram contains the *encoder* and the right side contains the *decoder*.

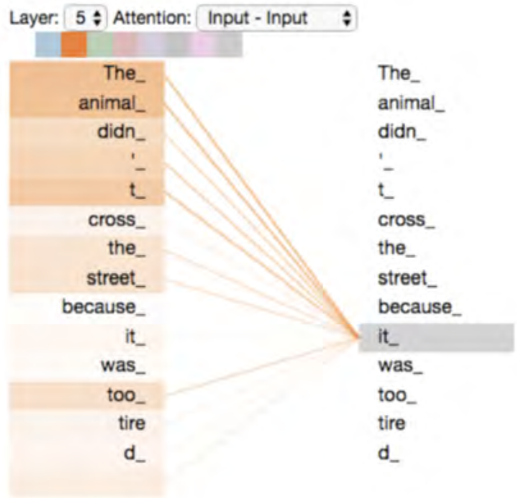


Fig. 2.2 Illustration of the Attention mechanism in practice from Alammar (2018) where the model is encoding "it". The darker the orange the more the model is "attending" to those words; in this case the words being associated most strongly with "it" are "The animal". The tool used for this visualisation can be found at this notebook.

The Transformer takes in a sequence of tokens (usually words) and auto-regressively generates new tokens. At the heart of the Transformer are multiple Attention[2] blocks that allow the model to selectively focus on the inputs it predicts to be most relevant. Figure 2.2 contains a visual representation of the Attention mechanism in action. This structure can be used for a very wide range of language modelling tasks, from neural machine translation, to text summarization and dialogue systems.

Liu et al. (2018) showed that you can perform very effective language modelling without the encoder in the original Transformer architecture. They made a change to the decoder blocks, removing the second Multi-Head Attention layer, but otherwise the architecture was

[2]The Attention mechanism was first introduced by Bahdanau et al. (2015).

very similar. Figure 2.3 shows the Transformer-*Decoder*; a modified version of the right half of Figure 2.1.

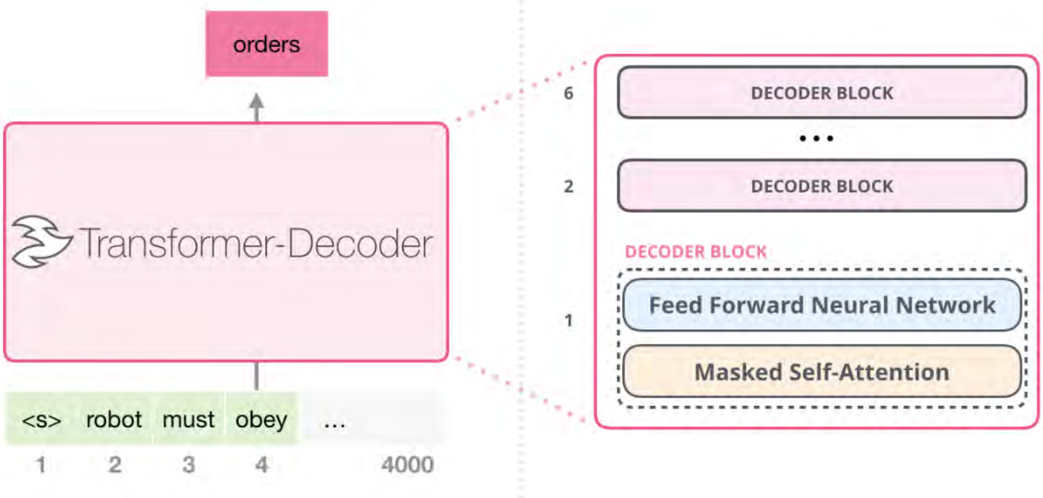


Fig. 2.3 Illustration of the Transformer-Decoder, used by GPT-2, from Alammar (2019). Each of the six decoder blocks on the right are identical (one is expanded). This version of the Transformer-Decoder was originally proposed by Liu et al. (2018).

2.1.2 GPT-2

In this thesis we fine-tune GPT-2 (Radford et al., 2019) to build an E2E Transformer-based TOD system, and an E2E US. GPT-2 is a *decoder-only* model, meaning it follows the structure shown in Figure 2.3. It is autoregressive: it generates one token at a time, taking each new taken and appending it to the input sequence. This leads to a causal *uni-directional attention* scheme where the model attends to all previously generated tokens as it generates the next token. GPT-2 has proven to not only be a very powerful LLM for many general language modelling tasks[3] (Budzianowski and Vulić, 2019; Ham et al., 2020; Hosseini-Asl et al., 2020) but also for TOD systems specifically. In particular, the ability of GPT-2 to implicitly learn fairly complex syntactic structure with limited fine-tuning is what makes it so powerful for building E2E TOD systems where data scarcity is a challenge (Yang et al., 2020).

[3]GPT-3 (Brown et al., 2020), the next generation of GPT-2, is over one order of magnitude larger in terms of number of parameters, and significantly outperforms it. It is, however, only accessible via. API, and open-source models of similar size are too large for us to practically utilise with the computational resources available to us.

2.2 E2E Task-Oriented Dialogue Systems

In the previous section we covered LLMs—specifically looking at the Transformer architecture (Section 2.1.1) and GPT-2—a *decoder-only* Transformer model (Section 2.1.2). In this section we cover E2E TOD systems, starting with the pipelined approach (Section 2.2.1) and moving on to the recently developed Transformer-based approach (Section 2.2.2).

2.2.1 Pre-LLM TOD System Design

Previously, E2E TOD systems have been built by combining a number of separate modules, each of which is responsible for a different major function within the overall system (Chen et al., 2017). The ability for LLMs to model rich and syntactically complex language interactions has caused a recent shift toward single-model E2E TOD systems, but this is a relatively recent phenomenon. We can describe pre-Transformer E2E TOD systems in terms of a *pipeline* of four individual modules and a database that are connected together, as shown in Figure 2.4. The function of each of the five components is as follows:

1. **Natural Language Understanding (NLU):** the semantic context is extracted from the most recent user utterance and is represented as a system act.

2. **Dialogue State Tracker (DST):** the system act is used to update the current dialogue state, which tracks the information the user has provided across turns.

3. **Database (DB):** the relevant DB is queried (e.g. the "restaurant" DB) based on the constraints present in the output of the DST. The relevant entity or entities are returned.

4. **Dialogue Policy (DP):** the DST output and the DB response are used to generate a semantic representation of the system response, effectively capturing the intent of the system in this turn.

5. **Natural Language Generation (NLG):** the output of the DP module is used to generate a natural language representation to pass to the user.

There is a rich literature on TOD systems that focus on specific parts of a pipelined TOD system, whether that be the NLU step Mehri et al. (2020), the DST step Balaraman et al. (2021), the DP step (Kwan et al., 2022), or the NLG step Libo et al. (2022). However, due to the complexity that comes from the pipelined approach to TOD systems, where four separate models are needed to create an E2E system, there has been significant research into simplifying the process. This includes word-level DST modules which combine the NLU and DST modules

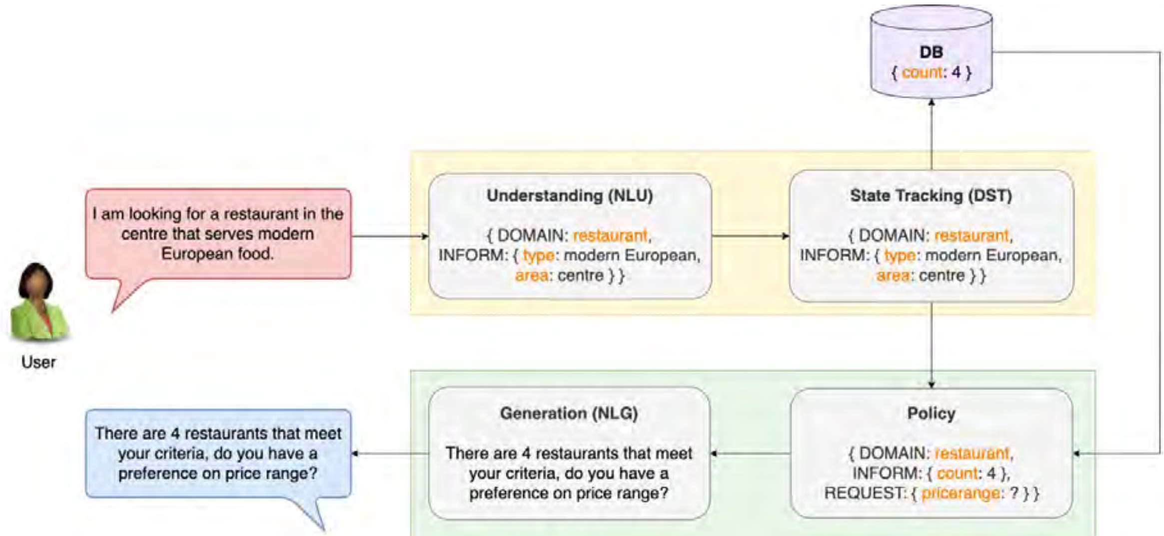


Fig. 2.4 Illustration of the pipelined approach to E2E TOD systems made up of 4 connected modules and a DB. In this case the first turn of a dialogue is shown where the user wants to book a restaurant and the system has found 4 entities in the database that match the user's constraints. The yellow rectangle represents word-level DST modules which combine the NLU and DST modules, and the green rectangle represents word-level policy modules which do the same for NLG and Policy modules.

into one (Mrksic et al., 2016; Wu et al., 2019) (as represented by the yellow rectangle in Figure 2.4), and word-level policy modules, which similarly combine the DP and NLG modules (Chen et al., 2019; Zhao et al., 2019) (as represented by the green rectangle in Figure 2.4).

An important part of the NLG process that is not included in the high-level overview in Figure 2.4 is *lexicalisation*. Lexicalisation refers to filling general *slots* in a *delexicalised* generated system response with specific values from the database, as shown in Figure 2.5. Delexicalisation is used to enable the model can learn the *general* structure of outputs, rather than dealing with the very large set of specific values contained in the database. We present a more comprehensive example of lexicalisation in Section 3.1 where we detail how UBAR-E works.

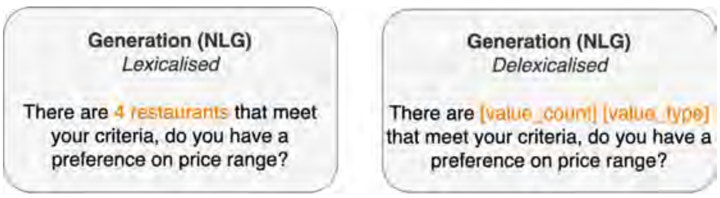


Fig. 2.5 Example of lexicalisation.

Over the last five years we have seen a movement away from pipelined, modular approaches, toward E2E (sequence-to-sequence) TOD systems take in a user utterance in natural language, and return a system response in natural language. The movement of the research field toward E2E approaches began with Lei et al. (2018) who published Sequicity; a two-stage CopyNet (Gu et al., 2016) that jointly generates belief spans and system responses via. a single sequence-to-sequence model. Many E2E approaches were introduced thereafter (Eric et al., 2017; Liang et al., 2019; Wen et al., 2017; Zhang et al., 2020a). This shift toward E2E models led to very strong TOD systems being developed using *LLM*s, where a single Transformer model is used.

2.2.2 Transformer-based TOD Systems

The pace at which Transformers have taken over as the go-to approach in E2E TOD systems has been so rapid that it is a challenge to stay abreast of the latest developments. Of the four highest ranking models on the official E2E MultiWOZ benchmark[4] (He et al., 2021; Lee, 2021; Su et al., 2021; Sun et al., 2022), all of them were published in the last *12 months*. Budzianowski and Vulić (2019) were the first to apply the Transformer architecture to the TOD system domain, leveraging GPT-2 to produce system responses conditioned on the dialogue context. Lin et al. (2020) and Yang et al. (2020) then introduced models which took things a step further, releasing the first E2E Transformer-based TOD systems—meaning they consumed user utterances in natural language, and produced system responses in natural language. Lin et al. (2020) released Minimalist Transfer Learning, MinTL, using fine-tuned versions of T5 (Raffel et al., 2019) and BART (Lewis et al., 2019), while Yang et al. (2020) released UBAR, using a fine-tuned version of GPT-2. Both these models effectively use a single Transformer model to generate all four components of a pipelined TOD system NLU, DST, DP, and NLG.

Pre-trained Conversation Models

Over the last two years there has been a shift in the research field toward utilising Pre-trained Conversation Models (PCMs). These models are variants of LLMs (discussed in Section 2.1), where they have been specifically adapted for conversational modelling. This includes DialoGPT that was trained on 147M conversation-like exchanges from Reddit (Zhang et al., 2020b), Blender, that was trained on 1.5B conversations also from Reddit, (Roller et al., 2020), and Meena, that was trained on a conversational dataset of 40B words from public domain social media conversations (Adiwardana et al., 2020). For reference, GPT-2 (Radford et al., 2019) was trained on around 10% of the amount of data that Meena was trained on.

[4]The E2E MultiWOZ benchmark (Budzianowski et al., 2022) is discussed in detail in Section 4.2.

2.3 User Simulators

In the previous section we covered E2E TOD systems, from the *pipelined* modular approach (Section 2.2.1) through to the recent and rapidly progressing approach of fine-tuning LLMs (Section 2.2.2). In the next section we cover User Simulators. Similar to the above section, we start with pre-Transformer approaches (Section 2.3.1) and move on to Transformer-based models (Section 2.3.2).

Conceptually, an E2E US can be thought of in terms of a three-step process[5]:

1. **Context Analysis:** the most recent user utterance is analysed along with the dialogue context, which is used to generate a state that represents the intentions of the TOD system with its latest response: *the system act*.

2. **Response Planning:** the generated system act is analysed along with the dialogue context to determine the task status, whether a goal change is warranted, and to update the goal state if it is. This analysis then leads to the generation of the user semantic response, based on the entire context it currently has: *the user act*.

3. **Response Generation:** Finally, the outcome of the Context Analysis and Response Planning steps are analysed and used to generate a natural-language response to communicate the relevant information to the system: *the user utterance*.

2.3.1 Pre-LLM User Simulator Design

User Simulators are generally made up of multiple separate components, just as TOD systems were before Transformer-based models become common. There have been a wide range of approaches to building user simulators, including lattice-based USs (Scheffler and Young, 2001), Markov model based USs (Eckert et al., 1997; Pietquin and Dutoit, 2006), and inverse reinforcement learning based USs that model user behaviour as a Markov decision process (Chandramohan et al., 2012).

Another common approach, the Agenda Based User Simulator (ABUS) (Schatzmann et al., 2007), is built on hand-crafted rules that are based on an "agenda". ABUS effectively models the user state as a stack, ordered according to the priority of user actions. Despite being very effective for small, simple domains, the ABUS approach is limited by its requirement for manually defined rules, which makes it intractable for complex dialogues. More recently, data-driven component-based neural user simulators became common. These models leverage Recurrent Neural Networks (RNNs) to model different components of the US.

[5]Thanks to Alexandru Coca for inspiring this description with his thesis "On Evaluating User Models for Task-Oriented Dialogues".

To build a data-driven model, the sequence-to-sequence (Seq2Seq) model structure is widely used. Asri et al. (2016) were the first to propose a Seq2Seq encoder-decoder US, which leverages Long Short Term Memory (LSTM) RNNs and outputs sequences of dialogue acts. Crook and Marin (2017) were the first to propose an *E2E* US (that takes in a system utterance in natural language, and outputs a user utterance in natural language), though their approach was not goal-driven. This means the model cannot be used to train a policy because the US cannot be controlled in regard to its "intentions". Kreyssig et al. (2018) published the pivotal Neural User Simulator (NUS) which is also a sequence-to-sequence model, but it outputs semantic utterances rather than natural language.

Over the last five years many other data-driven user simulators have been published (Gür et al., 2018; Hou et al., 2019; Nie et al., 2019). However, they are all domain-dependent, meaning they need to be retrained if you were to apply them to a new domain, significantly limiting their utility when it comes to real world application. Moreover, the vast majority of USs discussed thus far operate at the level of semantic representations. They can effectively capture user intent, but this is insufficient if one wishes to use them with a fully E2E TOD system that consumes and produces natural language.

One exception is the Joint Optimisation with a User Simulator (JOUST) framework published by Tseng et al. (2021) last year, which operates on natural language *and* is domain-independent. However, JOUST is built with a component-based system (leveraging a number of LSTMs), and its performance is superseded by Transformer-based models[6] (Budzianowski et al., 2022).

2.3.2 Transformer-based User Simulators

Compared to TOD systems, the literature on Transformer-based User Simulators is very limited. Last year Lin et al. (2021) published the first (and as far as we are aware, only) Transformer-based User Simulator for TOD systems: TUS. TUS is domain-independent, addressing a significant limitation that most data-driven USs had up to that point. However, critically, TUS operates at the *semantic level*; it takes in the system action from the TOD system and returns the user act. This means the US relies on template-based NLG to convert the semantic-level actions into user utterances.

Figure 2.6 shows that ABUS and graph-based models are domain dependent and require significant hand-crafting (they aren't data-driven). On the other hand, data-driven models such as NUS, VHUS (Gür et al., 2018), and the original Seq2Seq model from Asri et al. (2016) are able to leverage data to learn, but are limited to one domain.

[6]Here we are referencing the performance of the JOUST framework (TOD system included) on the MultiWOZ dataset, as it is a joint optimisation framework, rather than a stand-alone US.

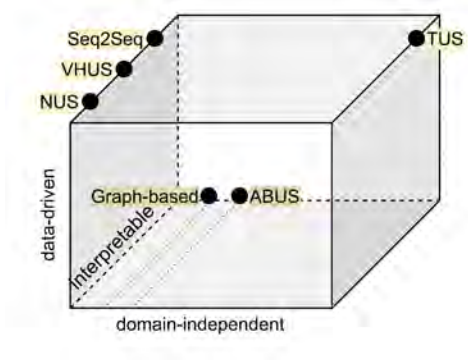


Fig. 2.6 Comparison between TUS and other types of User Simulators. The extent to which a model is data-driven, domain-independent, and interpretable are used. We see that TUS performs strongly on all three axes, whereas other approaches have at least one domain they do not score highly on. Image from Lin et al. (2021).

While TUS represents a novel and valuable contribution as the first Transformer-based US model for training TOD systems, its semantic-level operation is limiting. To fully leverage the power of the Transformer architecture, a logical next step is to train an E2E US that is able to operate at the natural language-level, enabling it to interact directly with a Transformer-based TOD system. Importantly, we are not aware of any *E2E* Transformer-based US models for training TOD systems that have been developed—something we address with our novel US: LSP-US.

2.4 Reinforcement Learning for TOD Systems

In the previous section we covered User Simulators, starting with pre-Transformer approaches (Section 2.3.1) and moving on to Transformer-based models (Section 2.3.2). In this section we cover RL for TOD systems. We first present a brief overview of the field (Section 2.4.1). We then provide an overview of Proximal Policy Optimisation (PPO) (Section 2.4.2) and PPO for Transformer (Section 2.4.3). Finally, we outline the JOUST framework (Section 2.4.4). We keep this section fairly high-level, as RL for TOD systems is not the primary focus of this thesis.

2.4.1 Overview

RL (Sutton et al., 1999) is a useful paradigm in dialogue *policy* learning as it formulates the task as a long-term sequential decision-making process. The reward used is usually associated with dialogue success, i.e., “Did the TOD system provide the user with appropriate entities?”.

One reason RL can be a powerful paradigm for training dialogue policy is that the reward signal can be very flexible, and can be used to model things like user satisfaction (Schmitt and Ultes, 2015). RL can be applied to optimising TOD systems via. both online RL (Liu et al., 2017) and offline RL (Jang et al., 2022). In online RL new dialogues are generated with either a human user or a US. In offline RL no new data is generated and the agent no longer has the ability to interact with the environment, i.e., it cannot generate new dialogues. Offline RL is closer to supervised learning, and is generally a lot simpler to implement, though online RL has the benefit of being able to learn as it generates new dialogues.

Online RL can address the data scarcity issue that supervised learning faces through the generation of additional data with a US. However, the vast majority of USs developed to-date operate at the semantic level (they do not consume and produce natural language) (Asri et al., 2016; Casanueva et al., 2018). This has limited the potential that online RL systems can have in optimising TOD systems, as it restricts them to dialogue *policy* learning, rather than E2E learning.

Beyond applying RL to train a dialogue policy from scratch, in recent years it has also been applied to jointly train a TOD system and US. This approach can be powerful as the systems are able to first be trained with supervised learning and then refined over further interaction, or *self-play*, between the two agents. Prior work has shown benefits from applying this approach to dialogue policy learning at the semantic level, improving dialogue success rate (Casanueva et al., 2018; Sordoni et al., 2015; Young et al., 2013). Tseng et al. (2021) introduced the first multi-domain E2E joint optimisation framework, showing impressive results and providing a strong case for using RL to refine TOD systems jointly with a US (covered in Section 2.4.4 below).

2.4.2 Proximal Policy Optimisation

A highly influential approach to refining deep neural networks in recent years is the online RL algorithm Proximal Policy Optimization (PPO) (Schulman et al., 2017). PPO was developed as an improved version of Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) that is easier to implement and tune, and has improved sample efficiency. In PPO, the Policy Gradient Loss is the expectation over the log of the policy actions, multiplied by an estimate of the Advantage function:

$$L^{PG}(\theta) = \hat{E}_t \left[\log \pi_\theta (a_t | s_t) \hat{A}_t \right]$$

where $\log \pi_\theta (a_t | s_t)$ represents the log probability from the output of the policy network, and $\hat{A}_t$, the Advantage function, is an estimate of the relative value of the selected action. The

Advantage function is calculated as the discounted reward minus a value function, where the value function provides a (noisy) estimate of the discounted sum of future rewards.

While it would be straight-forward to perform multiple gradient descent steps on L^{PG} using the same trajectory, this empirically leads to destructively large policy updates (Schulman et al., 2017). PPO deals with this by adding a KL penalty to limit the policy from moving too far from the original policy when running gradient descent:

$$L^{PG}(\theta) = \hat{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t - \beta \text{KL}\left[\pi_{\theta_{\text{old}}}(a_t|s_t), \pi_\theta(a_t|s_t)\right]\right]$$

where $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the ratio between the action under the current policy and the action under the previous policy, and β is a hyperparameter that controls the weighting of the KL penalty.

While the KL penalty does limit destructively large gradient updates, it adds complexity to the optimisation process, which can lead to unstable training behaviour. OpenAI implements a solution to this instability called Clipped Surrogate Objective (Schulman et al., 2017), which stops the active model's outputs moving too far from the reference mode, without needing to use a KL penalty.

2.4.3 PPO with Transformers

PPO was successfully applied to LLMs by Ziegler et al. (2019) in a process that we refer to as *PPO Transformer RL*, that consists of the following three steps (visually shown in Figure 2.7):

1. **Rollout:** The LLM generates a continuation of a sequence that is passed into it. Following the terminology used by von Werra (2022) we refer to the input sequence as the *query* and the output as the *response*.

2. **Evaluation:** The response is evaluated, and a reward is produced for that response. In the case of a TOD system or US, this is based on how "correct" the generated response is.

3. **Optimization:** The log probabilities of the tokens in the response, given the query, are then calculated. This is done for both the trained (active) model and a reference model (which is the same model as the active model pre-training and does not change). The KL-divergence of the log probabilities is then taken and is used as an additional reward signal to ensure the responses generated by the active model do not deviate too far from the reference model. Finally, the active model is trained using PPO.

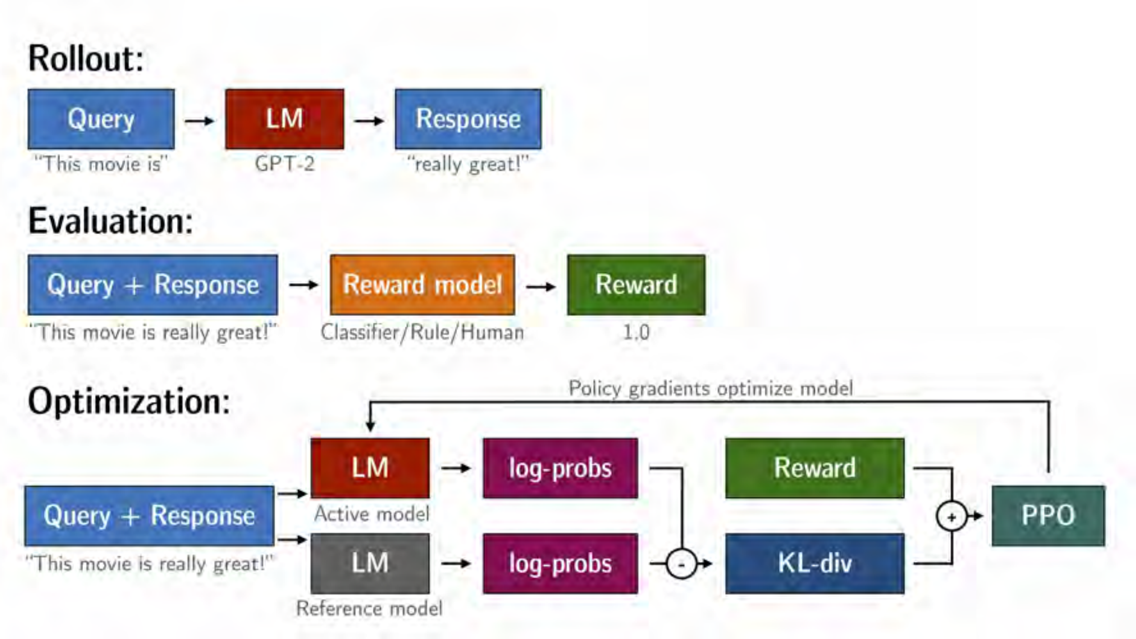


Fig. 2.7 Illustration of the PPO Transformer RL workflow, from von Werra (2022). The example task used in this diagram is taking an LLM that is fine-tuned on movie reviews, and using PPO Transformer RL to refine it to only generate positive reviews.

2.4.4 JOUST

As mentioned above, JOUST works by jointly optimising a TOD system and a US. Figure 2.8 demonstrates the JOUST architecture, where the two systems are built using a pipelined approach. JOUST utilises LSTMs in the DST, context encoder, policy, and NLG modules. The two systems are pre-trained on MultiWOZ 2.0, containing complex multi-domain TOD data[7]. This pre-training enables the two agents to be able to converse using natural language, and therefore provides the ability for RL to be used to jointly optimise the two systems. Importantly, this approach enables the agents to depart from known strategies learned from a fixed limited corpus—and to explore new, potentially improved policies.

The Policy Gradient Theorem (Sutton et al., 1999) is used for joint optimisation. Note that the RL approach taken in JOUST works on the dialogue *policy* module specifically, rather than at the E2E level.

JOUST found that joint optimisation using RL provided a significant boost to performance of the TOD system—including a greater than 10% increase in both the *inform rate* and *success rate*[8]. While these results are promising, the framework is very complex due to its many components. We hypothesise that utilising the simpler and more powerful Transformer-based

[7]The MultiWOZ dataset is discussed in Section 2.5.
[8]The success and inform rates are discussed in Section 4.2.

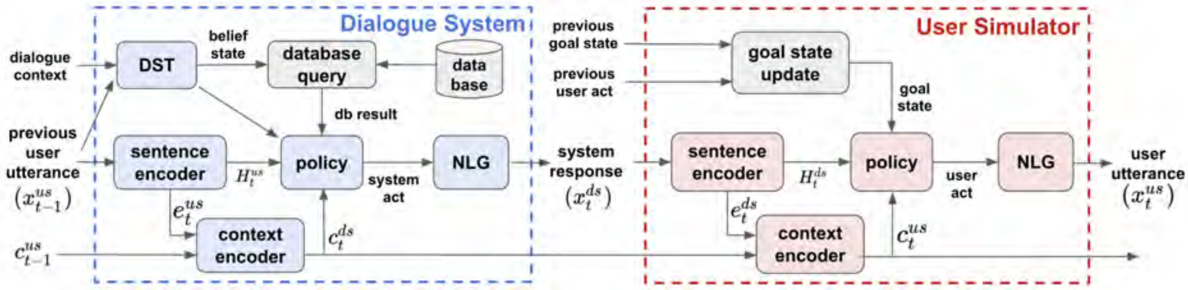


Fig. 2.8 Architecture of the JOUST framework for jointly optimising a TOD system and US, from Tseng et al. (2021). The two systems discourse with each other in a process where RL is used to refine the policy modules of each system jointly. t denotes dialogue turn index. The context encoder is shared between the two agents.

models for both the TOD system and US would lead to significant improvements in performance. We explore this hypothesis in our implementation of PPO Transformer RL, described in Section 3.3.

2.5 The MultiWOZ Dataset

In the above section we covered RL for TOD systems, starting with a brief overview of the field (2.4.1), providing an overview of PPO (Section 2.4.2) and PPO Transformer RL (Section 2.4.3), and finally moving on to the JOUST framework (Section 2.4.4). In this section we cover the MultiWOZ dataset, specifically looking at the MultiWOZ goal model (Section 2.5.1).

MultiWOZ is a fully-labelled collection of human-human written task-oriented conversations spanning over seven domains and a wide range of topics (Budzianowski et al., 2018). It became a very influential dataset in the research community for the development of TOD systems, as it was at least one order of magnitude larger than other previously annotated task-oriented corpora. The domains covered in MultiWOZ are *hotel*, *restaurant*, *taxi*, *train*, *tourist attraction*, *hospital*, and *police*. The full dataset contains approximately 10,400 dialogues, with 1,000 of those used for each of the validation and test sets.

2.5.1 The MultiWOZ Goal Model

The MultiWOZ corpora is based on a multi-domain database that contains a wide range of slot-value pairs representing different real-world entities from Cambridge, United Kingdom. Each dialogue in MultiWOZ is driven by a single user *goal*, where each goal consists of the completion of between 1 and 5 *tasks*. Each task concerns one domain, and is made up of three primary components: *general constraints*, *booking constraints*, and *requests*. The booking constraint is optional, and is only populated in goals where the user wants the system to book something for them (referred to as a booking *sub-goal*).

Within the general and booking constraints there are slot-value pairs which are used to query the database, such as “pricerange: cheap” or “attraction: kings chapel”. The database returns the entities that match all slot-values in the constraints. For some goals there are no entities that match the constraints, and in these cases *fail* constraints are provided which do have at least one match in the database (meaning all goals are possible to successfully complete). Finally, the *requests* component describes information the user wants from the system, such as a reference number or a phone number.

Each goal in MultiWOZ is generated using random sampling, where each of the following parameters are sampled according to set probabilities:

1. Domains that make up the tasks (between 1 and 5)
2. General constraints
3. Fail general constraints **[optional]**
4. A booking sub-goal and relevant constraints if so **[optional]**
5. Requests **[optional]**

An example of the MultiWOZ goal model is shown in Figure 2.9.

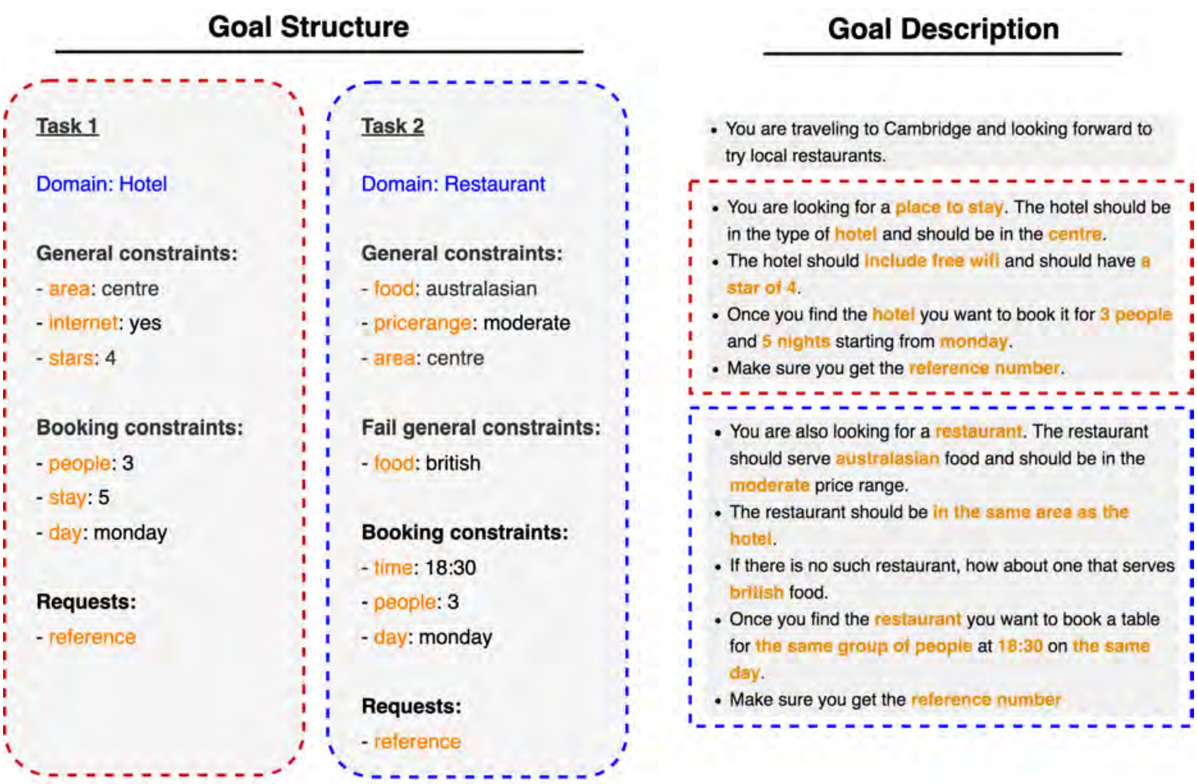


Fig. 2.9 Visual example of a MultiWOZ goal model presented alongside the goal description that was used in the annotation process when the dataset was created. The goal descriptions are from Budzianowski et al. (2018).

Chapter 3

System Design

In the previous chapter we covered the relevant background for this thesis, starting with LLMs (Section 2.1), and moving on to E2E TOD systems (Section 2.2) and User Simulators (Section 2.3). We then covered RL for TOD systems (Section 2.4) and the MultiWOZ dataset (Section 2.5). In this chapter we describe the design and architecture of our two core models: UBAR-E (Section 3.1) and LSP-US (Section 3.2). We then describe our implementation of PPO Transformer RL, a framework for refining LLMs that we adapt to refine UBAR-E (Section 3.3).

3.1 Our Transformer-based TOD System: UBAR-E

Our TOD System is based on the UBAR model developed by Yang et al. (2020), with two extensions outlined below in Sections 3.1.3 and 3.1.4. To distinguish our system from the core UBAR system, we name our system UBAR-E, as in, UBAR-Extended. UBAR-E is trained by fine-tuning GPT-2 (Radford et al., 2019) on the MultiWOZ 2.0 dataset (Budzianowski et al., 2018) using the Huggingface Transformers library (Wolf et al., 2019).

3.1.1 Inference

The architecture of the UBAR-E system is detailed in Figure 3.1, which represents the system's operation during one turn in the middle of a dialogue. At its heart, the system has two sequential generation steps; the first to generate the belief state for the current turn, and the second to generate the system action and system response for the current turn. Both of these generations take the current linearised context history as inputs.

The UBAR-E system effectively models a component-based pipelined TOD System design, leveraging the ability of GPT-2 (Radford et al., 2019) and other large Transformer-based language models to implicitly learn syntactic structure (Brown et al., 2020). Pipelined TOD

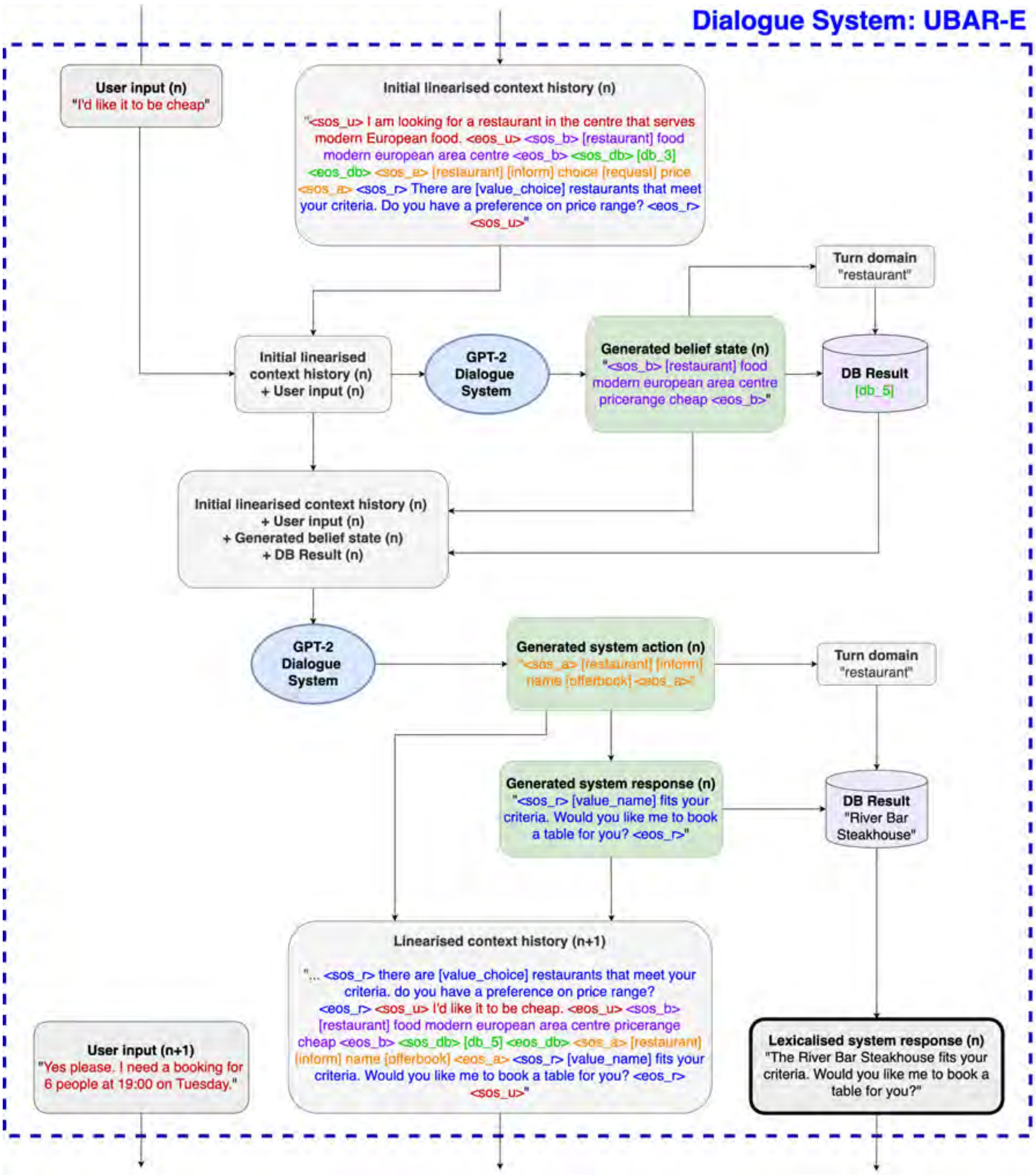


Fig. 3.1 Architecture of UBAR-E, based on UBAR from Yang et al. (2020). We show the operation of the system for the second turn ($n = 2$) in a dialogue where the user wants the system to book a restaurant for them with a number of constraints. The "Initial linearised context history (n)" shows the context from the first turn, and the "Linearised context history (n+1)" shows the context from the second turn (which is processed in this diagram). Blue ovals represent the fine-tuned GPT-2 model that does the actual sequence generation. Green boxes represent generated data. Spans associated with different sub-goals are colour-coded.

systems are generally broken down into four main components, NLU, DST, DP, and NLG, as outlined in Section 2.2. The approach taken by Yang et al. (2020) with UBAR is to model all of these separate modules with one Transformer model.

To make this process more concrete, consider a dialogue session composed of T turns. In the initial turn, $t = 0$, the system takes in a user utterance, U_0, and the fine-tuned GPT-2 model generates a belief state, B_0, which is used to a) infer the turn domain, and b) query the database to get the number of entities that match the constraints contained in the belief state, D_0. It is important to note that both a) and b) are performed outside the Transformer model, as is shown in Figure 3.1. The fine-tuned GPT-2 model then takes in "U_0, B_0, D_0" and generates a system act, A_0, followed by a delexicalised system response, R_0, completing the current turn. As the system moves onto the next turn, $t = 1$, the concatenation of the linearised context history and the next user utterance, "$U_0, B_0, D_0, A_0, R_0, U_1$" are used as input for the fine-tuned GPT-2 model to generate the next belief state, B_1, and so on. The linearised context history builds up over each turn throughout a given dialogue. We can therefore represent the full linearised context history for our dialogue consisting of T turns, as "$U_0, B_0, D_0, A_0, R_0, ..., U_T, B_T, D_T, A_T, R_T$".

As is shown in Figure 3.1 all five components that make up the linearised context history of UBAR are wrapped in span tags, $< sos\_X >$ and $< eos\_X >$, where X is one of $\{u, b, db, a, r\}$, representing user utterance, belief state, database result, system act, and system response, respectively.

3.1.2 Training

In the previous section we described the linearised context history as a concatenation of all five components of the UBAR system, "$..., U_t, B_t, D_t, A_t, R_t, ...$", across all T turns in the dialogue. UBAR-E follows the same training process outlined by Yang et al. (2020) where GPT-2 is fine-tuned on the full linearised context history for each dialogue. An example pre-processed dialogue for UBAR (containing the full linearised context history) is shown in Figure 3.2.

Fig. 3.2 Example pre-processed training dialogue for UBAR. Colours are added for readability.

The training objective is the language modelling objective (Bengio et al., 2003), which maximises the probability of predicting the next word:

$$L = \sum_i \log\left(P w_i | w_{<i}\right) \tag{3.1}$$

Though remarkably simple, this E2E training approach generates very strong results in E2E modelling on the MultiWOZ dataset. Expanding the above, we can present a formal training procedure for UBAR-E[1]:

Let $\mathscr{D} = \{\mathbf{x}_n\}_{n=1}^{N}$ be the training data, comprising N sequences, each of which is a full linearised context history. Learning the UBAR-E model can be formalised as learning a set of model parameters, θ, which characterise the joint probability $p_\theta(\mathscr{D})$. Considering the symbolic notation in Table 3.1:

Symbol	**Definition**
c	Linearised context history
u	User utterance
b	Belief state
d	DB result
a	System act
r	System response

Table 3.1 Symbolic notation for sequence elements in UBAR-E.

where c contains the linearised context history up to the end of the previous turn. We also define K as the number of turns in the current dialogue, n.

$$\mathbf{c}_k = \{\mathbf{u}_i, \mathbf{b}_i, \mathbf{d}_i, \mathbf{a}_i, \mathbf{r}_i\}_{i=1}^{k-1}$$

The joint probability over the sequence $\mathbf{x}_n$ can be factorised in a left-to-right autoregressive manner. We have three terms; one for generating the current turn's belief state, $\mathbf{b}_k$, system act, $\mathbf{a}_k$, and system response, $\mathbf{r}_k$. We take the product over each turn, k, to get the probability of generating the K turns in the dialogue, $\mathbf{x}_n$.

$$p_\theta(\mathbf{x}_n) = \prod_{k=1}^{K} p(\mathbf{r}_k, \mathbf{a}_k, \mathbf{d}_k, \mathbf{b}_k, \mathbf{u}_k, \mathbf{c}_k)$$

$$\propto \prod_{k=1}^{K} p(\mathbf{b}_k | \mathbf{u}_k, \mathbf{c}_k) \;\; p(\mathbf{a}_k | \mathbf{d}_k, \mathbf{b}_k, \mathbf{u}_k, \mathbf{c}_k) \;\; p(\mathbf{r}_k | \mathbf{a}_k, \mathbf{d}_k, \mathbf{b}_k, \mathbf{u}_k, \mathbf{c}_n)$$

[1]We adapt the formalisation from Alexandru Coca's description of a Transformer-based US in his 2021 Cambridge University first year PhD Report: "On Evaluating User Models for Task-Oriented Dialogues".

We can then define three training objectives corresponding to the three sequential generation tasks:

$$\mathcal{L}_b = \sum_{k=1}^{K} \sum_{t=1}^{T_{b,k}} \log p_\theta(b_{t,k} | b_{<t,k}, \mathbf{u}_k, \mathbf{c}_k)$$
$$\mathcal{L}_a = \sum_{k=1}^{K} \sum_{t=1}^{T_{a,k}} \log p_\theta(a_{t,k} | a_{<t,k}, \mathbf{d}_k, \mathbf{b}_k, \mathbf{u}_k, \mathbf{c}_k)$$
$$\mathcal{L}_r = \sum_{k=1}^{K} \sum_{t=1}^{T_{r,k}} \log p_\theta(r_{t,k} | r_{<t,k}, \mathbf{a}_k, \mathbf{d}_k, \mathbf{b}_k, \mathbf{u}_k, \mathbf{c}_k)$$

where $T_{b,k}$, $T_{a,k}$, $T_{r,k}$ represent the number of tokens in the belief state, system act, and system response, respectively, for the current turn, k. The full objective can then be trained with maximum-likelihood training:

$$\mathcal{L}_\theta(\mathcal{D}) = \sum_{i=1}^{N} (\mathcal{L}_b(\mathbf{x}_i) \ + \ \mathcal{L}_a(\mathbf{x}_i) \ + \ \mathcal{L}_r(\mathbf{x}_i))$$

Which links back with Equation 3.1, where the language modelling objective is used to simply predict the next word in what is effectively a very long linear context history for each dialogue.

3.1.3 Lexicalisation

To use UBAR (Yang et al., 2020) with a User Simulator, lexicalisation must be implemented, so the simulator can accurately track its goal state throughout the conversation[2].The lexicalisation process used in UBAR-E is outlined in Figure 3.3. At the heart of the system is the Lexicalisation State Manager (LSM) that keeps track of the most recent constraints provided in the current dialogue, and is updated every turn. This is important as the user can request information about an entry the system has provided in an earlier turn. For dialogues with multiple tasks, the LSM will contain an entry per task that has a valid database entry.

When running through a turn, if there is at least one entity in the database that matches the given constraints in the belief state, then a random entry will be selected from the database, and the values from that entity are used to update the LSM. If there are no entities in the database that match the user's constraints, then the slot-value pairs from the belief state are used to update the LSM. Once the delexicalised system response is generated, the LSM (which has been updated in the current turn), is used to lexicalise the system response.

[2]A general description of lexicalisation is provided in section 2.2.1 and the goal state is outlined in Section 3.2.1

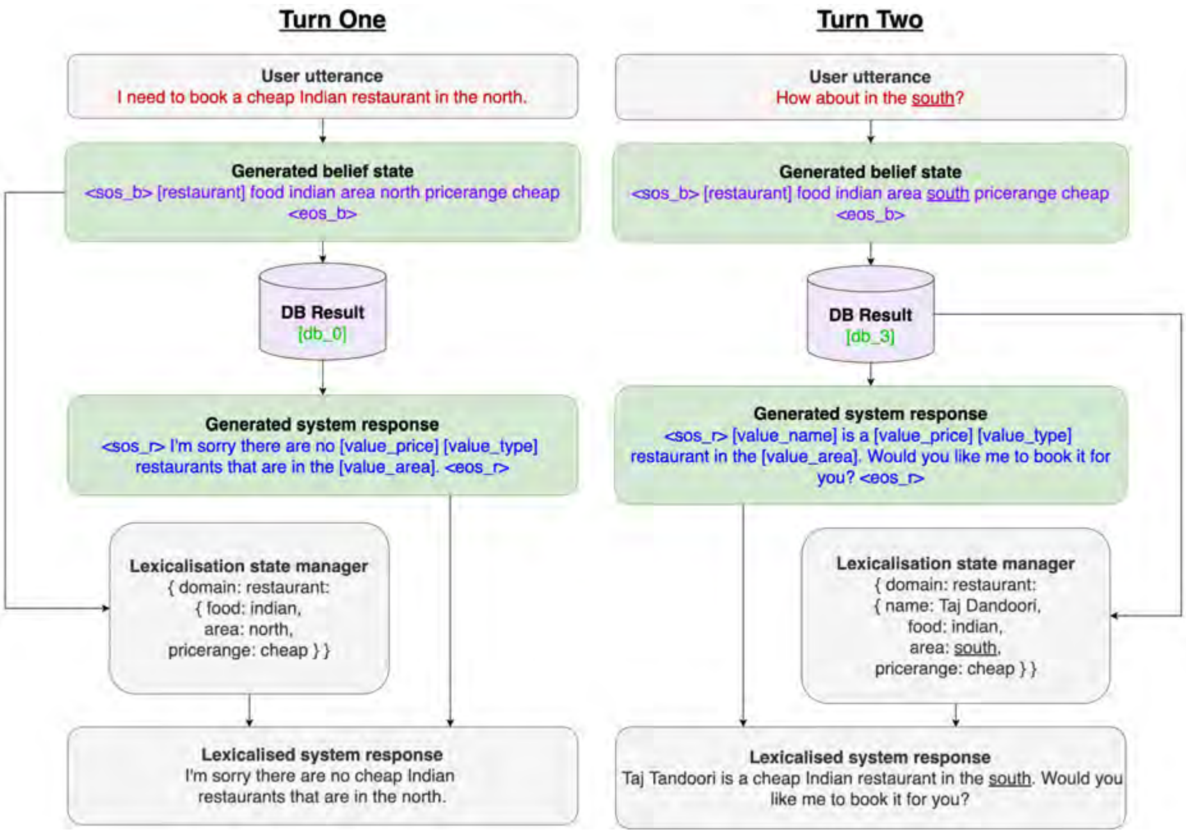


Fig. 3.3 Illustration of the lexicalisation process we implement in UBAR-E. For brevity, we do not show the system act, which is generated before the system response. Note also that this diagram is a simplified version of what actually happens, and should be taken as a high-level representation rather than technical detail. For turn one on the left, we see that there were no entities in the database that satisfied the constraints from the user, and therefore the slot-value pairs from the belief state are used to update the lexicalisation state manager (LSM), which is then used to fill the slots in the "Generated system response". For turn two on the right, we see that there were entities that matched the user's constraints. In this case the LSM is updated with the new information from one of the matched entities, and it is then used to fill the slots in the "Generated system response".

3.1.4 Turn Domain Inference

UBAR uses ground-truth turn-domains to query the database during inference. This is very limiting as it means the model can only operate on labelled dialogues from the MultiWOZ dataset. For an E2E TOD system to be useful in a real-world context, it must be able to operate on new unseen dialogues, driven by whichever goal the user might have. Therefore, as is the case with lexicalisation, turn-domain inference must be implemented for UBAR to interact with a User Simulator that generates new unseen dialogues. Performing turn-domain inference not only makes the system usable in real-world settings, but it also has the potential to improve performance due to annotation errors in the turn domain labels in the MultiWOZ dataset.

Inferring the turn-domain in UBAR-E is a two-step process and is outlined in Figure 3.4. First, after the belief state is generated it is parsed and the turn-domain is extracted from it (the turn-domain is the only part of the generation in square brackets). The turn-domain is then used to query the database to determine how many entries match the constraints from the belief state.

Second, after the system act and system response are generated (the second generation step in the same turn), the domain is extracted once again, but this time using the system act. This is done because when the belief state doesn't contain the correct turn domain, the system act often does[3]. Note that when this discrepancy occurs the database query result is likely incorrect (as it was queried from an incorrect belief state), but this does not appear to stop the model from producing accurate system acts. The turn-domain in the system act follows the same syntax as the belief state, so it can be easily extracted.

If the turn-domain extracted from the system act is different to the turn-domain from the belief state, then the turn-domain tracked by the system is updated and the entire generation process for the current turn (except for the belief state generation) is repeated[4]. This means the intermediary database call will use the updated turn-domain, and this will be used in the generation of a new system act and system response.

3.2 Our Transformer-based User Simulator: LSP-US

In the previous section we covered UBAR-E: our E2E Transformer-based TOD system based on UBAR. We covered the details of UBAR-E inference (Section 3.1.1) and training (Section 3.1.2), and then covered lexicalization (Section 3.1.3) and Turn Domain Inference (Section 3.1.4); our two extensions to UBAR. In this section we cover LSP-US, our novel US. We follow

[3]It would be valuable to investigate why the system act often contains the correct turn domain when the belief state doesn't.

[4]The belief state is updated so that the context provided in future generations is correct.

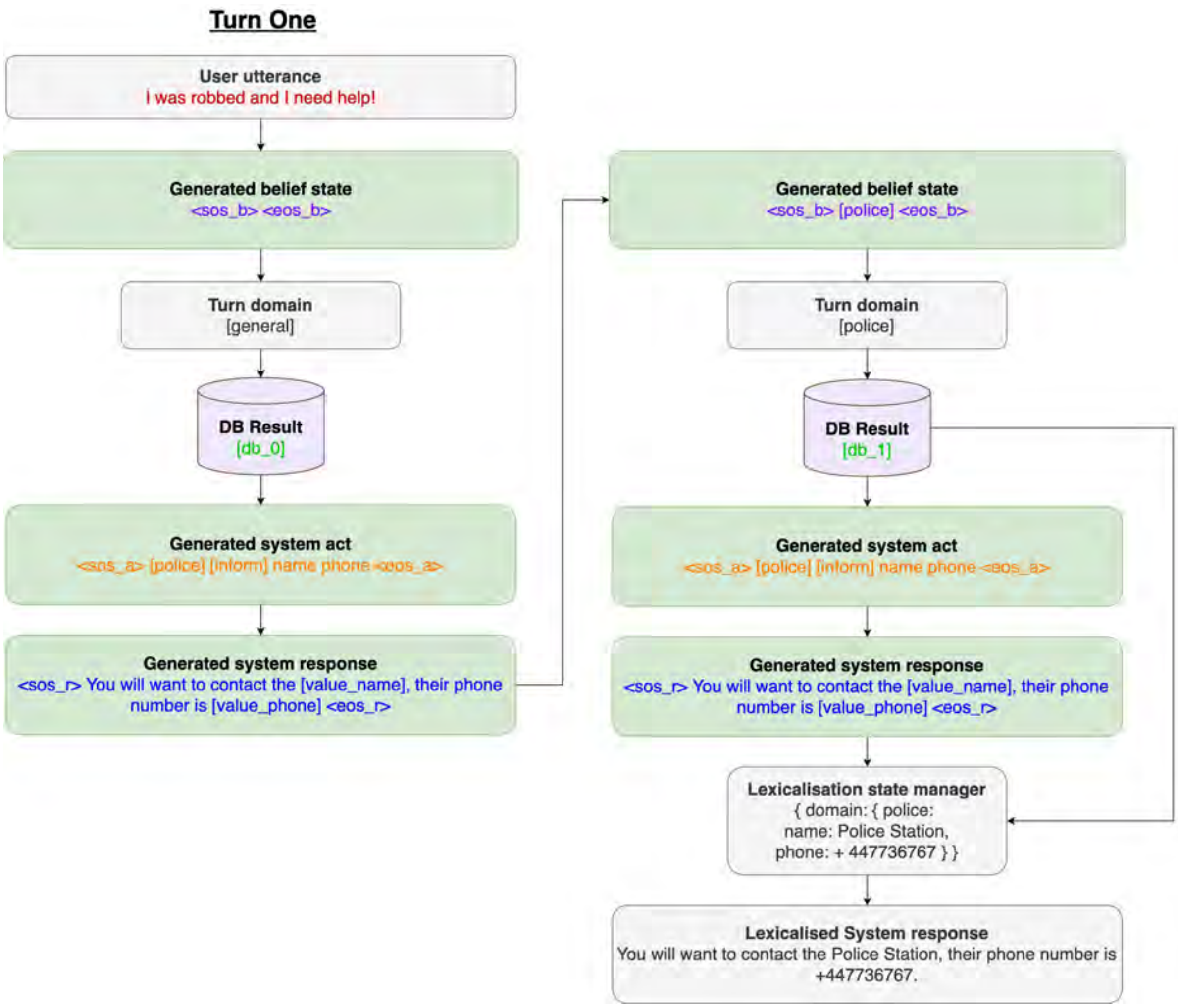


Fig. 3.4 Illustration of the turn-domain inference process we implement in UBAR-E. Note that this diagram is a simplified version of what actually happens, and should be taken as a high-level representation rather than technical detail. In this turn the system has failed to generate the turn domain in the generated belief state, and therefore the turn domain is set to "general". Once the system act is generated, the turn domain is correctly generated ("[police]"), and we therefore update the turn domain internally, update the belief state, and begin the process again at the initial database query.

a similar structure to the previous section, first presenting the details of LSP-US inference (Section 3.2.1), followed by the details of LSP-US training (Section 3.2.2).

LSP-US follows a similar design to UBAR-E at a high-level. There are, however, a number of key differences:

1. LSP-US has is no database.

2. LSP-US uses a goal state module which takes in pre-defined goals and tracks which parts of these goals have been achieved as the dialogue progresses.

3. LSP-US doesn't need to infer the turn domain, as it's tracked in the goal state.

4. LSP-US doesn't maintain a full linearised *context* history, it maintains a linearised *dialogue* history, which is made up of only the user utterances and system responses for all previous turns. Intermediary components, such as actions and goal states, are used by the model, but only in their respective turns (a history is not maintained for generation). More details on this difference are contained in Section 3.2.2.

LSP-US is trained by fine-tuning GPT-2 (Radford et al., 2019) on the MultiWOZ 2.0 dataset Budzianowski et al. (2022) using the Huggingface Transformers library (Wolf et al., 2019), just as UBAR-E is. User goals, which are in the form outlined in Section 2.5.1, are used to drive LSP-US (See Figure 2.9 for a visual example). User goals are randomly sampled using the Convlab-2 goal model, developed by Zhu et al. (2020).

3.2.1 Inference

The architecture of LSP-US is outlined in Figure 3.5 which represents the system's operation during one turn in a given domain. It follows a similar design to Figure 3.1 which outlines the architecture of UBAR-E, and therefore comparing the two diagrams is an effective way to understand the differences between the two models.

Similar to UBAR-E, LSP-US comprises two sequential generation steps for each turn. However, instead of generating a belief state, followed by a system act and response, LSP-US generates a system act and three state tags, followed by a user action and a user response. To make this process more concrete, consider the same scenario described in Section 3.1.1 where we have a dialogue session composed of T turns. In turn, t, the fine-tuned GPT-2 model takes in the concatenation of the linearised *dialogue* history "$R_0, U_0, ..., R_{t-1}, U_{t-1}$"[5] and the system

[5]Note that due to the way LSP-US is designed it always starts by taking in a system response, R_0. In practice, we pass in an empty string as the initial system response, which enables the model to "start" the conversation with a user utterance based on the initial goal state, G_0.

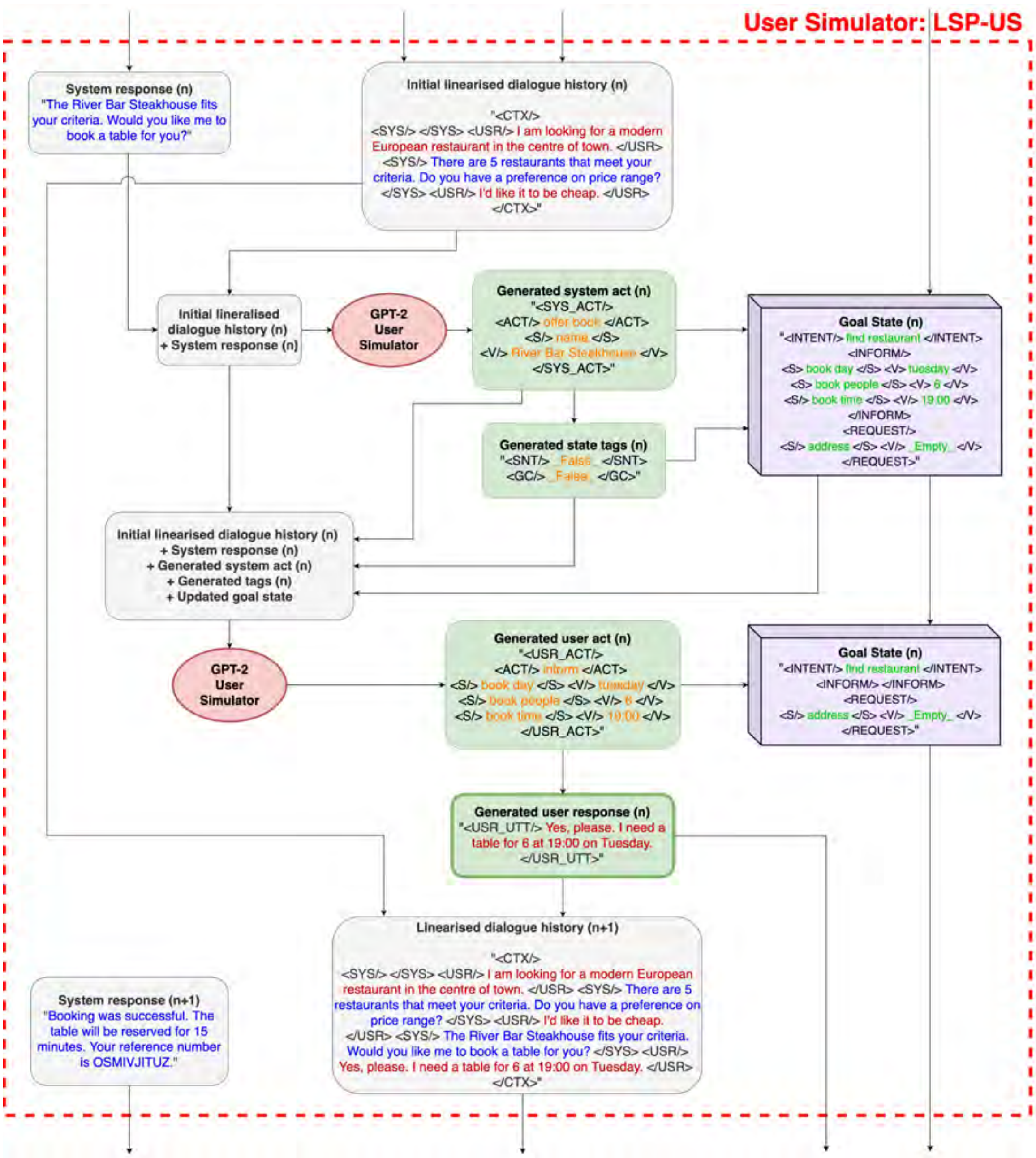


Fig. 3.5 Architecture of our User Simulator: LSP-US. We show the operation of the US for the third turn $(n = 3)$. Figure 3.1 shows the process on the TOD system side for the previous turn $(n = 2)$. In this case the "Initial linearised dialogue history (n)" shows the dialogue history for the first two turns, and the "Linearised dialogue history (n+1)" shows the dialogue history for the first three turns (including the third turn processed in this diagram). Red ovals represent the fine-tuned GPT-2 model that does the actual sequence generation. Green boxes represent generated data. Spans associated with different sub-goals are colour-coded.

response, R_t. It then generates the system act, A_t^S, and three state tags, S_t. The three state tags, $< SNT >$, $< GC >$, and $< RA >$, stand for "Start New Task", "Goal Change", and "Request Another", respectively.

$< SNT >$ indicates whether the US should instruct the dialogue system to begin a new task. It is set to true only when the current task is finished and there are incomplete tasks remaining in the goal. $< GC >$ indicates that the US should change the goal entirely, and leads to the generation of a new goal state. This is set to true when the system fails to fulfil the US's goal, but it could also be used to introduce variability to the system, modelling a user changing their mind mid-way through a certain goal. Finally, $< RA >$ is set to true when the US "wants" the system to provide an alternative entity that meets their constraints, i.e., different from the one offered. This is an optional tag, and like $< GC >$ it can be used to introduce randomness to the system, modelling a user having a pre-determined preference against the entity the system provides.

Together, the system act and state tags effectively represent the new information contained in the system response, R_t. Together, they are used to update the goal state for the first time in the turn, G_t^1. For example, if the TOD system has provided the US with a value that the US requested in the previous turn, such as the phone number for a restaurant, this will be reflected in the goal state at this stage.

In the subsequent generation step for turn t, the fine-tuned GPT-2 model then takes in the linearised dialogue history, system response, system act, state tags, and goal state; "$R_0, U_0, ..., R_{t-1}, U_{t-1}, R_t, A_t^S, S_t, G_t$", and generates a user act, A_t^U, and a user utterance, U_t. At this stage, A_t^U is used to update the goal state for the second time in the turn, G_t^2. An example of this is shown in Figure 3.5 where the user is *inform*ing the TOD system of three pieces of information. In this case, the goal state removes these values from its "INFORM" span, resulting in G_t^2. As a final step, the linearised dialogue history is updated to include the most recent system response and user utterance: "$R_0, U_0, ..., R_{t-1}, U_{t-1}, R_t, U_t$".

3.2.2 Training

LSP-US follows a similar training process to UBAR-E where GPT-2 is fine-tuned using the language modelling objective (Bengio et al., 2003) which maximises the probability of predicting the next word (Equation 3.1). However, in LSP-US the training data is split into *turns*, rather than *dialogues*, where each turn contains a linearised combination of the *dialogue* history and the *current* system response, system act, state tags, goal state, user act, and user response (in their respective spans). An example pre-processed dialogue for LSP-US is shown in Figure 3.6.

Fig. 3.6 Example pre-processed training dialogue for LSP-US. Newline characters are added for readability, along with bold span tags and colours. The actual training data is all one space seperated string with no newline characters. Note that training examples are at the *turn*-level for LSP-US, rather than the *dialogue*-level for UBAR-E.

Learning the LSP-US model can be formalised similarly to UBAR-E[6]:

Let $\mathscr{D} = \{\mathbf{x}_n\}_{n=1}^{N}$ be the training data, comprising N sequences (note that in this case $\mathbf{x}_n$ is a *turn*, not a dialogue as it is in UBAR-E training. Learning the LSP-US model can be formalised as learning a set of model parameters, θ, which characterise the joint probability $p_\theta(\mathscr{D})$. Considering the symbolic notation in Table 3.2:

Symbol	Definition
$\mathbf{c}$	Linearised dialogue context
$\mathbf{r}$	System response
$\mathbf{a}^s$	System act
$\mathbf{s}$	State tags
$\mathbf{g}$	Goal state
$\mathbf{a}^u$	User act
$\mathbf{u}$	User utterance

Table 3.2 Symbolic notation for sequence elements in LSP-US.

where c contains the *dialogue* context up to the end of the previous turn:

$$\mathbf{c}_n = \{\mathbf{u}_i, \mathbf{r}_i\}_{i=n-m}^{n-1}$$

where m is the number of turns that have passed since the current dialogue started.

The joint probability over the sequence $\mathbf{x}_n$ can be factorised in a left-to-right autoregressive manner. We have four terms[7] to generate the system act, $\mathbf{a}^s$, the state tags, $\mathbf{s}$, the user act, $\mathbf{a}^u$, and the user utterance $\mathbf{u}$:

$$\begin{aligned} p_\theta(\mathbf{x}_n) &= p(\mathbf{u}_n, \mathbf{a}^u_n, \mathbf{g}_n, \mathbf{s}_n, \mathbf{a}^s_n, \mathbf{r}_n, \mathbf{c}_n) \\ \propto \quad & p(\mathbf{a}^s_n | \mathbf{r}_n, \mathbf{c}_n) \;\; p(\mathbf{s}_n | \mathbf{a}^s_n, \mathbf{r}_n, \mathbf{c}_n) \;\; p(\mathbf{a}^u_n | \mathbf{g}_n, \mathbf{s}_n, \mathbf{a}^s_n, \mathbf{r}_n, \mathbf{c}_n) \;\; p(\mathbf{u}_n | \mathbf{a}^u_n, \mathbf{g}_n, \mathbf{s}_n, \mathbf{a}^s_n, \mathbf{r}_n, \mathbf{c}_n) \end{aligned}$$

Note that while g_n is updated after the generation of the state tags (i.e. it changes in the middle of turn n), we can represent it as one single variable as it follows a deterministic update mechanism. For the first update in turn, n:

$$p(\mathbf{g}_n | \mathbf{s}_n, \mathbf{a}^s_n, \mathbf{r}_n, \mathbf{c}_n) \begin{cases} 1, & \text{if } \mathbf{g}_n = \mathbf{g}_0 \\ 0, & \text{otherwise} \end{cases}$$

[6]As in Section 3.1.2, we adapt the formalisation from Alexandru Coca's description of a Transformer-based US in his 2021 Cambridge University first year PhD Report: "On Evaluating User Models for Task-Oriented Dialogues".

[7]As opposed to three for UBAR-E (Section 3.1.2).

and for the second update in turn, n:

$$p(\mathbf{g}_n|\mathbf{a}^u_n, \mathbf{s}_n, \mathbf{a}^s_n, \mathbf{r}_n, \mathbf{c}_n) \begin{cases} 1, & \text{if } \mathbf{g}_n = \mathbf{g}_0 \\ 0, & \text{otherwise} \end{cases}$$

where $\mathbf{g}_0$ is the true goal state. We can then define four training objectives corresponding to the four sequential generation tasks:

$$\mathscr{L}_{as} = \sum_{t=1}^{T^s_{a,n}} \log p_\theta(a^s_{t,n}|a^s_{t,<n}, \mathbf{r}_n, \mathbf{c}_n)$$
$$\mathscr{L}_{s} = \sum_{t=1}^{T_{s,n}} \log p_\theta(s_{t,n}|s_{t,<n}, \mathbf{a}^s_n, \mathbf{r}_n, \mathbf{c}_n)$$
$$\mathscr{L}_{au} = \sum_{t=1}^{T^u_{a,n}} \log p_\theta(a^u_{t,n}|a^u_{t,<n}, \mathbf{g}_n, \mathbf{s}_n, \mathbf{a}^s_n, \mathbf{r}_n, \mathbf{c}_n)$$
$$\mathscr{L}_{u} = \sum_{t=1}^{T_{u,n}} \log p_\theta(u_{t,n}|u_{t,<n}, \mathbf{a}^u_n, \mathbf{g}_n, \mathbf{s}_n, \mathbf{a}^s_n, \mathbf{r}_n, \mathbf{c}_n)$$

Where $T^s_{a,n}$, $T_{s,n} = 3$, $T^u_{a,n}$, $T_{u,n}$ represent the number of tokens in the system act, state tags, user act, and user utterance, respectively, for the current turn. The full objective can then be trained with maximum-likelihood training:

$$\mathscr{L}_\theta(\mathscr{D}) = \sum_{i=1}^{N} (\mathscr{L}_{as}(\mathbf{x}_i) \; + \; \mathscr{L}_{s}(\mathbf{x}_i) \; + \; \mathscr{L}_{au}(\mathbf{x}_i)) \; + \; \mathscr{L}_{u}(\mathbf{x}_i))$$

Which links back with Equation 3.1, where we are using the language modelling objective to simply predict the next word.

3.3 PPO Transformer Reinforcement Learning

In the previous section we covered LSP-US, our E2E US, starting with the details of inference (Section 3.2.1) and moving on to the details of training (Section 3.2.2). In the final section of this chapter we outline the details of PPO Transformer RL which we use to optimise UBAR-E.

As outlined in Chapter 1 we implement RL as an extension to our core contributions, with the goal of improving UBAR-E through self-play between it and LSP-US. To be clear we are using RL to optimize UBAR-E, *not* LSP-US; we keep the LSP-US model fixed throughout this process[8]. Our RL implementation uses Proximal Policy Optimization (PPO) (Schulman et al., 2017) in the setup described by Ziegler et al. (2019). We use the Transformer Reinforcement Learning library (von Werra, 2022) in our implementation.

3.3.1 Implementation for UBAR-E

We can split the design of our PPO Transformer RL implementation into three steps that repeat for each turn we process. Fundamentally, these are the same three steps outlined in Section 2.4.3, however they are slightly more complex due to the fact that we perform *two distinct generations* per turn: the generation of the belief state, and the generation of the system act and system response (as described in Section 3.1.1).

First, as shown in Figure 3.7, a Rollout step is performed to generate the belief state for the current turn, using the linearised context history from the end of the previous turn as the *query*. This is followed by an Evaluation step for the generated belief state *response*, where the reward is calculated using the Belief State Reward Module (BSRM). Second, the same process is repeated for the generation of the system act and system response. However, in this step, the reward is calculated using the System Act & Response Reward Module (SARRM). Both reward modules are outlined below.

Third, as shown in Figure 3.8, the Optimization step is then performed for each of the query/response pairs in previous two steps.

Reward Modules

As we outline above, our implementation of PPO Transformer RL contains two distinct Reward Modules—the BSRM and the SARRM. As described, these modules act on each of the two types of generation that occurs for each turn in UBAR-E.

Both modules are rule-based systems, calculating a reward signal based on a set of heuristics for the quality of the generated response. We describe the criteria each module follows, first outlining the commonalities and then the individual criteria.

Common Criteria

- The domain in the active task within the goal state of the US should match the domain in the generated *response*.

[8]In Section 5.5.3 we discuss the limitations of this approach, and recommend jointly optimising both systems as is done in Tseng et al. (2021).

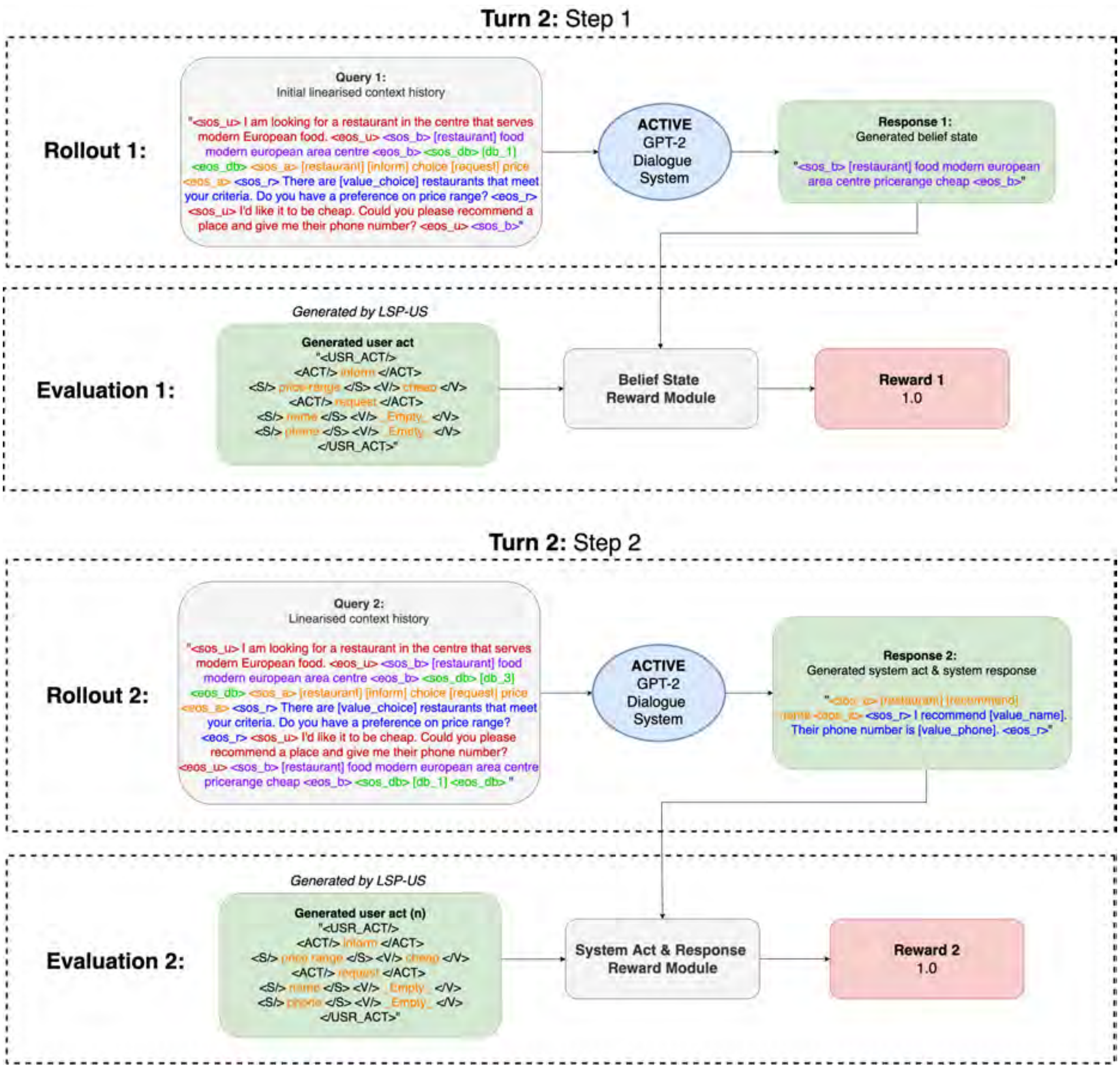


Fig. 3.7 Step one and two of the PPO Transformer RL workflow used to refine UBAR-E. In the first Rollout step the belief state is generated. Then, in the first Evaluation step (which occurs *after* the user act is generated by LSP-US), the user act is compared to the belief state to generate the first reward. In the second Rollout step, the system act and system response are generated. Then, in the second Evaluation step, the user act is compared to the system act and system response to generate the second reward.

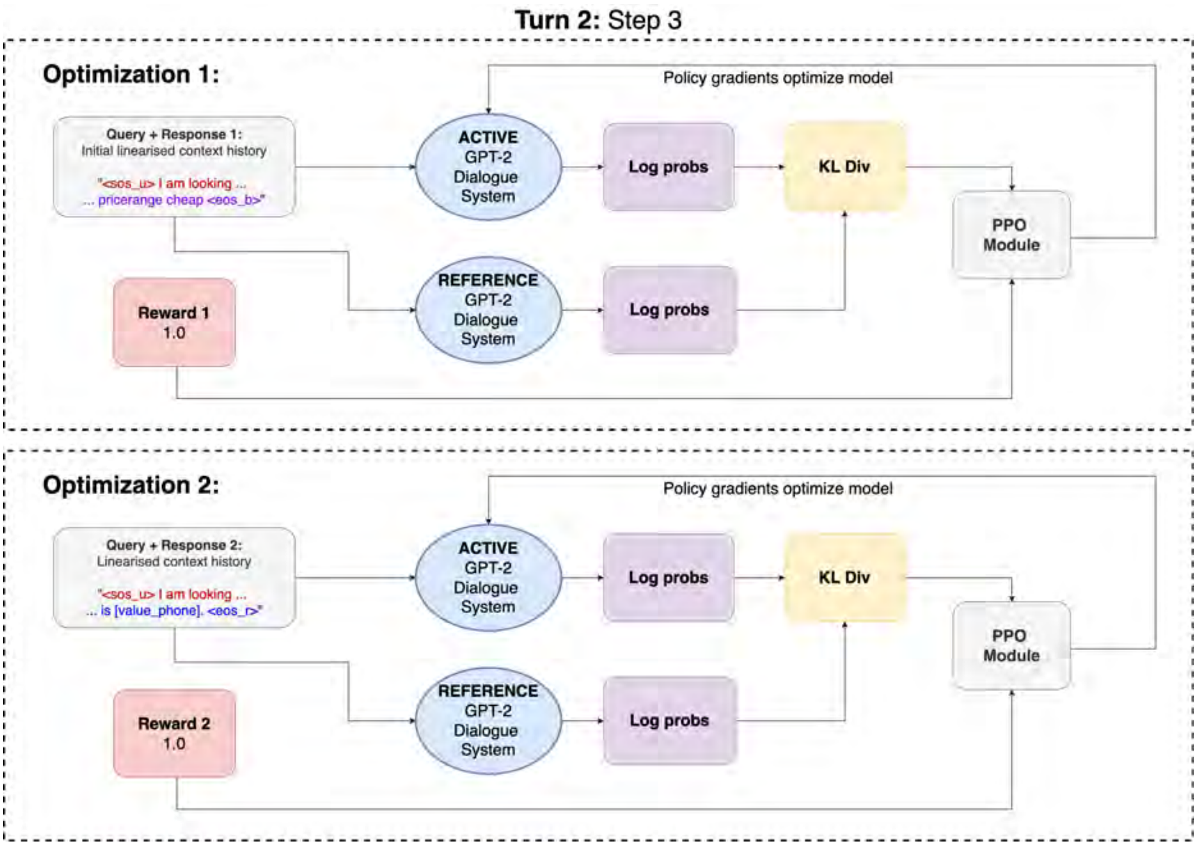


Fig. 3.8 Step three of the PPO Transformer RL workflow used to refine UBAR-E. Each Optimization step uses the respective Query + Response pair, and associated reward, from steps 1 and 2 (outlined in Figure 3.7).

- Every slot-value pair that is generated should be unique, as in, the system should not duplicate pairs. This is an important signal as we have found GPT-2 (Radford et al., 2019) often generates repetitive sequences as it learns.

- The appropriate start-of-sentence and end-of-sentence tags should appear *once* per span, in the correct positions, and no other span tags should be generated. For example, in the generation of the belief state, one “<sos_b>” should start the span, and one “<eos_b>” should end the span.

Belief State Reward Module Criteria

- Every slot-value pair in the “inform” part of the generated user act should be present in the belief state. For example, if “<S/> book day </S> <V/> tuesday </V>” is the entry in the “inform” part of the generated user act, the belief state should include "day tuesday".

- The belief state should always contain at least the turn-domain.

System Act & Response Reward Module Criteria

- Every slot in the “request” part of the generated user act should be present in the generated system act. For example, if “<S/> phone </S> <V/> _Empty_ </V>” is in the “request” part of the generated user act, the system act should include "[inform] phone [phone_value]".

- The generated system act should always contain at least the current turn-domain, an action type (e.g. “request”), and one value (e.g. “price”).

- The generated system response should contain all the information present in the slot-value pairs in the system act.

This rule-based reward signal has obvious limitations, including the fact that we are not factoring in the fluency of the system response (we merely ensure the correct information is provided to the user). This is discussed in Section 5.5.2.

Chapter 4

Experimental Setup

In the previous chapter we described the design and architecture of our two core systems: UBAR-E (Section 3.1), and LSP-US (Section 3.2). We then described our implementation of PPO Transformer RL (Section 3.3). In this chapter we describe the dataset (Section 4.1), metrics (Section 4.2), and training details (Section 4.3) for our core experiments. Finally, we cover the experimental setup of our PPO Transformer RL experiments (Section 4.4).

4.1 Dataset

We report results on the MultiWOZ 2.0 dataset (Budzianowski et al., 2018) (introduced in Section 2.5). We utilise all seven domains in training (*hotel*, *restaurant*, *taxi*, *train*, *tourist attraction*, *hospital*, and *police*). MultiWOZ 2.0 has 8439 dialogues in the training set and 1000 in each of the validation and test sets. Each dialogue contains 2 to 5 different domains (where there is one *task*[1] per domain. We utilise the fully annotated version of MultiWOZ 2.0 from Lee et al. (2019) which contains belief states and system acts for each turn.

4.1.1 Pre-processing for UBAR-E

We perform domain-adaptive pre-processing to prepare the MultiWOZ dataset for training, as outlined by Yang et al. (2020). This can be summarised with a 6-step process (an example of the structure of a fully pre-processed dialogue is shown in Figure 3.2):

1. **Delexicalisation:** system responses are delexicalised, which is important for the model to learn value-independent parameters (Wen et al., 2015). A domain-adaptive delexicalisation scheme (Zhang et al., 2020a) is used, which decouples the domain and slot name

[1]See section 2.5.1 for an explanation of tasks.

of placeholders in the delexicalised system responses. For example, instead of having <*restaurant-value_name*>, <*attraction-value_name*>, <*hotel-value_name*> etc., there is simply <*value_name*> that represents the name of an entity in any domain.

2. **Process Belief States:** similar to the delexicalisation process outlined above, the belief states are converted using a domain-adaptive scheme. This allows the model to generalise across domains that share the same ontology. Belief states are originally represented with {*domain-slot: value*} pairs; for example, {*restaurant-name: river bar steakhouse*}. In the domain-adaptive scheme, we represent belief states in the format {*[domain1] slot value slot value [domain2] slot value ...*}. For example: {*[restaurant] name river bar steakhouse day tuesday [hotel] name university arms hotel ...*}.

3. **Process System Acts:** The same domain-adaptive scheme that is applied to the belief states is applied to the system acts.

4. **Process Database Query Results:** for each turn, the number of entities in the database that match the constraints outlined by the user are represented with a set of four special tokens: *[db_0]* for 0 matching entities, *[db_1]* for 1, *[db_2]* for 2-3, and *[db_3]* for >3. These tokens are processed slightly differently for the train domain, as it contains significantly more entities than the other domains, due to each departure time constituting an entity. The same tokens are used, but they represent 0, 1-5, 6-10, and >10 matching entities, respectively.

5. **Add span tags:** span tags are added to each component within a dialogue so the model can more easily learn the correct syntactic structure, e.g. <*sos_a*> and <*eos_a*> are added to the start and end of the system acts, as per Figure 3.2.

6. **Linearise dialogues:** as UBAR is trained on fully *linearised context histories* (described in Section 3.1.1), each dialogue is linearised, including its intermediary states[2].

4.1.2 Pre-processing for LSP-US

Preprocessing for LSP-US follows a related but different process to UBAR. We outline these steps below (an example of the structure of a fully pre-processed dialogue for training LSP-US is shown in Figure 3.6):

1. **Process system acts**: a domain-adaptive scheme is applied to system acts. This is similar to UBAR, however the syntax is different in that it does not include the domain. This is

[2]Intermediary states are the belief state, the database result, and the system act.

because LSP-US operates at the *turn*-level, and each turn only deals with one domain. The domain can therefore simply be represented in the goal state.

2. **Process user acts**: the same domain-adaptive scheme that is applied to the system acts is applied to the user acts.

3. **Create goal states**: the goal state must be created for each turn, where it is updated based on the user act generated in the previous turn (see Section 3.2.1 for details). The goal state can be thought of as a list of sub-goals the US is aiming to achieve, and as each sub-goal is achieved it is removed from the goal state. When the goal is empty, it has been successful and the US can terminate the dialogue.

4. **Create state tags**: state tags must be created for each turn, representing whether a new task should be started (*<SNT>*), a goal change should be triggered (*<GC>*), or another entity should be requested (*<RA>*). As above, see Section 3.2.1 for details.

5. **Add span tags**: as with UBAR, span tags are added to each component within a dialogue so the model can more easily learn the correct syntactic structure, e.g. *<GOAL/>* and *</GOAL>* tags are added to the start and end of the goal state, as per Figure 3.6.

6. **Linearise *turns***: this step contains a key difference to UBAR in that training examples are *turns* rather than dialogues. As demonstrated in Figure 3.6 each linearised turn contains the full *dialogue history*[3] up to that point, followed by the additional components *for that turn*. This is a critical difference between the two models, and is best understood by comparing Figures 3.2 and 3.6.

4.2 Metrics

We use the official MultiWOZ evaluation scripts released by Nekvinda and Dušek (2021) to assess the performance of our TOD systems. The official list of models and their performance are maintained at the MultiWOZ GitHub page (Budzianowski et al., 2022). The evaluation scripts report a number of metrics, with the main ones being *BLEU*, *Inform*, *Success*, and *Score*, where $Score = BLEU + ((Inform + Success)/2)$.

BLEU (Bilingual Evaluation Understudy Score) (Papineni et al., 2002) was originally designed for machine translation and is based on the comparison of n-grams between machine-generated sequences and corresponding human-written reference sequences. A BLEU score of

[3] As explained in Section 3.2.1, the dialogue history contains a concatenation of user utterances and system responses.

1.0 correlates with a perfect match and 0.0 with a perfect mismatch. Wen et al. (2017) first used BLEU as a *fluency* metric, and since then it has become a standard metric used in this regard for E2E TOD systems. While simple to compute, BLEU is a rather poor fluency metric as there is only a single reference available and the set of valid responses is arguably larger for dialogues than machine translation (Nekvinda and Dušek, 2021). Further, Liu et al. (2016) show that metrics adopted from machine translation have very weak correlation with human judgements in dialogue responses. Despite these flaws, it is still used as a key part of the official evaluation criteria for TOD systems on MultiWOZ.

The *Inform* rate relates to *informable* slots, which allow the user to *inform* the TOD system of constraints related to its goal (e.g. price range or day of booking). The Inform rate measures the proportion of dialogues where the last offered entity (from the database) satisfies the goal state, *for each domain in the user's dialogue goal*. On the other hand, the *success* rate relates to both *informable* and *requestable* slots, where *requestable* slots allow the user to request specific information (e.g. a booking reference or phone number). The *success* rate measures the proportion of fully successful dialogues, where a successful dialogue is one where a) the inform rate is 100%, and b) the system has mentioned all requestable slots to the user. At a high-level, the *inform* rate effectively measures how effective the model is at providing appropriate entities to the user, and the *success* rate effectively measures how effective the model is at successfully completing all requirements within full dialogues. Both *Inform* and *Success* rates are calculated at the dialogue-level.

On top of the metrics already introduced, in line with the results reported on the official MultiWOZ GitHub page, we also report four metrics related to the lexical richness of the model. These consist of the average number of tokens in generated responses (A. Len.), the conditional bigram entropy (CBE), the number of distinct uni-grams (#U1), and the number of distinct tri-grams (#U3).

4.3 UBAR-E and LSP-US Training Details

All experiments were performed with an NVIDIA A100-SXM-80GB GPU, and all experiments use HuggingFace's Transformer library (Wolf et al., 2019).

4.3.1 UBAR-E

UBAR-E and related models are all trained by fine-tuning Distil-GPT2 (Li et al., 2021), except for one model trained by fine-tuning GPT-2 small[4] (Radford et al., 2019). Both models are provided by the HuggingFace Transformers Library (Wolf et al., 2019). Distil-GPT2 is a faster, lighter version of GPT-2[5] that was trained using knowledge distillation, presented by Sanh et al. (2019). Distil-GPT2 has 84 million parameters, compared to 124 million for GPT-2 small. We find that fine-tuning takes approximately 2.5X less time using Distil-GPT2 compared to GPT-2 small (approximately 1 minute vs. 2.5 minutes per epoch).

An AdamW optimizer was used with a learning rate of $1e-4$ and no weight decay. A linear warm-up schedule was used over the first 20% of training steps. A batch size of 16 was used with gradients accumulated for 2 batches (effectively a batch size of 32)[6]. Greedy decoding with a temperature of 0.7 was used.

4.3.2 LSP-US

LSP-US was trained by fine-tuning GPT-2 small using teacher forcing. An AdamW optimizer was used with a learning rate of $6.25e-5$ and weight decay of 0.01. No warm-up schedule was used. A batch size of 4 was used, with gradients accumulated for 4 batches (effectively a batch size of 16). Greedy decoding with a temperature of 1.0 was used.

4.4 PPO Transformer Reinforcement Learning

As with training UBAR-E and LSP-US, we used an NVIDIA A100-SXM-80GB GPU for our PPO Transformer RL experiments. We used Adaptive KL control (Schulman et al., 2017), a batch size of 1 (with a forward batch size of 1), a learning rate of $8.8e-7$, and default parameters for the rest[7]. This implementation (without batching) gave us the opportunity to quickly evaluate our proposed design outlined in Section 3.3. It takes approximately 6 seconds on average to run the entire PPO Transformer RL process for each dialogue using our GPU. We use *both* a KL penalty and a Clipped Surrogate Objective to ensure our model outputs do not move too far from our reference model (see Section 2.4.2).

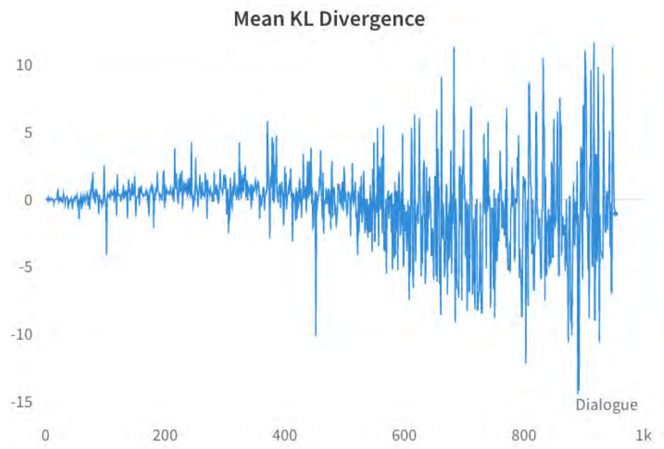
We experience a phenomenon when refining UBAR-E with PPO Transformer RL where the model "collapses" at an early stage in the training process, usually after a few hundred

[4] GPT-2 is available in a number of different sizes, with small being the smallest and most commonly used version.
[5] An outline of the architecture of GPT-2 is provided in Section 2.1.2.
[6] Yang et al. (2020) used a batch size of 2 with 16 gradient accumulation steps when training UBAR.
[7] Default parameters as defined by von Werra (2022).

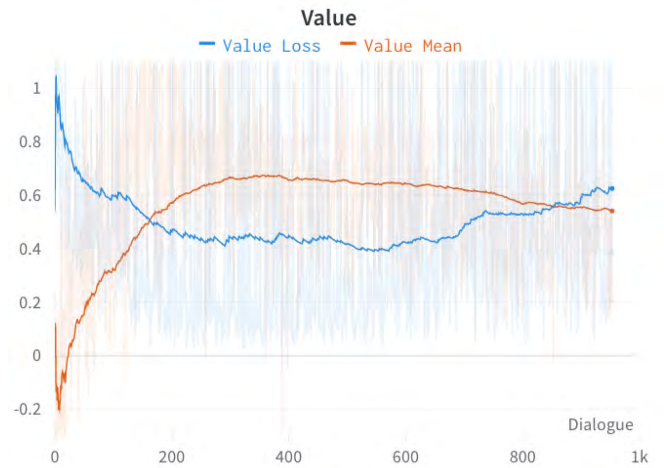
dialogues. When we say the model "collapses", we mean the outputs all of a sudden go from "sensible" (where most dialogues are successful and the syntactic structure generated is robust), to complete nonsense—at which point the model gets stuck in a state from which the model cannot recover. While we have not been able to diagnose this behaviour, we believe it is related to instability in the KL divergence in our implementation. As Figure 4.1a shows, the KL divergence oscillates between negative and positive values during training. We do expect its magnitude to increase over time, as we observe, due to the log probabilities output by the active model slowly diverging from those of the reference model. However, the oscillating behaviour suggests that regularization is not working effectively.

We also present the value loss and value mean during training in Figure 4.1b, where the value is used to estimate the (literal) *value* of the current state of the model (as outlined in Section 2.4.2). We see that the behaviour of the value mean and value loss are as expected up until approximately 400 dialogues. The value mean is increasing, and the value loss is dropping—suggesting the model is improving in line with our reward signals. However, after around 400 dialogues, performance starts to degrade. Finally, we present the mean rewards during training in Figure 4.1c. Here we once again observe unexpected behaviour, where both reward signals gradually decrease over time.

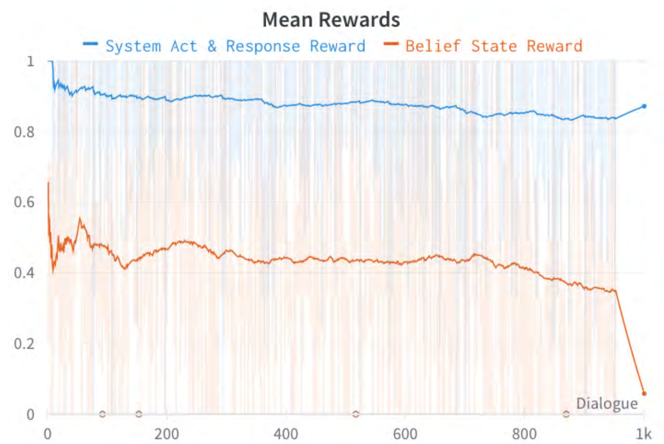
In summary, we observe a range of unexpected behaviour when refining UBAR-E with our initial implementation of PPO Transformer RL. We believe this behaviour is related to instability in our implementation of the KL divergence. As a next step, an implementation using just the Clipped Surrogate Objective, with no KL penalty, should be explored. In the next chapter we discuss these results further, including recommendations for future work.



(a) Mean KL Divergence



(b) Value Loss and Value Mean



(c) Mean Rewards

Fig. 4.1 Training curves from refining UBAR-E using PPO Transformer RL. The first 1000 dialogues are presented.

Chapter 5

Results and Discussion

In the previous chapter we described the dataset (Section 4.1), metrics (Section 4.2), and training details (Section 4.3) for our core experiments. We then covered the experimental setup of our PPO Transformer RL experiments (Section 4.4). In this chapter we present the results from extending UBAR (Section 5.1), from our experiments using synthetic data (Section 5.2), and from our evaluation of LSP-US (Section 5.3). We then present an analysis of MultiWOZ evaluation more broadly (Section 5.4), and finally we present the results from our PPO Transformer RL experiments (Section 5.5).

A summary of our main results are presented in Table 5.1 below, which is accompanied by Table 5.2. We discuss these results in the subsequent sections.

5.1 Extending UBAR

As described in Section 3.1.3 and 3.1.4 we extend UBAR to include the use of lexicalisation and inferred turn domains (ITDs) to create UBAR-E. This enables the system to interact with a user or User Simulator on unseen dialogues—making it truly E2E. In this section we focus on the ITD extension, specifically referring to the results in lines 1 to 4 of Table 5.1.

We assess how the use of ITDs effect the performance of both the best model provided by the authors of the UBAR paper ("UBAR best"), and our best version of UBAR trained from scratch ("UBAR-E"). We see that in both cases the use of ITD provides very similar BLEU, Inform, and Success rates to the models that use ground truth turn domain labels. We do see the use of ITD slightly decreases the BLEU and Success scores in both cases (resulting in the slightly lower Scores), but it is very encouraging to see that this drop is relatively small. We also compare the results on the four metrics concerned with lexical richness. We see no significant change in the average length of system responses, nor do we see a significant change in the conditional bigram entropy. We do, however, see a notable increase in the number of

	Model	BLEU	Inform	Success	Score	A. Len.	CBE	#U1	#U3
1	UBAR best	17.6	86.4	73.4	97.5	13.5	2.09	478	5,206
2	UBAR best ITD	16.9	86.8	72.5	96.6	13.6	2.11	501	5,410
3	UBAR-E	17.8	86.9	**75.4**	**99.0**	13.3	2.04	424	4,214
4	UBAR-E ITD	16.7	86.7	73.3	96.7	13.4	2.05	443	4,544
5	UBAR-E Syn. Utt. A.	15.3	81.9	58.3	85.4	12.8	2.06	430	4,544
6	UBAR-E Syn. Data	11.3	64.4	42.7	64.9	11.5	1.75	331	2,256
7	UBAR-E Syn. Data A.	17.0	81.0	67.9	91.4	13.4	2.1	491	5,085
8	UBAR-E GPT-2	**18.2**	85.0	72.0	96.8	13.4	2.10	462	5,211
9	UBAR-E PPO-TRL	11.5	**90.5**	71.7	92.6	**17.1**	**2.47**	**550**	**7,463**
10	UBAR official	17.6	83.4	70.3	94.4	13.5	2.10	478	5,238
11	PPTOD official	**18.2**	83.1	72.7	96.1	12.7	1.88	301	2,538
12	Reference	-	93.7	90.9	-	14.0	3.01	1407	23,877

Table 5.1 Summary of our main results. “A. Len.” refers to the average number of tokens in the system responses, “CBE” stands for Conditional Bigram Entropy, “#U1” refers to the number of distinct uni-grams, and “#U3” refers to the number of distinct tri-grams. All models are trained on MultiWOZ 2.0. See Table 5.2 for descriptions of each of the models listed above.

distinct uni-grams and tri-grams in the system responses. This suggests the use of ITD may be introducing increased diversity to the system responses. However, another possible explanation is that in cases where the ITD is wrong, the system response generated is very different to the model’s “standard” responses. This hypothesis is consistent with the small decrease in the BLEU and Success rates when using ITDs.

Overall, these results show that our implementation of ITDs works well in practice, and while there is room for improvement, it is a strong foundation for self-play experiments that require UBAR to interact with unseen dialogues.

5.2 Using Synthetic Dialogues in Training

In this section we discuss the use of synthetic data for training, where the synthetic data is created from the outputs of one or both of our core models: LSP-US and UBAR-E.

5.2.1 Synthetic User Utterances

In this experiment we first replaced the human-labelled user utterances in the MultiWOZ 2.0 dataset with synthetic user utterances, generated by LSP-US. We then reproduced the training procedure for UBAR-E outlined in Section 4.3.1, where we fine-tune Distil-GPT2, but in this

	Model	TDs	Description
1	UBAR best	GT	The best model from the UBAR paper, published on GitHub (Yang et al., 2021).
2	UBAR best ITD	ITD	UBAR best, assessed using inferred turn domains (ITDs) rather than GT turn domains.
3	UBAR-E	GT	Our best replication of UBAR trained from scratch with Distil-GPT2.
4	UBAR-E ITD	ITD	UBAR-E, assessed using ITDs rather than GT turn domains.
5	UBAR-E Syn. Utt. A.	GT	UBAR-E trained on a version of the MultiWOZ 2.0 dataset with synthetic utterances (generated by LSP-US) in place of the human-labelled user utterances, in *Addition* to the original MultiWOZ 2.0 dataset. A total of 16,878 training dialogues were used (2X the original dataset).
6	UBAR-E Syn. Data	GT	UBAR-E except it is trained on a fully synthetic dataset of 8439 dialogues generated by LSP-US and UBAR-E, i.e., Distil-GPT2 is fine-tuned on a synthetic dataset following the same training process as UBAR-E.
7	UBAR-E Syn. Data A.	GT	UBAR-E Syn. Data except instead of being trained solely on the synthetic dataset, it is trained on the synthetic dataset in *Addition* to the original MultiWOZ 2.0 dataset. A total of 16,878 training dialogues were used (2X the original dataset).
8	UBAR-E GPT-2	GT	UBAR-E but trained by fine-tuning GPT-2 instead of Distil-GPT2.
9	UBAR-E PPO-TRL	GT	UBAR-E refined on 900 dialogues using sampled goals, with our PPO Transformer RL training approach (described in Sections 3.3 and 4.4).
10	UBAR official	OE	Official results for UBAR from the MultiWOZ GitHub repository (Budzianowski et al., 2022).
11	PPTOD official	OE	Official results for PPTOD from the MultiWOZ GitHub repository. We include PPTOD as it uses a related approach to UBAR, and is therefore useful as an additional baseline.
12	Reference	-	The reference corpus for MultiWOZ 2.2. from the MultiWOZ GitHub repository (MultiWOZ 2.2 is used in the official MultiWOZ evaluation scripts (Nekvinda and Dušek, 2021)).

Table 5.2 Descriptions of each model outlined in Table 5.1. “TDs” stands for Turn Domains. Within the “TDs” column: “GT” stands for Ground Truth, “ITD” stands for Inferred Turn Domain (and refers to our ITD implementation outlined in Section 3.1.4), and “OE” stands for Official Evaluation which refers to the inference process used by Nekvinda and Dušek (2021) when a model uses GT TDs.

case we use the training set with synthetic user utterances *plus* the original dataset. That is, we train Distil-GPT2 on twice as much data. The intention of this experiment is to assess whether the inclusion of the synthetic user utterances provides enough diversity to the dataset to improve UBAR-E's lexical richness, while also *not* causing overfitting due to the rest of the data in the two training sets being identical.

We see in line 5 of Table 5.1 that the BLEU, Inform, and Success rates all *decrease* relative to our baseline model, UBAR-E. It is interesting to note that the Success rate drops substantially more than the BLEU and Inform rates. This suggests that the model is still relatively effective at providing the required *informable* slots the majority of the time. However, the model is either relatively ineffective at providing the required *requestable* slots, or frequently fails on certain *informable* slots (either of which would explain the low Success rate and fairly high Inform rate).

Overall, we see that this approach does not work well. The vast majority of the data in the new training set with synthetic user utterances is identical to the original dataset (everything but the user utterances are the same). We therefore hypothesise that UBAR-E does indeed overfit to the data, as suspected, and this degrades performance.

5.2.2 Fully Synthetic Dataset

Next, we experiment with using *fully synthetic* data to train UBAR-E, in place of the standard MultiWOZ 2.0 data. To generate a fully synthetic dataset of 8,349 dialogues (the same number as the MultiWOZ 2.0 training set) we first sample 12,000 goals using the Convlab-2 goal model, developed by Zhu et al. (2020). We then use *self-play* to get LSP-US and UBAR-E to interact, generating full unseen dialogues based on the previously generated goals. See Figure 1.2 for a high-level representation of the self-play process. Due to mistakes both systems make, approximately 22% of dialogues are unsuccessful, which is why we need to sample the 12,000 goals to be confident we will get at least 8349 successful dialogues. These dialogues contain the user utterances from LSP-US and the system responses and intermediary components (the belief states, system acts etc.) from UBAR-E.

Once we have a fully synthetic dataset that matches the size of the MultiWOZ 2.0 training set, we perform two experiments. The first is to reproduce the training procedure for UBAR-E where we fine-tune Distil-GPT2, but in this case we do so using just the synthetic dataset. The second is to do the same but instead of using just the synthetic dataset, we use the combination of both the synthetic dataset and the MultiWOZ 2.0 training set[1]. We see in lines 6 and 7 of Table 5.1 that the BLEU, Inform, and Success rates all drop for both approaches, relative to

[1]Resulting in a combined training set of 16,878 dialogues.

UBAR-E. We see these metrics drop dramatically for the version trained on *just* the synthetic data, on line 6. We also see that all four of the lexical richness metrics are significantly lower for this model than any other reported. The poor performance of the model trained only on synthetic data is very likely due to the propagation of errors from the model being trained on its own outputs. This circular approach has obvious challenges—the model outputs effectively need to be better than the human-labelled responses (that the model was trained on) for this to lead to performance improvements, so our results here are unsurprising.

What is more interesting is what happens when we reproduce the training procedure for UBAR-E using the combination of the synthetic dataset and the MultiWOZ 2.0 training set. While this approach has some of the same problems just outlined, we are interested in whether the synthetic data will increase the lexical richness of the responses due to increased diversity in the training set. In this setup, we now have twice as many unique dialogues, all based on different goals. Looking at line 7 in Table 5.1 we see that the model does indeed present a higher lexical richness than UBAR-E, with 5,085 unique tri-grams generated as opposed to 4,214. We see a similar trend for unique uni-grams generated, with 491 compared to 424. Encouragingly, the conditional bigram entropy has *increased* slightly, and the average response length has not increased[2].

While these results are very encouraging, unsurprisingly, we also see the BLEU, Inform, and Success scores degrade relative to UBAR-E. However, this degradation is not catastrophic (a Score of 91.4 vs 99.0). We believe it would be valuable to continue this research to try to find ways to leverage the increased diversity in the system responses that this approach has generated, while achieving lower degradation in the Inform and Success rates.

5.3 Evaluating LSP-US

In this section we assess the lexical richness of LSP-US and compare it to UBAR-E. For LSP-US to be an effective User Simulator, it must retain sufficient diversity in its responses. In Table 5.3 we outline the results of two experiments assessing this. To perform these experiments we modified the official MultiWOZ evaluation scripts from Nekvinda and Dušek (2021) to assess the lexical richness of user utterances rather than system responses. In line 4 and line 7 we compare the ratio of the results from our models, LSP-US and UBAR-E, to the results using their respective reference corpora. We see that A. Len. and CBE drop further from the reference corpus for LSP-US compared to UBAR-E. However, we also see that both #U1 and #U3 drop substantially *less* relative to the reference corpus for LSP-US compared to UBAR-E.

[2] A substantial increase in the average response length would suggest the other metrics may have simply increased due to the model generating longer responses.

	Model	A. Len.	CBE	#U1	#U3
1	User Reference (UR)	11.7	3.43	1851	25,904
2	LSP-US Test Data	10.4	2.07	744	6458
3	LSP-US Synthetic Data	10.4	2.01	728	6081
4	LSP-US Test Data/UR	0.89	0.60	0.40	0.25
5	System Reference (SR)	14.0	3.01	1507	23,877
6	UBAR-E	13.3	2.04	424	4214
7	UBAR-E/SR	0.95	0.68	0.28	0.18

Table 5.3 Results of assessing the lexical richness of LSP-US, including a comparison with UBAR-E. "A. Len." refers to the average number of tokens in the system responses, "CBE" stands for Conditional Bigram Entropy, "#U1" refers to the number of distinct uni-grams, and "#U3" refers to the number of distinct tri-grams. User Reference and System Reference refer to the human-labelled user utterances and system responses, respectively, from the reference corpus for MultiWOZ 2.2. LSP-US Test Data (line 2) is assessed using the MultiWOZ test data, where the human-labelled *system responses* are used (effectively the inverse of the process used to assess UBAR-E and related models in Table 5.1 where the human-labelled *user utterances* are used). LSP-US Synthetic Data (line 3) represents the average of six different evaluations using synthetic data. The user utterances for each of the six evaluations are generated through the interaction of LSP-US and UBAR-E over 1000 dialogues. Each of these synthetic dialogues are driven by a randomly sampled goal from Convlab-2 (Zhu et al., 2020). The results from UBAR-E on line 6 are from Table 5.1.

These results are encouraging in that they suggest the loss of lexical richness in LSP-US is comparable to UBAR-E. They could also be interpreted as a substantial improvement if you consider the additional diversity represented by the number of distinct uni-grams and tri-grams (and the similar resulting CBE values).

In line 3 we see that the results obtained from evaluating LSP-US on synthetically generated dialogues are comparable to the results we obtain from evaluating LSP-US on the dialogues in the test set on line 2 (with human labelled system responses). We do observe the CBE, #U1, and #U3 are all slightly lower on line 3 compared to line 2. However, this makes sense as the system responses that LSP-US is interacting with are significantly less diverse for the synthetic data compared to the test data, due to the fact they come from UBAR-E (line 6), rather the System Reference corpus (line 5). The fact the results from the synthetic data are only slightly lower than the test data is encouraging, as it suggests that LSP-US is able to achieve relatively strong lexical richness compared to UBAR-E on *synthetic data* generated through self-play.

We present an example of LSP-US interacting with a test dialogue in Figure 5.1. We see that the responses are relatively similar, and it is interesting that in this case the dialogues

LSP-US generates aren't notably less fluent or diverse than the human labelled responses. We even observe an example where LSP-US produces an arguably more fluent utterance, in turn 4.

Overall, these results suggest that we have built a relatively diverse User Simulator with comparable or greater lexical richness than UBAR-E and related models presented in Table 5.1. As a future step, it would be valuable to evaluate the *accuracy* of the User Simulator using a framework such as the GCDF1 Score developed by Coca et al. (2021), to see if the accuracy of LSP-US is also comparable to UBAR-E.

5.4 Evaluating E2E TOD systems on MultiWOZ

5.4.1 UBAR Evaluation vs. Official Evaluation

In this section we compare the UBAR evaluation scripts released by Yang et al. (2020) and the official evaluation scripts released by Nekvinda and Dušek (2021). The official evaluation scripts are used in the reported results on the MultiWOZ GitHub repository (Budzianowski et al., 2022). Table 5.4 contains a comparison of 6 different models from Table 5.1, using the two different evaluation scripts.

	Model	Official Evaluation				UBAR Evaluation			
		BLEU	Inform	Success	Score	BLEU	Inform	Success	Score
1	UBAR best	17.6	86.4	73.4	97.5	13.4	**90.2**	**77.0**	97.0
2	UBAR best ITD	16.9	86.8	72.5	96.6	12.7	89.4	75.3	95.1
3	UBAR-E	17.8	**86.9**	**75.4**	**99.0**	15.3	88.1	76.8	97.7
4	UBAR-E ITD	16.7	86.7	73.3	96.7	14.2	87.6	75.5	95.8
5	UBAR-E Syn. Data A.	17.0	81.0	67.9	91.4	14.5	83.7	70.5	91.6
6	UBAR (GPT-2)	**18.2**	85.0	72.0	96.8	**15.6**	89.2	76.3	**98.4**
7	UBAR reported	-	-	-	-	17.0	95.4	80.7	105.1

Table 5.4 Comparison of the official evaluation scripts and the UBAR evaluation scripts. See Table 5.2 for descriptions of the first six models referenced. "UBAR reported" on line 7 refers to the results published in the UBAR paper (Yang et al., 2020) for their best E2E model. We see a consistent trend of higher BLEU scores and lower Inform and Success scores for the "Official Evaluation" scripts compared to the UBAR evaluation scripts.

We observe that the official evaluation scripts produce BLEU scores that are consistently significantly higher, and Inform and Success rates that are consistently lower, than those produced using the UBAR evaluation scripts. We also see that while the overall Scores are relatively close for both evaluation types, the differences are substantial enough that one can draw very different conclusions depending on which system is used. For example, using the

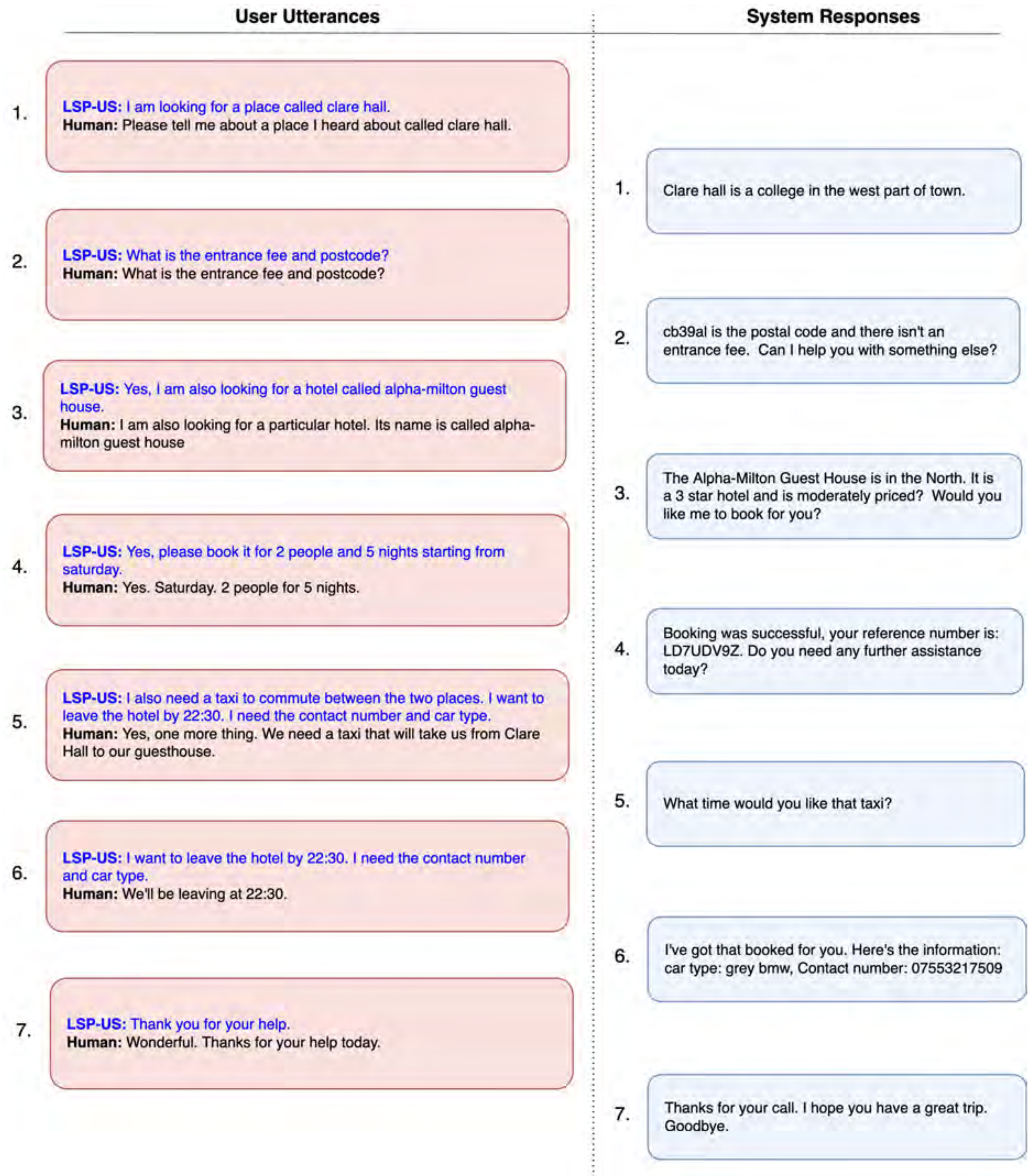


Fig. 5.1 Example dialogue "pmul2179" from the MultiWOZ test set containing three domains, with user utterances from LSP-US and human labellers. We note that despite there being three different domains covered in a relatively short number of turns, LSP-US successfully finishes the dialogue with comparable fluency to the human-labelled utterances.

official scripts, UBAR-E is the best performing model overall, however, with the UBAR scripts UBAR-E GPT-2 is the best performing model. Further, the systems are in disagreement on the best systems for the Inform and Success rates, and are only in agreement that UBAR GPT-2 has the highest BLEU score.

These discrepancies occur due to the two evaluation scripts using different scripts for BLEU scoring, along with a more lenient implementation of the Inform and Success rates by UBAR compared to the official scripts. For BLEU scoring, the official evaluation scripts use the Sacre-BLEU package released by Post (2018) which has become a de-facto standard in the field of machine translation since its release (Nekvinda and Dušek, 2021). Whereas the UBAR evaluation scripts use the GentScorer interface from Wen (2016) which was released prior to Sacre-BLEU and uses a different version of BLEU.

These inconsistencies demonstrate the importance of having a unified, official way of evaluating these systems. We discuss this further in Section 5.4.3 below.

5.4.2 Published UBAR Results

Looking at line 7 in Table 5.4 we see the reported results for the E2E version of UBAR (from the paper) are *significantly* higher than the results we find when using their best model, and their evaluation code (both of which were published by Yang et al. (2021)). We also note that the reported Inform rate of 95.4 in the UBAR paper is higher than the reference corpus for the improved MultiWOZ 2.2 dataset, as shown in Table 5.1, which is unlikely with supervised learning alone. Moreover, in the paper there is no mention that their E2E implementation uses GT turn domains[3]. This is important to note as it means the implementation they use is unable to interact with unseen dialogues and can only work with the human-labelled MultiWOZ data. In summary, our re-implementation suggests the results published in the UBAR paper for their E2E model are very likely an overestimate. We would like to acknowledge and thank Yang et al. (2020) for open-sourcing their code and their model, without which this research would not have been possible.

5.4.3 MultiWOZ Evaluation Improvements

Nekvinda and Dušek (2021) provided a highly valuable contribution to the TOD system research field with the standardised evaluation scripts for MultiWOZ that they developed. As we show in Section 5.4.1, differences in the way metrics are reported makes it challenging to compare models, and the official MultiWOZ scripts have made a big difference to this. However, as Nekvinda and Dušek (2021) note in their paper, there are still a range of significant challenges

[3] This implementation detail is present in their published code.

with the core metrics currently used for evaluating the performance of MultiWOZ. One of the major issues is the inclusion of the BLEU score in the *Score* metric, which is used as the primary measure of the performance of a given model (where $Score = BLEU + ((Inform + Success)/2)$). As we discuss in Section 4.2, the BLEU score is a flawed metric for the fluency of TOD systems. We suggest that the BLEU score should be removed from this equation, and replaced with a combination of more nuanced and appropriate metrics that assess the lexical richness of the system responses—such as those used in the official evaluation scripts presented by Nekvinda and Dušek (2021).

Another limitation with the current way the Score is calculated is it doesn't fully account for whether the model can appropriately deal with changing constraints from the user (e.g. the time and day or number of people for a restaurant booking). UBAR and other E2E TOD systems, including MinTL (Lin et al., 2020) and DAMD (Zhang et al., 2020a), use a *reduced search* database implementation that ignores all additional search constraints if a venue name or train ID is present. This simplifies the E2E model as it doesn't have to be able to deal with the situation where: 1) the user provides a set of constraints, 2) the model provides an entity that matches those constraints, 3) the user provides an additional constraint that invalidates that entity. In this scenario, UBAR and other models that use reduced search will assume the booking is possible, no matter what the additional constraint is, as an entity has already been provided. Ideally, a model would alert the user that the additional constraint (e.g. time of booking) is not available, and ask them to provide another option. This requires a more complex system design, but results in a significantly more useful system in these scenarios.

To illustrate what happens in these scenarios, we provide an example dialogue from the MultiWOZ test set, "mul2376", in Figure 5.2. In turn 5 the human-labelled response recognises that the additional constraint the user has provided is not available and asks for an alternative. However, we see that UBAR-E informs the user the booking was successful (when it was actually impossible), due to its use of reduced search. Under the current implementation of the Score for MultiWOZ models this significant mistake would not be penalised[4] as the model proceeds to update its constraints to the alternative time provided by the user—which is then picked up as the correct slot-value pair at the end of the task. A potential way to fix this would be to update the Inform metric to be assessed for *every turn*, rather than simply checking if the *last* offered entity for each domain fulfils the constraints in the goal.

An additional example of a mistake UBAR-E makes but is not penalised for[5] in the example dialogue shown in Figure 5.2, is in turn 2, where UBAR-E provides the phone number to the user even though the user didn't acknowledge UBAR-E's offer to provide it. This is awkward

[4] With the exception that the BLEU score would be decreased slightly due to this, and this is an example of where the BLEU score is actually a useful metric.

[5] Except for a small penalty via. the BLEU score, as with the previous footnote.

Fig. 5.2 Example dialogue "mul2376" containing two domains, with system responses from human labellers, UBAR-E (line 3 in Table 5.1), and UBAR-E refined with PPO Transformer RL (PPO TRL) (line 9 in Table 5.1). PPO TRL is refined on 900 dialogues using randomly sampled goals. We note that in turn 5 UBAR-E confirms a booking for a time when the restaurant is not open (discussed in Section 5.4.3). We also note that the PPO TRL model uses twice as many slots as UBAR-E (16 compared to 8) (discussed in Section 5.5).

and suboptimal, and would ideally be penalised through a Score that penalised the provision of additional, unnecessary slot-value pairs[6].

In summary, we believe that a more nuanced and appropriate measurement of model performance is needed. We suggest two main ways this could be improved: 1) by swapping the BLEU score for more nuanced lexical richness metrics, and 2) by improving the way the Inform rate is calculated to incorporate turn-level belief states. After all, *what is measured is what is optimised for*, and if the Inform rate was calculated according to our proposal then the reduced search implementation may not have been chosen by Yang et al. (2020) and others.

5.5 PPO Transformer Reinforcement Learning

In this section we discuss our initial results from our PPO Transformer RL training regime (Section 5.5.1), and the implications they have for improving our reward signal (Section 5.5.2).

5.5.1 Initial Results

We experiment with refining UBAR-E using PPO Transformer RL, and find encouraging preliminary results despite the instability in training outlined in Section 4.4. Line 9 of Table 5.1 presents the results of refining UBAR-E on 900 dialogues using randomly sampled goals. We see that our reward modules (outlined in Section 3.3.1) are effective at improving the Inform rate, pushing it from 86.9% to 90.5%—within 4% of the reference Inform rate of 93.7%. However, we see that the Success rate drops a few percentage points, and more significantly, the BLEU score drops from 17.8 to 11.5.

We believe there are two phenomena that explain these results. First, the increased Inform rate can be attributed to the model learning to generate more accurate belief states. It makes sense that this occurs because of the negative reward the model receives when it misses (or misinterprets) information that the US has provided. Second, the drop in the BLEU score can be attributed to the model learning to provide significantly more information to the user than it needs to. It does this as it has learnt that it receives a negative reward when it fails to provide the appropriate information the user *requests*. Because there is no penalty on the length of the generated text, a way to address this is to provide significantly more information than is necessary. We see this in the example dialogue in Figure 5.2 where the average response length for the PPO TRL model is longer than UBAR-E on average (18.5 compared to 15.4 average words per turn), and the PPO TRL model provides *twice* as many slots as UBAR-E throughout the dialogue (16 compared to 8). A clear way to address this second phenomena,

[6]We recognise that such a scheme is non-trivial, though we believe such a scheme could be worth developing.

and the negative effect it has on the BLEU score, is to add a small negative reward for every word that is generated. If calibrated correctly (a non-trivial problem), this would encourage the model to provide concise responses containing the relevant information only.

Further, we see that the PPO TRL model has significantly higher results than UBAR-E on the lexical richness metrics. However, if we account for the average number of tokens in the system responses (by dividing the scores by their respective A. Len.), we see that our results do not suggest an increase in lexical richness. We get a lower CBE for the PPO TRL model, a similar #U1, and a higher #U3. This suggests the diversity of the responses are not well distributed. It is also in line with our empirical finding that the model is over-providing information in its responses. Our suggestion above of adding a per-token negative reward should address this.

While these results present a lot to be improved, they also present a case for the PPO Transformer RL approach we propose. The improved Inform rate gives us reason to believe the regime is working as intended in that the updated model outputs appear consistent with the reward signals we have provided, despite the training instability described in section 4.4. Along with addressing this instability, another important next step is to improve the current reward modules.

5.5.2 Reward Modules

There are a number of limitations in the design of the reward modules outlined in Section 3.3.1. First, as outlined in the previous section, the model currently has an incentive to produce overly verbose responses. A negative per-token reward should be added to encourage concise responses. Second, there is currently no reward associated with the fluency, or lexical richness, of the generated system utterances. This is a very important part of what makes a successful (useful) system response for a real-world TOD system, and therefore it should be explicitly optimised to ensure the model doesn't drift too far from "fluent" generation.

These two specific issues allude to a more general challenge in our approach of defining the reward function in a hand-crafted, rule-based paradigm. We believe that the reward signal is insufficient to capture the intended (very complex) generation behaviour of the fully E2E model, and it is under-specified. This a particularly difficult challenge to address as we effectively need to provide a sufficiently well-specified reward signal for all four of the "tasks" UBAR is performing in each turn: (NLU, DST, DP, and NLG). If we under-specify even one of these, we risk the generation of the model "drifting" from the intended behaviour. In Section 6.2, we propose an improved regime that addresses these challenges.

5.5.3 Future Work

In the final section of this chapter we outline three improvements that could be made to our PPO Transformer RL implementation, and are extensions we would implement if we had more time to continue this research.

First, the training procedure should be parallelised to enable more efficient experimentation. Second, the instability in the KL divergence that we outline in Section 4.4 should be addressed. We believe this can be done by updating our implementation to use just the Clipped Surrogate Objective (Schulman et al., 2017), with no KL penalty. Third, our implementation should be adapted to *jointly* optimise both LSP-US and UBAR-E. Tseng et al. (2021) demonstrated the power of such an approach. Moreover, we rely on an assumption we know is flawed in our optimisation of UBAR-E, where we keep LSP-US static: we assume that LSP-US is perfect. That is, we assume the information in the user act of LSP-US appropriately incorporates the goal state and the previous system response, and that the generated user utterance correctly communicates the information in the user act. We know that this is not true through empirical analysis (a valuable extension to this work would be to focus on evaluating LSP-US quantitatively). A joint optimisation paradigm doesn't make any such assumptions, and has the major promise of improving both systems simultaneously.

Finally, it would be beneficial to utilise a pre-trained *conversation* model as the "backbone" of UBAR-E and LSP-US (as opposed to Distil-GPT2). These models are trained on large corpora of dialogue data that is specifically conversational in nature, and the use of such models has recently been shown to be a very promising approach to training E2E Transformer-based TOD systems (as we discuss in Section 2.2.2).

Chapter 6

Conclusion

6.1 Summary

In this thesis we explore the research goal: can we build a fully E2E Transformer-based TOD system *and* User Simulator, that are good enough to be used to drive joint optimisation techniques such as those outlined in JOUST (Tseng et al., 2021)? We have developed and evaluated UBAR-E and LSP-US, and have demonstrated that these two systems perform well individually, and that together they form a strong foundation for joint optimisation experiments through self-play.

We outline the design of our E2E Transformer-based TOD system, UBAR-E, an extension of UBAR that enables it to work on unseen dialogues by using *Inferred Turn Domains* as opposed to ground-truth turn-domains. We also present our implementation of lexicalisation, the other key extension to make UBAR fully E2E. We show that UBAR-E achieves similar performance to UBAR across all MultiWOZ metrics, including lexical richness, despite it not having access to ground-truth turn-domains. We also show that UBAR-E sets a very high baseline performance for subsequent work to improve.

We outline the novel design of our E2E Transformer-based User Simulator, LSP-US. As far as we know, LSP-US is the first E2E Transformer-based User Simulator. We show that LSP-US achieves comparable or greater lexical richness than UBAR-E on the MultiWOZ dataset. Further to the development and analysis of our two core models, we also perform a number of experiments utilising synthetic data generated through self-play between the two models. We find that the approaches used are not effective, and conclude that RL approaches have higher promise.

We analyse differences in the evaluation scripts in the published UBAR code (Yang et al., 2021) and the official MultiWOZ evaluation scripts (Nekvinda and Dušek, 2021). We also analyse the E2E results in the UBAR paper (Yang et al., 2020) and conclude that they are not consistent with other evaluation frameworks. We then discuss the current evaluation criteria used for the MultiWOZ dataset and suggest a number of improvements.

Finally, we develop an RL regime using Proximal Policy Optimisation, PPO Transformer RL, to refine our two models. We present encouraging preliminary results using PPO Transformer RL to refine UBAR-E and outline future work to improve our implementation.

6.2 Future Work

In this thesis we have developed an E2E Transformer-based TOD system and User Simulator, which together form a strong foundation for joint optimisation experiments through self-play. In this final section we summarise what we believe to be the highest-value approach to jointly optimise our two models, and thus the highest-value future direction for this research. As we have outlined, applying RL to E2E Transformer-based systems like UBAR-E and LSP-US is very challenging. We have discussed how one must effectively target all four "components" that the Transformer is modelling (NLU, DST, DP, and NLG) with the reward signal.

We suggest an alternative where we draw inspiration from the pre-Transformer, *pipelined* approach to building E2E TOD systems made of four modules, each performing a different task. We propose using this approach, but with a fine-tuned LLM for *each* of the respective modules. This would enable an adapted implementation of PPO Transformer RL (along with the improvements outlined in Section 5.5.3) to be used that focuses specifically on the Dialogue Policy model. Tseng et al. (2021) demonstrated the power of using RL specifically to jointly optimise the Dialogue Policy modules across both their E2E TOD system and User Simulator. We suggest effectively replicating this but with Transformers instead of LSTMs for text generation at each step.

Using RL to optimise the Dialogue Policy also has the advantage that there is already a mature body of research demonstrating its effectiveness (as outlined in Section 2.4.1). Finally, and critically, this approach enables us to continue to leverage the power of the Transformer architecture, while also leveraging a proven RL approach that is much less likely to lead to an under-specified reward signal[1].

In summary, whichever direction this rapidly developing research field goes in next, we can be fairly confident Transformers will be a part of it—at least for some time. Transformers seem to be taking over the entire field of dialogue systems, and as the now famous paper says: *Attention is all you need.*

[1]We note that this approach does have the downside of requiring four fine-tuned large language models which presents increased challenges for training time and latency when deployed. However, with powerful compressed models like Distil-GPT2 we believe this is not a limiting factor.

References

Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. Towards a human-like open-domain chatbot. *CoRR*, abs/2001.09977, 2020. URL https://arxiv.org/abs/2001.09977.

J Alammar. The illustrated transformer. Github, 2018. URL https://jalammar.github.io/illustrated-transformer/.

J Alammar. The illustrated gpt-2 (visualizing transformer language models). Github, 2019. URL https://jalammar.github.io/illustrated-gpt2/.

Layla El Asri, Jing He, and Kaheer Suleman. A sequence-to-sequence model for user simulation in spoken dialogue systems. *CoRR*, abs/1607.00070, 2016. URL http://arxiv.org/abs/1607.00070.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

Vevake Balaraman, Seyedmostafa Sheikhalishahi, and Bernardo Magnini. Recent neural methods on dialogue state tracking for task-oriented dialogue systems: A survey. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 239–251, Singapore and Online, July 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.sigdial-1.25.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, mar 2003. ISSN 1532-4435.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

Paweł Budzianowski and Ivan Vulić. Hello, it's GPT-2 - how can I help you? towards the use of pretrained language models for task-oriented dialogue systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 15–22, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5602. URL https://aclanthology.org/D19-5602.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. Multiwoz – a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling, 2018. URL https://arxiv.org/abs/1810.00278.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. Multiwoz. Github, 2022. URL https://github.com/budzianowski/multiwoz.

Iñigo Casanueva, Paweł Budzianowski, Stefan Ultes, Florian Kreyssig, Bo-Hsiang Tseng, Yen-chen Wu, and Milica Gašić. Feudal dialogue management with jointly learned feature extractors. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 332–337, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5038. URL https://aclanthology.org/W18-5038.

Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. Clustering behaviors of spoken dialogue systems users. page 4 pages, 03 2012. doi: 10.1109/ICASSP.2012.6289038.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *SIGKDD Explor. Newsl.*, 19(2):25–35, nov 2017. ISSN 1931-0145. doi: 10.1145/3166054.3166058. URL https://doi.org/10.1145/3166054.3166058.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. *CoRR*, abs/1905.12866, 2019. URL http://arxiv.org/abs/1905.12866.

Alexandru Coca, Bo-Hsiang Tseng, and Bill Byrne. GCDF1: A goal- and context- driven F-score for evaluating user models. In *The First Workshop on Evaluations and Assessments of Neural Conversation Systems*, pages 7–14, Online, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eancs-1.2. URL https://aclanthology.org/2021.eancs-1.2.

Paul A. Crook and Alex Marin. Sequence to sequence modeling for user simulation in dialog systems. In *INTERSPEECH*, 2017.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *ArXiv*, abs/1901.02860, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

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

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49, Saarbrücken, Germany, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5506. URL https://aclanthology.org/W17-5506.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1154. URL https://aclanthology.org/P16-1154.

Izzeddin Gür, Dilek Hakkani-Tür, Gokhan Tür, and Pararth Shah. User modeling for task oriented dialogues. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 900–906, 2018. doi: 10.1109/SLT.2018.8639652.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.54. URL https://aclanthology.org/2020.acl-main.54.

Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, Jian Sun, and Yongbin Li. GALAXY: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. *CoRR*, abs/2111.14592, 2021. URL https://arxiv.org/abs/2111.14592.

Matthew Henderson, Ivan Vulic, Iñigo Casanueva, Pawel Budzianowski, Daniela Gerz, Sam Coope, Georgios Spithourakis, Tsung-Hsien Wen, Nikola Mrksic, and Pei-Hao Su. Polyresponse: A rank-based approach to task-oriented dialogue with application in restaurant search and booking. *CoRR*, abs/1909.01296, 2019. URL http://arxiv.org/abs/1909.01296.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. A simple language model for task-oriented dialogue. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/e946209592563be0f01c844ab2170f0c-Paper.pdf.

Yutai Hou, Meng Fang, Wanxiang Che, and Ting Liu. A corpus-free state2seq user simulator for task-oriented dialogue. *CoRR*, abs/1909.04448, 2019. URL http://arxiv.org/abs/1909.04448.

Youngsoo Jang, Jongmin Lee, and Kee-Eung Kim. GPT-critic: Offline reinforcement learning for end-to-end task-oriented dialogue systems. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=qaxhBG1UUaS.

Florian Kreyssig, Iñigo Casanueva, Paweł Budzianowski, and Milica Gašić. 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, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5007. URL https://aclanthology.org/W18-5007.

Wai-Chung Kwan, Hongru Wang, Huimin Wang, and Kam-Fai Wong. A survey on recent advances and challenges in reinforcement learning methods for task-oriented dialogue policy learning, 2022. URL https://arxiv.org/abs/2202.13675.

Sungjin Lee, Qi Zhu, Ryuichi Takanobu, Xiang Li, Yaoqin Zhang, Zheng Zhang, Jinchao Li, Baolin Peng, Xiujun Li, Minlie Huang, and Jianfeng Gao. Convlab: Multi-domain end-to-end dialog system platform. *CoRR*, abs/1904.08637, 2019. URL http://arxiv.org/abs/1904.08637.

Yohan Lee. Improving end-to-end task-oriented dialog system with a simple auxiliary task. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1296–1303, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.112. URL https://aclanthology.org/2021.findings-emnlp.112.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *ACL (1)*, pages 1437–1447, 2018. URL https://aclanthology.info/papers/P18-1133/p18-1133.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *Speech and Audio Processing, IEEE Transactions on*, 8:11 – 23, 02 2000. doi: 10.1109/89.817450.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019. URL http://arxiv.org/abs/1910.13461.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *CoRR*, abs/1606.01541, 2016. URL http://arxiv.org/abs/1606.01541.

Tianda Li, Yassir El Mesbahi, Ivan Kobyzev, Ahmad Rashid, Atif Mahmud, Nithin Anchuri, Habib Hajimolahoseini, Yang Liu, and Mehdi Rezagholizadeh. A short study on compressing decoder-based language models, 2021. URL https://arxiv.org/abs/2110.08460.

Weixin Liang, Youzhi Tian, Chengcai Chen, and Zhou Yu. MOSS: end-to-end dialog system framework with modular supervision. *CoRR*, abs/1909.05528, 2019. URL http://arxiv.org/abs/1909.05528.

Qin Libo, Li Zhouyang, Lou Jieming, Yu Qiying, and Che Wanxiang. A survey of natural language generation in task-oriented dialogue system. 36(1):1, 2022. URL http://jcip.cipsc.org.cn/EN/abstract/article_3241.shtml.

Hsien-chin Lin, Nurul Lubis, Songbo Hu, Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, and Milica Gasic. 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, July 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.sigdial-1.47.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. MinTL: Minimalist transfer learning for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3391–3405, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.273. URL https://aclanthology.org/2020.emnlp-main.273.

Bing Liu and Ian R. Lane. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. *CoRR*, abs/1709.06136, 2017. URL http://arxiv.org/abs/1709.06136.

Bing Liu, Gökhan Tür, Dilek Hakkani-Tür, Pararth Shah, and Larry P. Heck. End-to-end optimization of task-oriented dialogue model with deep reinforcement learning. *CoRR*, abs/1711.10712, 2017. URL http://arxiv.org/abs/1711.10712.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1230. URL https://aclanthology.org/D16-1230.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hyg0vbWC-.

Shikib Mehri, Mihail Eric, and Dilek Z. Hakkani-Tür. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. *ArXiv*, abs/2009.13570, 2020.

Teruhisa Misu and Tatsuya Kawahara. Speech-based interactive information guidance system using question-answering technique. volume 4, pages IV–145, 05 2007. ISBN 1-4244-0728-1. doi: 10.1109/ICASSP.2007.367184.

Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. Neural belief tracker: Data-driven dialogue state tracking. *CoRR*, abs/1606.03777, 2016. URL http://arxiv.org/abs/1606.03777.

Tomáš Nekvinda and Ondřej Dušek. Shades of bleu, flavours of success: The case of multiwoz, 2021. URL https://arxiv.org/abs/2106.05555.

Xinrui Nie, Zehao Lin, Xinjing Huang, and Yin Zhang. Graph neural net-based user simulator. In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings*, page 638–650, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN 978-3-030-32380-6. doi: 10.1007/978-3-030-32381-3_51. URL https://doi.org/10.1007/978-3-030-32381-3_51.

Alexandros Papangelis, Yi-Chia Wang, Piero Molino, and Gokhan Tur. Collaborative multi-agent dialogue model training via reinforcement learning. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 92–102, Stockholm, Sweden, September 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5912. URL https://aclanthology.org/W19-5912.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. Composite task-completion dialogue system via hierarchical deep reinforcement learning. *CoRR*, abs/1704.03084, 2017. URL http://arxiv.org/abs/1704.03084.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. SOLOIST: few-shot task-oriented dialog with A single pre-trained auto-regressive model. *CoRR*, abs/2005.05298, 2020. URL https://arxiv.org/abs/2005.05298.

O. Pietquin and T. Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):589–599, 2006. doi: 10.1109/TSA.2005.855836.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6319. URL https://aclanthology.org/W18-6319.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019. URL http://arxiv.org/abs/1910.10683.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. Recipes for building an open-domain chatbot. *CoRR*, abs/2004.13637, 2020. URL https://arxiv.org/abs/2004.13637.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL http://arxiv.org/abs/1910.01108.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 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, April 2007. Association for Computational Linguistics. URL https://aclanthology.org/N07-2038.

Konrad Scheffler and Steve Young. Corpus-based dialogue simulation for automatic strategy learning and evaluation, 2001.

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

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808, 2015. URL http://arxiv.org/abs/1507.04808.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. *CoRR*, abs/1506.06714, 2015. URL http://arxiv.org/abs/1506.06714.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. Multi-task pre-training for plug-and-play task-oriented dialogue system. *CoRR*, abs/2109.14739, 2021. URL https://arxiv.org/abs/2109.14739.

Haipeng Sun, Junwei Bao, Youzheng Wu, and Xiaodong He. Bort: Back and denoising reconstruction for end-to-end task-oriented dialog, 2022. URL https://arxiv.org/abs/2205.02471.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

Ryuichi Takanobu, Runze Liang, and Minlie Huang. Multi-agent task-oriented dialog policy learning with role-aware reward decomposition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 625–638, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.59. URL https://aclanthology.org/2020.acl-main.59.

Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. Transferable dialogue systems and user simulators. *CoRR*, abs/2107.11904, 2021. URL https://arxiv.org/abs/2107.11904.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Leandro von Werra. Transformer reinforcement learning. Github, 2022. URL https://github.com/lvwerra/trl.

Tsung-Hsien Wen. Rnnlg. Github, 2016. URL https://github.com/shawnwun/RNNLG.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 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, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1199. URL https://aclanthology.org/D15-1199.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 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*, pages 438–449, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-1042.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019. URL http://arxiv.org/abs/1910.03771.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. Transferable multi-domain state generator for task-oriented dialogue systems. *CoRR*, abs/1905.08743, 2019. URL http://arxiv.org/abs/1905.08743.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. UBAR: towards fully end-to-end task-oriented dialog systems with GPT-2. *CoRR*, abs/2012.03539, 2020. URL https://arxiv.org/abs/2012.03539.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. Ubar. Github, 2021. URL https://github.com/TonyNemo/UBAR-MultiWOZ.

Steve J. Young, Milica Gasic, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Comput. Speech Lang.*, 24:150–174, 2010.

Steve J. Young, Milica Gasic, Blaise Thomson, and J. Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101:1160–1179, 2013.

Yichi Zhang, Zhijian Ou, and Zhou Yu. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *AAAI*, 2020a.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online, July 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.30. URL https://aclanthology.org/2020.acl-demos.30.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. 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 1208–1218, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1123. URL https://aclanthology.org/N19-1123.

Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. Convlab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems. *CoRR*, abs/2002.04793, 2020. URL https://arxiv.org/abs/2002.04793.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593, 2019. URL http://arxiv.org/abs/1909.08593.