Building a Conversational User Simulator using GANs

Lucia Lopez Rivilla

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

St Edmund's College

August 2022

Declaration

I, Lucia Lopez Rivilla of St Edmund's 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.

All software is implemented in Python using the PyTorch library. The code for the LSTM model was built on the software developed on the previous MPhil project hosted by Toshiba (Dockes, 2021). For the transformer approach, all pre-trained model implementations, training and decoding tools were retrieved from the HuggingFace Library[1].

BLEU and ROUGE evaluation metrics are retrieved from the NLTK[2] package and PyPi ROUGE[3] package respectively.

Word count: 14,955

Lucia Lopez Rivilla
August 2022

[1]https://huggingface.co
[2]https://www.nltk.org/_modules/nltk/translate/bleu_score.html
[3]https://pypi.org/project/rouge-score/

Acknowledgements

First and foremost I would like to thank my supervisors Dr. Simon Keizer and Dr. Kate Knill for their continuous guidance and support. Your input throughout the course of this project has been invaluable. I would also like to extend my gratitude to Dr. Svetlana Stoyanchev and Dr. Rama Doddipatla for their helpful suggestions, and to Toshiba for providing me with the needed resources to complete this project.

Special thanks to DeepMind and the Cambridge Trust for giving me the opportunity to study in the University of Cambridge by funding this course.

Lastly and above all, I am deeply grateful to my family for their love and unconditional support, and to my coursemates and college family for walking this path with me and being my rock this whole year.

Abstract

User simulators have long been used to succesfully train, test and evaluate policies of reinforcement learning based dialogue systems. For this task, numerous types of simulators can be deployed (rule-based, data-driven, with different input/output forms, etc).

In this context, this dissertation presents a novel approach to user simulation, focused on developing a conversational simulator for task-oriented dialogue systems using Generative Adversarial Networks (GANs). The motivation behind training the simulator in an adversarial style, instead of with the standard maximum likelihood, resides in the possibility of achieving a simulator that generates more diverse output and does not aim to exactly replicate the training corpus, which we believe would be beneficial for training dialogue systems' policies.

We build up on work carried out in a previous MPhil project hosted by Toshiba (Dockes, 2021). This previous system targeted GAN-based user simulation at the semantic or abstract level of dialogue acts and results suggest that this training approach could benefit policy training. Despite its promising results, we believe adversarial training can bring greater gains to a simulator generating at the word level. In consequence, we expand the simulator to output utterances directly. Starting from a base LSTM encoder-decoder architecture, we experiment with the attention mechanism, a multitask scheme to jointly generate semantics and utterances, and adversarial training with several modifications. We evaluate our simulator on corpus-based metrics, achieving results comparable to a model trained with maximum likelihood, and with human evaluation, which shows that the resultant utterances have an acceptable level of naturalness and coherence. Lastly, we design and carry out initial experiments in a transformer-based natural language to natural language user simulator, which makes use of the Large Pre-trained Language Models GPT-2 and RoBERTa.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

The development of better dialogue systems for end-users has long been a goal of human-computer interaction research, given its promising potential and commercial appeal. Examples of such systems include the well-known Siri, Alexa or Cortana, to whom users can speak, give instructions or ask for help.

Depending on their applications, dialogue systems can be classified in two groups: task-oriented and non-task-oriented dialogue systems (the latter also known as chat-bots) (Chen et al., 2017). In the present work we focus on task-oriented systems, which are designed to aid users to accomplish a particular task, like hotel search or ticket booking, and are, thus, constrained to a specific domain and structure.

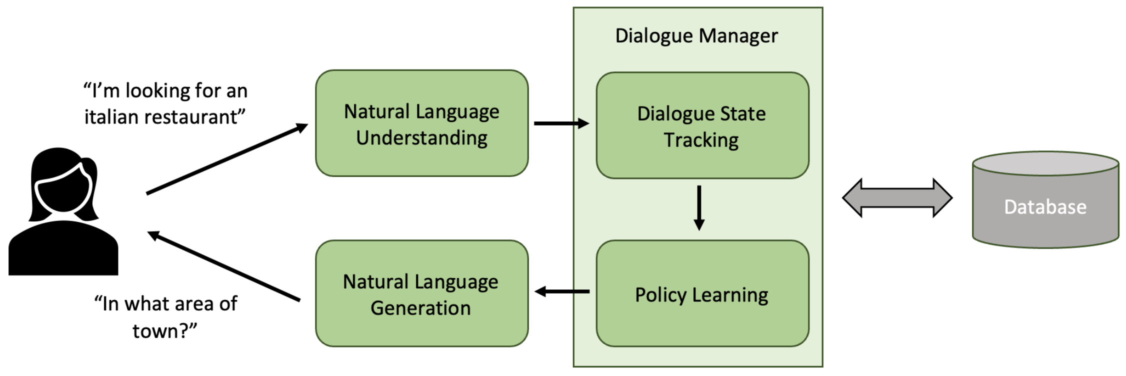

"I'm looking for an italian restaurant"
"In what area of town?"
Natural Language Understanding
Natural Language Generation
Dialogue Manager
Dialogue State Tracking
Policy Learning
Database

Fig. 1.1 Typical pipeline of a task-oriented dialogue system.

Task-oriented dialogues (TOD) have received a lot of attention in dialogue research (Eshky, 2014). This is in part because they contain a clear success indicator in terms of how well the task was completed. Additionally, TOD systems are more directly applicable to commercial applications, making them an attractive and quickly expanding field of study.

The wide approach to these systems is based on a pipeline composed of several modules as seen in Figure 1.1. The natural language understanding (NLU) component maps user utterances to their semantic representation in the form of dialogue acts. Following this, in the dialogue manager the state of the dialogue is tracked and any needed information is retrieved from knowledge databases. Given this state, the next action of the system is chosen at each turn, according to a learned policy. Finally, the system's action is transformed back to the surface form in the natural language generation (NLG) module.

All four of the aforementioned components have been the subject of extensive research, but the component that is most pertinent to the problem at hand is the dialogue manager. Its ultimate goal is to solve tasks as efficiently as possible, for which an optimal policy must be learned. In state-of-the-art systems reinforcement learning (RL) is used for this purpose, as it allows online learning with a user rather than supervised learning from a static dialogue corpus. However, RL requires training with an extensive number of dialogue interactions. As a result, learning from scratch through interactions with real users becomes infeasible.

Another alternative consists of using a fixed corpus to train the RL agent, but this results in little exploration of the policy state-space, given the stationarity of the data. To overcome these drawbacks, user simulators started to gain popularity and have now become a well-established tool for automatic dialogue management design and evaluation.

The aim of a user simulator is to mimic human behaviour in the most natural and rational manner possible, while keeping track of the dialogue history and unobservable aspects like the user's goal and preferences. Moreover, its range of simulated answers must be as broad as possible so that strategies can be learned for different types of user. After an initial training of a policy with a simulator, online learning can be carried out with real users for further optimization.

Among the investigated methods for simulation, which we describe in depth in Chapter 2, the use of Generative Adversarial Networks (GANs) is starting to spark interest. GANs are trained to generate data that is indistinguishable from real samples and, although they were initially proposed for image generation and have seen greater success in this application, there has recently been progress in text generation, a task which will be the focus of our system.

Based on their main application, GANs seem to be a good fit for the challenge of imitating real users. In this training approach a discriminator is introduced, which tries to distinguish synthesized samples from real ones. We hypothesize that the deployment of a GAN-based training approach, rather than the traditional Maximum Likelihood (ML), could lead to a lesser replication of the training corpus, which would be beneficial to train more robust TOD system's policies.

In fact, a previous investigation in GAN-based user simulation (Dockes, 2021) showed that policies trained with the GAN simulator could achieve higher success rates than those obtained with an MLE-trained one, supporting this alternative training.

However, although promising, this previous system was implemented at a semantic level (dialogue acts). This form of output could impose constraints as, in practice, the limited range of possible acts (only 30) could make the output too close to what is observed in the real data. In fact, the dialogue act combination generated is likely to appear in the training corpus.

Following this, adversarial training will likely have higher potential in a simulator generating words rather than semantics, considering the former's greater complexity. Moreover, word-level outputs entail an increased output diversity, something desirable in simulators as it allows greater exploration of the state-space when training dialogue systems. Lastly, the direct use of words frees us from the need of having dialogue act annotations for training, and avoids the need of an additional NLG component to transform semantics into utterances for training a dialogue manager that takes natural language (NL) as input. All these reasons support the proposed modification to a word-level simulator, the focus of this dissertation.

1.2 Contributions

As a first contribution, we have extended the semantic-level simulator from (Dockes, 2021) to word generation, and enhanced its performance by adding an attention mechanism. A second contribution is on multi-task learning for user simulation, which, to the best of our knowledge, has never been tested for our task and configuration. The most outstanding results in this regard are obtained for the case of Maximum Likelihood (ML) training, where we achieve higher scores for word generation by jointly training word and dialogue act generation. Another contribution, perhaps our main one, is the establishment of two methods to successfully train word-level user simulators via adversarial training, designed to avoid traditional issues that arise with GANs, and achieving comparable scores on traditional language generation metrics (BLEU/ROUGE) to the ML baseline, while displaying higher output diversity as given by the self-BLEU score.

Finally, we provide one of the first word-level user simulators that use large pre-trained language models, achieving very high BLEU and ROUGE scores. This transformer based simulator also incorporates adversarial training, featuring a GPT-2 based generator and a

RoBERTa based discriminator. We demonstrate that output diversity can be improved using this framework.

1.3 Overview

The remainder of the dissertation is structured as follows:

In Chapter 2 we revise the literature in user simulation, particularly focusing on deep learning approaches. We also present relevant advances in adversarial training, specifically for text generation.

Chapter 3 describes the different architectures investigated to model the word-level user simulator. We first build up on the previous semantic-level simulator (i.e. an LSTM encoder-decoder) and then move on to transformers, state-of-the-art models in numerous tasks.

Chapter 4 describes the experimental methods and evaluation metrics.

Chapter 5 includes the experiments carried out, along with results in terms of selected evaluation metrics.

Finally, in Chapter 6 we conclude the project and outline future work.

Chapter 2

Background

In the present chapter we cover the meaningful background for our investigation, including an overview of user simulation and its challenges as well as of adversarial training, particularly for the task that concerns us: text generation.

2.1 User Simulation for Task-Oriented Dialogue Systems

2.1.1 Why simulate users?

The purpose of TOD systems is to assist users in completing a specific task in several possible domains, like hotel or restaurant search, ticket booking and so on. In today's world, interaction with these systems, such as Siri or Alexa, continues to increase and, as a result, making them respond more naturally and accurately to users prompts is becoming a main point of focus.

These systems are typically trained through RL (Gašić and Young, 2014; Young et al., 2013a), which requires a considerable amount of interactions to learn a dialogue policy. Training with real users is infeasible given its cost and time consumption, aside from the instability that noisy feedback can introduce in the training process. On the other hand, directly learning from a corpus presents disadvantages like a lack of interactivity and a reduction in the number of dialogue trajectories explored. In this context, user simulation was raised as an alternative approach to automate interaction and to perform an extensive evaluation and optimization of a dialogue system without manual investigation (Eckert et al., 1997).

User simulators are trained on a reference dialogue corpus to generate synthetic utterances. Once a user simulator has been trained, as many dialogues as desired can be performed against a dialogue system, which allows for a more interactive, controllable and efficient training of dialogue policies (see Figure 2.1). Moreover, their dynamic behaviour entails a greater range of policy exploration than a static corpus, whose accessible state-space is limited by the recorded data.

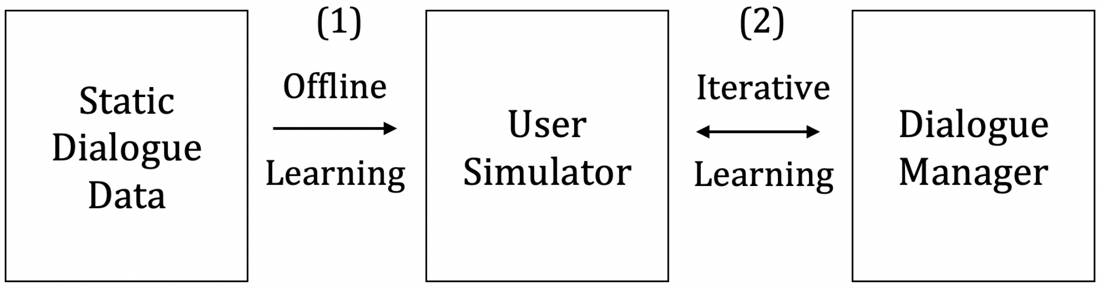


Fig. 2.1 Training scheme of user simulators and dialogue systems.

As one could expect, the performance of the simulator directly impacts the quality of the dialogue policy that is being trained with it (Ai et al., 2007; Schatzmann et al., 2005). Thus, building good user simulators is of the essence. Their aim is to replicate real user behaviour, a task that involves many challenges that have been addressed in the literature, and which continue being investigated. The different approaches to user simulation that have been explored can be classified according to aspects like output granularity (e.g. whether they perform at semantic or word level) and methodology (rule-based, data-driven and deep-learning approaches). As shown in the following sections, there has been a higher tendency towards semantic representation, but at the present time the idea of shifting towards natural language is gaining interest, as directly generating utterances could give outputs of greater diversity which, as previously mentioned, increase policy exploration. In this section, we review the advances that have been made in the field of user simulation since its first appearance.

2.1.2 Early probabilistic approaches

The first user models that were created approached the task as a probabilistic formulation. These models' parameters were either manually tuned or devised from real data, and they operated at the semantic level of dialogue acts. At this level, the aim of user simulators is to ultimately model the distribution over the plausible user acts given the dialogue history:

$$p(u_t|a_t, u_{t-1}, a_{t-1}, u_{t-2}...) \tag{2.1}$$

Where u_t and a_t represent the user and system act respectively. Among these first approaches, work in user simulation was pioneered by Eckert et al. (1997), who used n-gram models to predict the next user act based on previous machine and user utterances. They handcrafted the conditional distributions grounded on intuition and characteristics observed in their corpus, but they restricted the dialogue history to just the previous system act, which is why this model is known as the bigram model.

Later work by the same group of authors, (Levin et al., 2000) expanded on the bigram model to gain a more realistic degree of conventional structure in dialogues, by only allowing "sensible" pairs of user response and system action and setting all other probabilities to zero (Schatzmann et al., 2006).

Although this model had the advantage of being purely probabilistic and domain-independent, it lacked consistency in user behaviour, as it only considered a bigram context. Efforts in (Georgila et al., 2005) tried to overcome this by including higher n-grams, devising a model similar to Eckert's, and thus fully probabilistic, but completely induced from real data. Although their assumption that a broader context could avoid inconsistency in the dialogue acts appears reasonable, this context increase can be problematic due to data sparsity, leading to inevitably undertrained models (Young et al., 2013b).

Later approaches solve this by incorporating the goal explicitly with Bayesian networks (Rossignol et al., 2011), which can model rich conditional dependencies but also suffer from data sparsity. A promising alternative to solve the sparsity issue was presented in (Jung et al., 2009), where they make use of Conditional Random Fields (CRFs) to efficiently model long sequences. Other paths of research used a hidden Markov model (HMM) for user simulation (Cuayahuitl et al., 2005) to generate both user and system actions.

As another alternative to overcome the inconsistency drawback, goal-directed approaches were suggested (Pietquin, 2006; Rieser and Lemon, 2006; Schatzmann et al., 2007; Scheffler and Young, 2000). These are essentially deterministic but introduced trainable variables where user choice was a possibility. Among them, we specifically outline the Agenda-based simulator (Schatzmann et al., 2007), which is still widely deployed and frequently used as baseline for comparisons. This model is described in section 2.1.3. Prior to this simulator, the work by (Scheffler and Young, 2002) set handcrafted rules for actions that depend on user goals, and defined probabilistic parameters to select actions that were goal-independent. They laid out a detailed decision network that maps user behaviours to every scenario. Despite the advance these simulation methods entailed, they require extensive manual effort which can be impractical.

Finally, we outline work carried out by Chandramohan et al. (2011) which uses inverse reinforcement learning (Ng and Russell, 2000) to learn appropriate user's rewards from human-human dialogue interactions. This reward could potentially be used to train a POMPDbased user simulator alongside a dialogue system, by which each system would iteratively refine its policy to maximize this reward.

2.1.3 Agenda-Based Simulator

A simulator still widely used and that has become a baseline for comparisons is the Agenda-Based User Simulator (ABUS) presented in (Schatzmann et al., 2007). Despite its simplicity, it has proven to be effective. Under this approach, the user state, S, is described as an agenda, A, and a goal, G. The goal is composed of several constraints and requests that need to be met. In turn, the agenda consists of a stack-like structure of dialogue acts to be produced for each goal. As the dialogue progresses the agenda and goal are dynamically updated, and user acts are selected from the top of the agenda. Thus, the agenda becomes a convenient tool to track the progress of the dialogue.

Despite its common use, this simulator underperforms under interaction with real users. In fact, two important limitations can be signaled in the ABUS. Firstly, the need to extract the handcrafted and domain-specific rules requires expert knowledge and makes this approach infeasible for complex tasks. Secondly, these handcrafted rules may not capture the complexity of real-user behaviour to its full extent, and their specific design for a particular domain limits transferability. Moreover, rule-based models tend to lack response diversity, and the fact that it generates dialogue acts imposes an even greater limitation on the output variability, something highly desirable in order to create more realistic user behaviour and to exhaustively explore the state-space with RL.

With the aim to create more realistic user behaviour, some of the ABUS parameters can be derived from data. Schatzmann and Young (2009) describe how this can be done with the expectation maximisation algorithm. Keizer et al. (2010) built on the ABUS, incorporating a decision network as a way to achieve higher variability in user behaviour.

Since these simulators perform at semantic level, a NLG component could be introduced to transform dialogue acts into utterances as a following step. However, as found in the literature (Li et al., 2016), it is preferable to perform training in an end-to-end manner, limiting the number of independent components. Moreover, the use of handcrafted rules and templates still represents the major drawback of this approach. As a way to alleviate these disadvantages, deep-learning simulators emerged in recent years. The following section covers the different neural models that have been previously investigated.

2.1.4 Neural Simulators

The past non-deep-learning simulators suffer from disadvantages like the limited ability to represent dialogue history, the need to define extensive rules (which becomes infeasible for complex tasks and typically requires expert knowledge) or a degree of rigidity that limits the real-user replication ability. In contrast to these methods, the rise of deep learning led to promising alternatives which are specially relevant to our work, and which we review in the following sections. As it can be observed, variability arises from the architecture of these models, as well as from the form of input (feature vs natural language) and output (semantic vs utterance level).

Sequence-to-Sequence models

Due to the sequential nature of the task, sequence-to-sequence models, firstly proposed by Sutskever et al. (2014), became an excellent option to model user behaviour conditioning on the previous history. One of the first deep-learning-based approaches was the sequence-to-sequence model presented by El Asri et al. (2016). In it, user behaviour is learned directly from data and the model tracks the dialogue history and generates pertinent dialogue acts using an encoder-decoder recurrent neural network, in particular an LSTM. They use the DSTC-2 dataset for training and engineer feature vectors representing in detail the context of the dialogue, including system acts and the status of the user goal, which they use as input to the encoder. We initially use these context vectors, which we describe further in Chapter 3. They reported an improvement in the F-score metric compared to the ABUS, however, they did not use the simulator to train a dialogue system and evaluate the learnt policy.

Li et al. (2016) combined the seq-to-seq architecture with the agenda-based approach, using the latter for planning and relying on templates to generate natural language from dialogue acts. When a generated dialogue act did not match any of the templates, a seq-to-seq NLG was employed. Liu and Lane (2017) also employed an LSTM-based seq-to-seq model and a NLG module to transform the dialogue acts into utterances. The novelty in their approach resides in the way they train the simulator, through direct interactions with a dialogue manager in a joint RL optimization scheme. Other papers (Papangelis et al., 2019; Tseng et al., 2021) also train the simulator and dialogue system jointly through RL.

Further work by Kreyssig et al. (2018) also made use of an LSTM sequence-to-sequence architecture. Their setup is inspired by the work in (El Asri et al., 2016), as they use the same feature vector representation as input, as well as their exact goal generation approach. However, they moved away from semantic representation to alternatively generate words, with the goal of obtaining more diversified outputs and easing the semantic labelling requirements. Furthermore, semantic-level simulation will likely require an NLG component to transform acts into utterances that can be taken as input by a dialogue system. Arguably, a better alternative could be a word-level simulator like this one, which bypasses the additional component that could introduce more errors and inconsistencies. Additionally, their model is able to introduce goal changes, typically encountered in the available data. To generate the utterances they apply beam search and they train a dialogue system against their model. Their user simulator outperformed agenda-based simulators on several metrics, including dialogue success rate.

Nie et al. (2019) take a similar approach, but replaced the handcrafted feature extractor used by Kreyssig et al. (2018) with a graph convolutional network.

Another instance of utterance generation instead of semantics can be found in the work of Crook and Marin (2017). They additionally also use natural language as input, building a NL-NL model that can be automatically trained and deployed without requiring semantic annotations, which represents the main advantage of this method. Furthermore, they introduce connections from the GRU encoder's summary vector to each time step of the unrolled LSTM decoder, in order to avoid information loss, something particularly helpful when generating longer utterances. Nevertheless, they do not condition on a specific goal and, thus, only use their simulator for evaluating, and not for training, dialogue policies.

Work by Hou et al. (2019) implements a State2Seq model where they use the word "state" to convey the fact that they encode the sequence of vector representations of each turn into a general dialogue context, in the line of (El Asri et al., 2016). They focus on semantic generation and aim to remediate the common data scarcity problem of DL models by combining ideas from rule-based and data-driven approaches. More specifically, they generate realistic dialogue data from templates with an auxiliary RL-based built-in agent to improve sample diversity. They then use this data to train an LSTM-based architecture that makes use of attention to improve the dialogue context representation for decoding. The experiments carried out proved that attention and a refined context representation based on a forgetting mechanism allowed the State2Seq model to outperform the sequence-to-sequence baseline (El Asri et al., 2016) in terms of the F-score and average dialogue success rates of policies trained.

Lastly, work by Gur et al. (2018) introduced the Variational Hierarchical User Simulator (VHUS), inspired by hierarchical seq-to-seq models (Serban et al., 2017) which outperformed the plain seq-to-seq in dialogue generation. Their model doesn't require a hardcoded feature representation, instead system actions and user goal are encoded with a series of decoupled RNNs, allowing easier domain adaptation. They also introduce a novel goal generalization technique to avoid longer dialogues when user turns diverge from the initial goal.

Transformer-based models

The emergence of the transformer model (Vaswani et al., 2017) has revolutionised the field of NLP. Although they were originally proposed as a sequence-to-sequence model for machine translation, transformer-based models have achieved state-of-the-art performances on a diverse range of tasks, including recently that of user simulation as presented in the work of Lin et al. (2021). Their Transformer User Simulator (TUS) outperformed VHUS (Gur et al., 2018), the previous data-driven state-of-the-art simulator, and policies trained with it showed comparable success rates to those of the ABUS without requiring any handcrafting. Although TUS uses features to represent the dialogue context, its feature extraction is designed to be domain-independent so it can be used to model multiple domains at once, which represents the main advantage of this approach. The domain adaptation capability of the model was proved with a zero-shot transfer learning study. They achieved promising results, but outline how it would be interesting to move to natural language generation as a next step, as they currently generate dialogue acts.

Recent work by Kim and Lipani (2022) proposes a modified version of the original transformer architecture to model user behaviour at the word level, among other tasks.

In conclusion, although the work with transformers in user simulation has a somewhat limited literature, these models may be the key to deal with the barriers regarding intrinsic features of language, like grammar, syntax, and semantic properties, which represent a challenge to add on top of the simulating task.

2.1.5 Multitask learning approach

The literature reviewed so far has exclusively engaged in individual generation, either of semantics or utterances. Nevertheless, in recent years there have been advances in multitask learning (MTL), by which models are trained to perform several tasks, like for instance simultaneous semantic and word generation, in an attempt to leverage shared input representations that improve performance on all the tasks. MTL can be seen as a subset of transfer learning which has as inspiration source the human learning behaviour, considering we often apply prior knowledge when trying to learn a new task (Caruana, 1993; Zhang and Yang, 2017).

There has been significant research investigating MTL. It was introduced in the field of NLP by Collobert and Weston (2008), and Dong et al. (2015) extended the approach to the seq-to-seq task for machine translation, where they proposed using a single encoder and multiple decoders for translation from English to different languages. Although it has led to improvements in performance in several cases, the reason for this is generally unclear

(Bingel and Søgaard, 2017; Martínez Alonso and Plank, 2017). Some advantages potentially brought by MTL that are hypothesized in the literature are:

- Introducing model regularization (Bingel and Søgaard, 2017).
- Aiding the model to identify the most significant features (Ruder, 2017).
- Obtaining a training signal from linked tasks can potentially bias the model towards decisions that benefit both tasks (Caruana, 1993).

The literature presents several methods to perform MTL. Aside from combining the losses, one can implement an alternating training scheme, by which different batches of the corpus are used to train exclusively one particular task. This allows setting a main task (dedicating more batches to it) and auxiliary tasks.

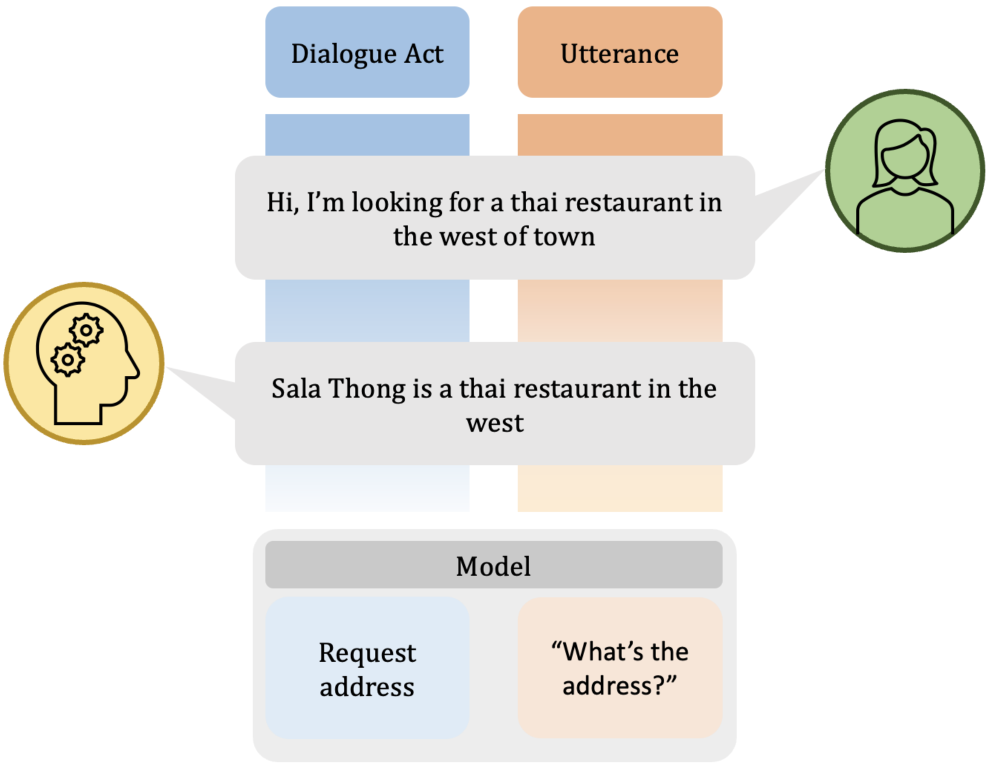

Dialogue Act
Utterance
Hi, I'm looking for a thai restaurant in the west of town
Sala Thong is a thai restaurant in the west
Model
Request address
"What's the address?"

Fig. 2.2 Multitask learning approach in user simulation.

Several attempts at MTL can be found in the subfield of dialogue systems, with the most recent work linked to the use of transformers. One of the first advances in (Xu et al., 2021) implemented a BERT-based architecture trained on four auxiliary tasks in addition to the main task of utterance generation. They demonstrated that the inclusion of all four tasks contributed to improve the generation performance. Zhang et al. (2021) presented DialogueBERT, where they pretrained BERT in five self-supervised tasks.

Literature in MTL specifically for user simulation is, however, very limited, with the only work being developed in parallel by Kim and Lipani (2022). In it, the authors fine-tuned a T5 transformer model (Raffel et al., 2020) in an MTL setting, implementing a user simulator that predicts users' satisfaction scores and actions, and generates users' utterances. They show that all three tasks help each other to better simulate users, further demonstrating the effectiveness of the MTL approach. We also hypothesized that the semantic generation task could give a positive transfer to utterance generation when trained jointly. In consequence, we explored this approach for our word-level simulator, limiting our tasks to semantic and utterance generation as seen in Figure 2.2.

2.2 Evaluation of User Simulators

Despite the advances in the field of user simulation, no standardised metric of evaluation has been developed and evaluating a simulator's performance remains a challenging area of research. This relates to the great diversity seen in simulators. Table 2.1 shows the approaches discussed so far classified in terms of input/output representation, additionally showing the evaluation methods used in each.

The literature contains several surveys on user simulation evaluation (Liu et al., 2016; Pietquin and Hastie, 2013). As a general trend one can distinguish between two groups of metrics: direct and indirect.

Direct methods are corpus-based evaluation metrics that compare the simulator's output with the available data from human users. For the case of semantic-level simulation, direct metrics include precision, recall, balanced F-score, Kullback–Leibler (KL) divergence, and Discourse BLEU (D-BLEU). However, for our word-focused implementation other metrics are potentially more relevant. Traditional language modelling metrics like perplexity, BLEU, ROUGE or NIST all belong to this group and are typically used to evaluate text generation. Thus, papers focused on utterance generation usually choose these metrics. Rather than using them alternatively, their complimentary use allows for a more complete evaluation, as each answers slightly different questions. We discuss them further in the experimental methods in Chapter 4.

On the other hand, indirect methods focus on the interaction of the user simulator with dialogue systems, by evaluating performance of strategies learned from the different simulation techniques. It can be argued that these methods are more representative of the quality of the simulator, given that its performance is being measured at their ultimate goal, which entails interacting with the external dialogue system.

Pietquin and Hastie (2013) outline several methods in this group. Among them, the most commonly used are task completion (which evaluates how many times the dialogue system is able to correctly decipher the goal from the user utterances, indirectly measuring coherence in the simulator) and evaluation of the policies learnt by a dialogue system trained against the simulator, which can be tested against another simulator (cross-model evaluation) or against real users.

Lastly, human evaluation can be considered. Human judges can be asked to rate a whole dialogue, or a simulator's utterance, and real-user transcripts can be included to evaluate how easy it is to discern between real and generated data. However, the main drawbacks of human evaluation are the time required and the complexity in choosing how to present this test to humans, as opposed to using automatic evaluations.

So far we have discussed the alternative approaches to user simulation described in the literature. However, to create our simulator, one key aspect is missing: adversarial networks, which we describe in the following section.

2.3 Generative Adversarial Networks

In this section we expand on the literature of GANs, giving special attention to the task in hand: text generation.

2.3.1 GAN architecture and training

GANs were presented by Goodfellow et al. (2014) as a new framework for estimating generative models. They were first suggested for the task of image generation, where they proved their ability to generate realistic fake images. Their success in this field demonstrated their potential to model complex distributions, and rapidly they began to be used for other purposes, varying from audio enhancement and synthesis to object detection and NLP.

The architecture they introduced is based on two models, a generator G, that seeks to capture the distribution of the training data and a discriminator, D, which differentiates between real and synthesized samples. Training can then be seen as a min-max game in which the generative model aims to maximize the probability of fooling the discriminator, by generating samples that are indistinguishable from real data, while the discriminator goal is to minimize its classification error.

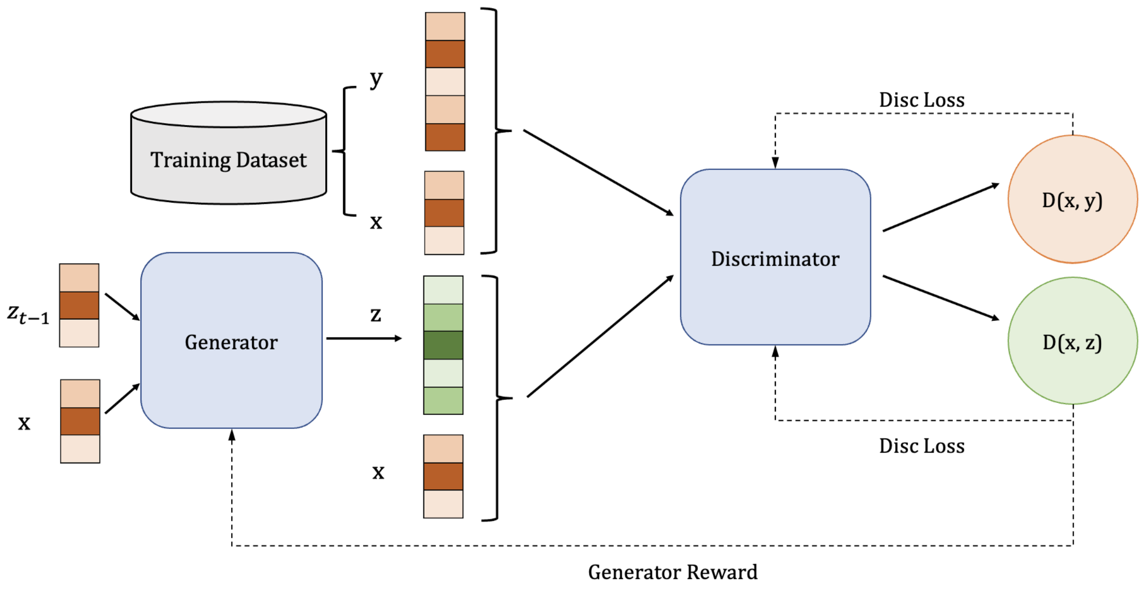


Fig. 2.3 Adversarial training framework.

In our case we are only concerned with the generation task, so once a training equilibrium is reached the discriminator is discarded. Moreover, we would like to introduce a dependence between the generated outputs and the context of the dialogue. This requirement can be met with conditional GANs, firstly introduced by Mirza and Osindero (2014). This modified GAN framework allows to generate samples with a specific property by using context information both in the generator and the discriminator, as seen in Figure 2.3.

The final objective function to optimize can be seen in Equation 2.2 where x represents the context we condition on, y the data that we wish to replicate (text transcripts), $p(x,y)$ the real data distribution, z the generated samples, $p_G(z|x;\theta_G)$ the generator output distribution and $D(x,y)$, $D(x,z)$ the discriminator's output. The generator and discriminator parameters, θ_G and θ_D respectively, are alternatingly trained. The discriminator, parameterized by θ_D, guides the generator with its output $D(x,z)$, indicating whether sequence z belongs to the real data or not. This discriminative model is trained with positive (real) and negative (artificial) samples and with standard ML.

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x,y \sim p(x,y)} \log(D(x,y;\theta_D)) + \mathbb{E}_{x \sim p(x), z \sim p_G(z|x;\theta_G)} \log(1 - D(x,z;\theta_D)) \tag{2.2}$$

2.3.2 GANs for Text Generation

GANs have demonstrated outstanding capabilities at generating plausible images. Nevertheless, their applicability to text generation is hindered, and they have been shown to be difficult to train in the domain of natural language (d'Autume et al., 2019).

The reason behind this is that these networks were designed to deal with continuous information rather than discrete data like text. When working with continuous data, the network parameters can be updated by optimizing the objective function in Equation 2.2 directly through gradient backpropagation. However, the sampling operation involved in text generation is non-differentiable, making direct backpropagation of the discriminator loss impossible. To overcome this issue, the literature in GANs widely resorts to two methods: (1) reliance on approximating the sampling operation with Gumbel-Softmax (Kusner and Hernández-Lobato, 2016) and soft-argmax (Zhang et al., 2017) as a strategy to make the model differentiable, or (2) using policy gradient methods (Scialom et al., 2020; Yu et al., 2017a) essentially deploying the REINFORCE algorithm (Williams, 1992) to get an unbiased gradient estimator for the generator.

In particular, we focus our investigation in the RL-based approach. Following this, the generator becomes an agent that seeks to learn a policy $\pi(a|s)$, where s is the state of the agent (text *already* generated), and a its actions (text *to be* generated). In essence, the agent generates text based on the policy π, where actions (words) are sampled according to the distribution defined by the same policy. Consequently, it is possible to optimize θ_G by employing policy gradient methods, using the probability coming from the discriminator as a reward signal to update the generator's parameters. The gradient estimator is defined as:

$$\nabla_{\theta_G}\mathbb{E}_{x\sim p(x),z\sim p_G(z|x;\theta_G)}[D(x,z)] = \mathbb{E}_{x\sim p(x),z\sim p_G(z|x;\theta_G)}[D(x,z)\nabla_{\theta_G}\log(p_G(z|x,\theta_G))] \tag{2.3}$$

In turn, the discriminator's parameters are still updated via ML.

Despite the efforts that have been made to accommodate GANs to sequence generation, training them for this task remains a challenge. Aside from the difficulties introduced by the discreteness of the data, the GAN architecture already suffers from shortcomings (Jabbar et al., 2021) like the Nash Equilibrium (Ratliff et al., 2013), internal covariate shift (Ioffe and Szegedy, 2015) or mode collapse (Arora et al., 2017). In consequence, several variants to the architecture and loss of these models have been studied to stabilize training.

Specifically concerning our task, the first work in RL-based GAN training is found in SeqGAN (Yu et al., 2017b). Despite the advance it achieved, the model lacked generalization power and produced text with minimum diversity (de Rosa and Papa, 2021). From this point on different configurations started to appear, aiming to find stable, better performing training methods and architectures. Some variants included additional information in the network to improve performance, like LeakGAN (Guo et al., 2017), which leaked high-level features from the discriminator into the generator, or VPGAN (Hu et al., 2020), which included features learned by Variational Autoencoders. This method can, nonetheless, reduce diversity in the outputs. Other work is focused instead on modifying the rewards, like DPGAN (Xu et al., 2018), designed to alleviate repetitiveness by assigning low rewards to repetitive text and high rewards to novel text, or the approach in (Xu and Fung, 2019), also aiming to tackle the repetition problem through the inclusion of a modified ROUGE metric that penalizes repeated words. Another example is RankGAN (Lin et al., 2017), whose rewards are based on a ranking of the outputs rather than the standard binary output. These ideas inspired different modifications performed in our experiments.

Other approaches seen in the literature slightly modify the RL training framework to include the ML objective, like in the case of the ARAML approach (Ke et al., 2019) in which samples to be fed to the discriminator come from a stationary distribution near the data instead of that of the generator, inspired by the work in MaliGAN (Che et al., 2017) and RAML (Norouzi et al., 2016).

Additional work focused on improving adversarial training can be found in (d'Autume et al., 2019) which introduce training modifications aimed to increase stability, like increasing the batch size, or obtaining dense rewards from the discriminator, in other words calculating rewards at each generation step. This last technique was also suggested by Li et al. (2017), who train a text generation model for open domain chit-chat dialogue systems through adversarial training, a similar task to ours.

Finally, work linking adversarial training and transformers has surfaced in recent years, in which adversarial fine-tuning is used on large pre-trained language models. For instance in TextGAIL (Wu et al., 2021), the authors make use of these models (RoBERTa and GPT-2) along with recent techniques in RL, like proximal policy optimization (PPO) to reduce variance in the gradients. They also introduce an imitation replay method to stabilize the training and, additionally, ground-truth sequences are used to train the generator as well, in order to force occasional constant rewards and stabilize convergence. They also propose modifications to the discriminator based on a contrastive approach, which estimates how much more realistic a real sentence is than a generated sentence, given a context.

While we presented here several approaches targeting robust adversarial text generation, we direct the reader to (de Rosa and Papa, 2021) for a complete review of all the advances in the field and techniques to improve GAN training. In Chapter 5 we reiterate in the procedures that we specifically followed to stabilize adversarial training.

Table 2.1 Classification of User Simulation in the Literature (DA stands for dialogue act).

Reference	Input	Output	Metrics	Objectives/Results
Eckert et al. (1997)	Last system act	DA	None	Initial study in user simulation based on a bigram probabilistic model.
Levin et al. (2000)	Last system act	DA	None	Expanded on (Eckert et al., 1997) by only allowing "sensible" pairs of user response and system action.
Scheffler and Young (2002)	Last system act	DA	Task completion (# of turns)	Devise a decision network mapping context-action, and train parameters for goal-independent actions.
Georgila et al. (2005)	Last n system/user acts	DA	Perplexity and Task completion (% slots filled)	Extended (Eckert et al., 1997) n-gram count.
Pietquin (2006)	Last system act	DA	Performance of learnt policy (evaluation with own model)	Established a probabilistic goal-directed user simulator.
Schatzmann et al. (2007)	Last system act	DA	Performance of Learnt Policy (human evaluation)	Implementation of the ABUS.
Schatzmann and Young (2009)	Last system act	DA	Performance of learnt policy (cross-model and human evaluation)	Introduce trainable parameters in the ABUS using the EM algorithm.
Jung et al. (2009)	Features (manual)	DA	BLEU, D-BLEU, KL Divergence	Model the user employing a linear-chain conditional random field.
Keizer et al. (2010)	Last system act	DA	F-score, KL-divergence and performance of policy (cross-model)	Introduce a decision network in the ABUS.
El Asri et al. (2016)	Features (automatic extractor)	DA	F-score	One of the first approaches to implement an LSTM-based simulator. They reported improvement in F-scores compared to the ABUS.
Li et al. (2016)	Last system act	DA	Performance of learnt policy (own-model)	Combination of seq-to-seq and agenda-based methods. Relied on templates to generate utterances from DAs and on a seq-to-seq NLG module in case the act wasn't present in said templates.

Table 2.2 (continued)

Reference	Input	Output	Metrics	Results
Crook and Marin (2017)	NL	NL	Perplexity and direct human evaluation	GRU/LSTM encoder-decoder with intermediate connections between both. Their system is goal-independent and they only test their approach in the task of evaluating TOD systems.
Kreyssig et al. (2018)	Features (automatic extractor)	NL	Performance of learnt policy (cross-model and human evaluation)	Extended (El Asri et al., 2016) to NL generation
Gur et al. (2018)	NL	DA	Perplexity and Task completion (% slots filled)	Introduced a new hierarchical process to encode dialogue history and user goal.
Nie et al. (2019)	Features (Graph CNN)	NL	BLEU	Replacement of the manual feature extractor with a Graph CNN
Hou et al. (2019)	Features (automatic extractor)	DA	F-score and performance of learnt policy	Combination of rule-based and data-driven approaches in an LSTM architecture with attention. They report an increase in the F-score and dialogue success rate with respect to (El Asri et al., 2016).
Shi et al. (2019)	Features (GRU encoder)	DA	Perplexity and taks completion (% slots filled)	Implementation of six user simulators trained with different dialogue planning and generation methods.
Lin et al. (2021)	Features (automatic, domain-independent extractor)	DA	F-score and performance of learnt policy	First implementation of a user simulator with transformers. Achieved state of the art results and domain independence
Kim and Lipani (2022)	NL	NL,DA	BLEU, ROUGE	Provided proof that a transformer-based MTL scheme can improve performance of several tasks related to TOD systems, included utterance generation.

Chapter 3

The word level GAN Simulator: Possible Approaches

In this section we describe the different approaches investigated to build the user simulator. Mainly we present two possible solutions, one based on an LSTM encoder-decoder architecture, previous state-of-the-art for sequential data, and a second method grounded on transformers, current state-of-the-art. These models differ both in the architecture and the form of the input, as we outline here.

3.1 First Approach: LSTM

Due to the general success across a wide range of tasks of the seq-to-seq model, as well as due to its simple design, we initially adopt this model for our user simulator.

Our investigation efforts started with a similar architecture to the one presented in Kreyssig et al. (2018), which we cover in detail in the following section.

3.1.1 Inputs

Initially, our simulator takes a feature vector related to the restaurant domain as input, as seen in (El Asri et al., 2016; Kreyssig et al., 2018). These vectors, automatically extracted using a handcrafted feature extractor, allow us to capture the most information while keeping the task of learning the context relatively simple for the encoder. Nevertheless, using features is less convenient than using natural language directly because it requires semantic annotations and a greater extent of manual feature engineering, supposes a more rigid representation and prevents automatic domain adaptation, as they specifically target the restaurant domain.

We follow the procedure in (El Asri et al., 2016) and (Kreyssig et al., 2018) to construct our binary-vector representations. They contain information related to four aspects: user constraints ($const_t$), slots requested by the user (req_t), previous system act (act_t) and the occurrence of inconsistencies (inc_t) between the system act and the user goal. These are concatenated as follows:

$$const_t + req_t + inc_t + act_t \tag{3.1}$$

The constraints vector, ($const_t$), contains information about the state of the informable slots (constraints the user has, e.g. an area), with a value of 0 meaning a slot is still to be informed by the user. This value turns to 1 when the information is given, and returns to 0 if the system asks again for the slot value or makes a mistake related to it. Values not included in the goal are set to 1 from the start.

$$const_t = \begin{bmatrix} \text{inform food} \\ \text{inform area} \\ \text{inform pricerange} \end{bmatrix}$$

The request vector represents which slots are still to be requested by the user. The slots contained in the goal, and thus the ones the user requires information about, are initially set to 0. They change to 1 once the user asks for the information. Note that this vector is linked to a venue, and thus if the restaurant under consideration changes this vector needs to be reset.

$$req_t = \begin{bmatrix} \text{request food} \\ \text{request area} \\ \text{request pricerange} \\ \text{request name} \\ \text{request address} \\ \text{request phone} \\ \text{request postcode} \\ \text{request signature} \end{bmatrix}$$

Thirdly, the inconsistency vector is used to signal whether any mistakes or incorrect assumptions about the user's constraints were made by the system. These values are set to 0 if no mistakes occurred.

$$inc_t = \begin{bmatrix} \text{inconsistency food} \\ \text{inconsistency area} \\ \text{inconsistency pricerange} \end{bmatrix}$$

Finally the system's act vector indicates the previous action taken by the system, out of the 16 possible.

$$act_t = \begin{bmatrix} \text{act welcomemsg} \\ \text{act offer} \\ \text{act inform} \\ \text{act canthelp} \\ \text{act canthelp exception} \\ \text{act confirm domain} \\ \text{act reqmore} \\ \text{act impl-conf} \\ \text{act expl-conf} \\ \text{act repeat} \\ \text{act request food} \\ \text{act request area} \\ \text{act request pricrange} \\ \text{act select food} \\ \text{act select area} \\ \text{act select pricerange} \end{bmatrix}$$

For illustration purposes, Figure 3.2 shows two turns of a dialogue with the corresponding vectors that would be used to get the next user response.

3.1.2 Generator: LSTM Encoder-Decoder

The first configuration uses the previous state-of-the-art models for sequential language generation: Recurrent Neural Networks, in particular a Long Short Term Memory (LSTM) encoder-decoder architecture (Hochreiter and Schmidhuber, 1997). In the next section we include a description of the LSTM building block.

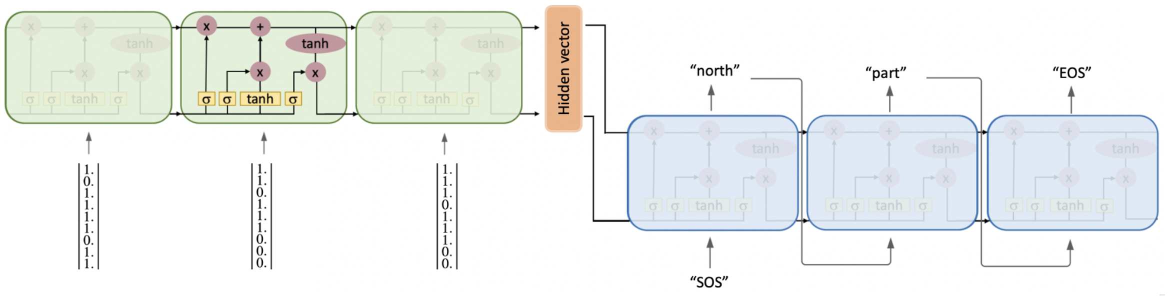
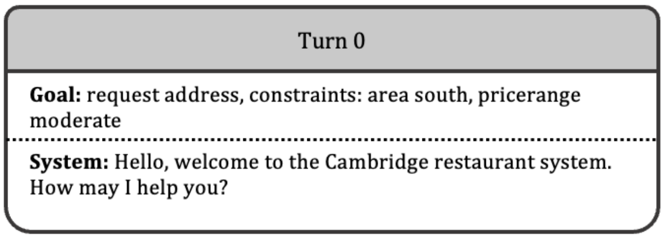


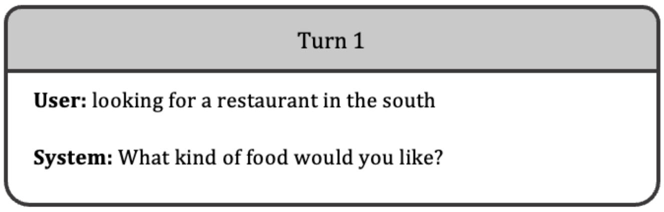
Fig. 3.1 LSTM encoder-decoder.


Turn 0

Goal: request address, constraints: area south, pricerange moderate

System: Hello, welcome to the Cambridge restaurant system. How may I help you?

$$x_0 = \begin{bmatrix} 1. \\ 0. \\ 0. \\ 1. \\ 1. \\ 1. \\ 1. \\ 0. \\ 1. \\ 1. \\ 1. \\ 0. \\ 0. \\ 0. \\ 1. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \end{bmatrix} = \begin{bmatrix} \text{inform food} \\ \textbf{inform area} \\ \textbf{inform pricerange} \\ \text{request food} \\ \text{request area} \\ \text{request pricerange} \\ \text{request name} \\ \textbf{request address} \\ \text{request phone} \\ \text{request postcode} \\ \text{request signature} \\ \text{inconsistency food} \\ \text{inconsistency area} \\ \text{inconsistency pricerange} \\ \textbf{act welcomemsg} \\ \text{act offer} \\ \text{act inform} \\ \text{act canthelp} \\ \text{act canthelp exception} \\ \text{act confirm domain} \\ \text{act reqmore} \\ \text{act impl-conf} \\ \text{act expl-conf} \\ \text{act repeat} \\ \text{act request food} \\ \text{act request area} \\ \text{act request pricerange} \\ \text{act select food} \\ \text{act select area} \\ \text{act select pricerange} \end{bmatrix}$$


Turn 1

User: looking for a restaurant in the south

System: What kind of food would you like?

$$x_1 = \begin{bmatrix} 0. \\ 1. \\ 0. \\ 1. \\ 1. \\ 1. \\ 1. \\ 0. \\ 1. \\ 1. \\ 1. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 1. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \end{bmatrix} = \begin{bmatrix} \textbf{inform food} \\ \text{inform area} \\ \textbf{inform pricerange} \\ \text{request food} \\ \text{request area} \\ \text{request pricerange} \\ \text{request name} \\ \textbf{request address} \\ \text{request phone} \\ \text{request postcode} \\ \text{request signature} \\ \text{inconsistency food} \\ \text{inconsistency area} \\ \text{inconsistency pricerange} \\ \text{act welcomemsg} \\ \text{act offer} \\ \text{act inform} \\ \text{act canthelp} \\ \text{act canthelp exception} \\ \text{act confirm domain} \\ \text{act reqmore} \\ \text{act impl-conf} \\ \text{act expl-conf} \\ \text{act repeat} \\ \textbf{act request food} \\ \text{act request area} \\ \text{act request pricerange} \\ \text{act select food} \\ \text{act select area} \\ \text{act select pricerange} \end{bmatrix}$$

Fig. 3.2 Representation of two dialogue turns with context vectors. The user had to inform constraints related to area and pricerange and request the address (as seen in the goal and the initial vector). Once it has informed the area in Turn 1 we see how the corresponding slot changes to 1 for the next turn. As the system explicitly asks for the food in Turn 1, this slot changes to 0 (although the user is not constrained in this regard it now has to inform as much).

3.1.3 LSTM Networks

LSTMs are recurrent networks capable of learning long-term dependencies. Its core particularity is the inclusion of a memory cell state, in which the flow of information is regulated through three gates (sigmoid layers), as shown in Figure 3.1. The first of them is the forget gate, f_t, which establishes how much information is to be discarded from the previous cell state, c_{t-1}. The second layer, the input gate, i_t, regulates how much new information is included in the update of the cell state. For this update in particular, a tanh function, g_t, plays the role of providing new candidates for the cell state, which are scaled by the input gate. Finally, the output gate, o_t determines how much of the new state, c_t, is actually passed on to the hidden state going to the next layer. The final output is the Hadamard product between the output gate and the cell state, firstly put through a tanh to push the values in the range $[-1,1]$.

All gates can be seen in Equations 3.2-3.7, where matrices W and biases b are parameters to be learnt.

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{3.2}$$

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{3.3}$$

$$g_t = tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{3.4}$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{3.5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{3.6}$$

$$h_t = o_t \odot tanh(c_t) \tag{3.7}$$

These LSTM networks compose the encoder and decoder of our first architecture. We include below the specific implementation of each.

Encoder

The encoder takes as input all past context vectors up to the turn for which we wish to generate a user utterance. The final configuration implemented consists of one hidden LSTM layer of size 32. After encoding the context vectors, the final hidden vector (created sequentially) is used to initialize the hidden state of the decoder.

Decoder

The decoder is composed of an LSTM layer of size 32, equivalent to the encoder [1]. Its hidden and cell states are initialized with the final values from the encoder, and the start of sequence token is fed to start the generation.

At each time step t the decoder generates continuous outputs from a linear layer, parameterised as:

$$y_t = h_t W + b \tag{3.8}$$

These continuous representations are transformed into valid probability distributions over the vocabulary with a softmax layer.

$$p_{it} = \frac{exp(y_{it})}{\sum_j exp(y_{jt})} \tag{3.9}$$

Where p_{it} represents the probability of the i_{th} token in the vocabulary. We sample from this distribution at each step to obtain the next word in the utterance. In particular, we employ a typical technique in LSTMs, teacher-forcing, with a probability of 0.5.

Whenever teacher-forcing is applied we feed the ground-truth tokens to the decoder at each time step, instead of the sampled tokens. Kreyssig et al. (2018) always use teacher-forcing during training, but we found that establishing a lower teacher-forcing ratio (0.5 specifically) was preferable. This could also avoid the problem of "exposure bias" (Bengio et al., 2015) (i.e. not having the ground-truth available) which could degrade the generation capability during inference.

Adding attention

Attention is a learning mechanism that enables the decoder to focus on different parts of the encoding when generating the output at each time step. Rather than relying solely on the last hidden vector to represent the context, attention enables shortcuts between the input sequence and the final context vector used at decoding. Then for each output step, the weights of these dynamic context vectors are learnt.

Experiments in (Hou et al., 2019) proved that adding attention to the dialogue context improved the user model and the agent policy. We tested this in our own user simulator.

Mathematically, the decoder is trained to predict the next word given all previous words and a context. Without attention, this context vector c is simply the encoder's final hidden representation:

[1]Experiments with different hidden sizes did not result in a meaningful fluctuation in performance.

$$p(\mathbf{y}) = \prod_{i=1}^{T} p(y_i|y_1,...,y_{i-1},c) \tag{3.10}$$

However, by including attention, this context can change at each time step of the generation:

$$p(\mathbf{y}) = \prod_{i=1}^{T} p(y_i|y_1,...,y_{i-1},c_i) \tag{3.11}$$

The variable context vector c_i will depend on the sequence of annotations outputted by the encoder at each step $(h_1,\ldots,h_T)$, each of them containing information on all previous inputs but with an emphasis on the ones closer. Specifically, c_i is computed as:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j \tag{3.12}$$

Where α_{ij} is a weight computed for each of the encoder outputs:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_k exp(e_{ik})} \tag{3.13}$$

And where e_{ij} represents the alignment between the j_{th} input and the i_{th} output. This score would depend in the previous decoder hidden state (h^D_{i-1}) and the encoder output h_j:

$$e_{ij} = a(h^D_{i-1}, h_j) \tag{3.14}$$

This alignment is implemented as a feed-forward network.

3.1.4 Discriminator

We initially consider the same discriminator used for the previous semantic-level simulator, a feed-forward (FF) network that takes as input the concatenation of the last context vector and the one-hot embeddings representing each of the generated tokens. We pad this input up to the maximum allowed generation length (25 tokens). The architecture then includes a hidden layer of size 64 and a final softmax-activated layer, outputting a 2-dimensional vector representing the probability that the utterance was real. We use this as the reward during training.

Additionally, an LSTM-based discriminator was also proposed. Among the several possible configurations, we implement one which takes the final encoded hidden and cell states from the generator's encoder as initialization for the discriminator's hidden layer. In this case, unlike in the FF discriminator, the sequence generated is fed one time step at a time.

3.1.5 Training

Training is performed with the REINFORCE algorithm, by generating sequences and feeding them to the discriminator along with the context to obtain a reward, R. This reward is then used along with the log-probabilities outputted by the model at each step to calculate the loss for the generator:

$$\mathscr{L} = -R\log(p_G(z|x,\theta_G)) \tag{3.15}$$

3.1.6 Multitask Approach

Although we consider this to be beyond the scope of this project, we would be interested in generating both dialogue acts and utterances for a future interaction of the LSTM-simulator with a dialogue system, as this requires the user dialogue acts to update the input context vector. This was not needed during training of the simulator, considering the dialogue acts are taken from the corpus, but when deploying it against a dialogue system it needs to actually generate them. Instead of developing a NLU module to decode the semantic meaning out of sentences, we propose a multitask learning scheme aiming to output accurate utterances along with dialogue acts. Further motivation to perform multitask learning relates to the positive transfer between tasks seen in the literature, as reviewed in Chapter 2. In essence, we also aim to determine whether learning both tasks simultaneously, leveraging common representations for different decoders, could lead to a performance improvement.

As described in Chapter 2 there are several ways to train a multitask learning model. We focus on two possibilities as shown in Figure 3.3. The first is to alternate between training decoders for a given number of batches, i.e. training only one task per batch. This allows a greater focus on our main task, utterance generation, by dedicating a higher number of batches to it while treating the semantic generation as an auxiliary task.

The second method combines both losses and updates both decoders and the encoder with it.

For MLE these updates are through backpropagation, whereas in adversarial training we continue using the REINFORCE algorithm, but with a network now composed of two discriminators, one for words and another for semantics. Additionally, each task could be trained with a different training approach.

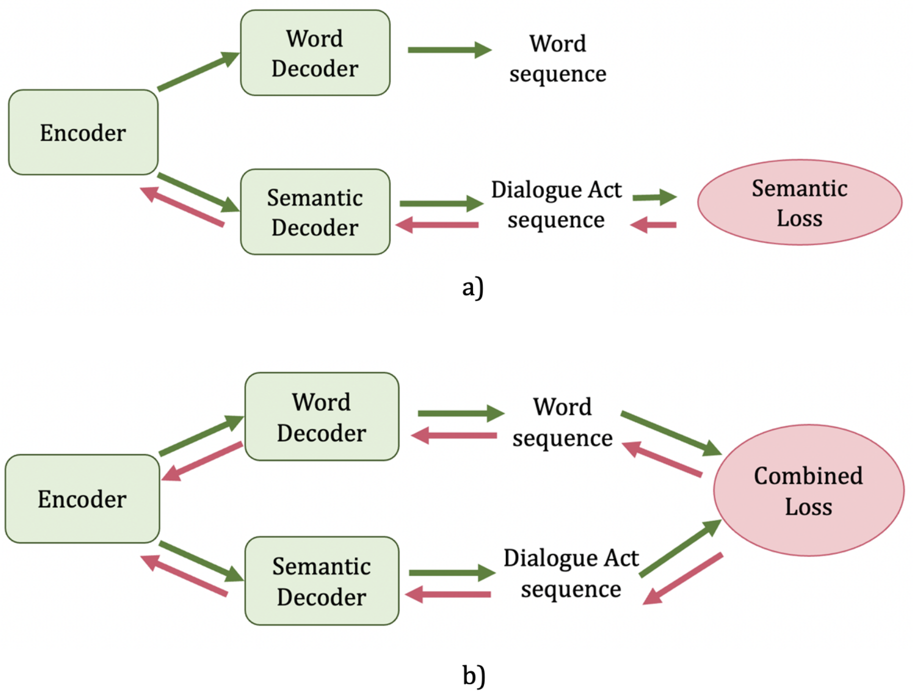


Fig. 3.3 Considered multitask learning approaches.

3.2 Second Approach: Transformer-based Architecture

One drawback of the use of DL approaches in the field of dialogue systems is related to data scarcity, which can lead to undertrained models or, contrarily, models biased towards a specific corpus as discussed by Shi et al. (2019).

Recent progress in pre-training language models has shown promising results in alleviating the data scarcity problem (Budzianowski and Vulić, 2019). These models, typically pre-trained on large texts with self-supervised objectives like language modelling (Radford et al., 2019), can be fine-tuned to a specific task, what has led to performance improvements on a wide range of NLP applications.

Moreover, while the use of LSTMs entails less complexity and computational cost, one major drawback they display is that no variability can be observed outside of what is present in the data, as the model is constrained to the relatively small vocabulary size of a particular dataset. Thus, the only output variability one can expect from these models comes essentially from grammatical and syntactical variations. Conversely, transformer models are pre-trained on a extensive vocabulary, which theoretically would allow other variations like synonymy. This is why we decide to apply these models to our user simulator.

In this section we describe how to incorporate pre-trained language models into our task, first highlighting several advantages brought by them.

3.2.1 Key Advantages

While architectural complexity and memory cost can greatly increase with the use of pre-trained transformer models, this approach can bring meaningful advantages compared to the LSTM simulator:

- Potential increase in output variability considering the increase in vocabulary size (previously 565 words, with this approach 50257 tokens[2]).
- Incorporation of previous knowledge in language modelling.
- Ability to use attention to capture information from long-range sequences.
- Parallelization thanks to its non-sequential operations (allowed by positional embeddings).
- Possibility to output the values for the slots without the need for delexicalization.
- Need for semantic annotations is eased, which reduces the efforts needed to obtain more data for training.
- Increase in the model's ability to capture past information. For instance, if the system proposed several restaurants, we would be able to refer to any of them (e.g asking questions about whether the newly proposed restaurant is more expensive than the previous one). This was impossible with the LSTM model due to the rigidity imposed by the context vector.
- Easier domain adaptation thanks to the flexibility given by the NL inputs.

We now present the pre-trained language models chosen for our generator and discriminator.

3.2.2 Generator: GPT-2

GPT-2 (Radford et al., 2019) is a transformer-based language model trained on the WebText dataset. Derived from the initial transformer architecture, GPT-2 only uses the decoder side of a transformer. More specifically, GPT-2 is made of stacked decoders, 12 in the particular version that we use (GPT-2 small), with 12 heads and 117M parameters.

[2]We use Byte Pair Encoding not Word Pair Encoding, so the 50257 are not exclusively complete words

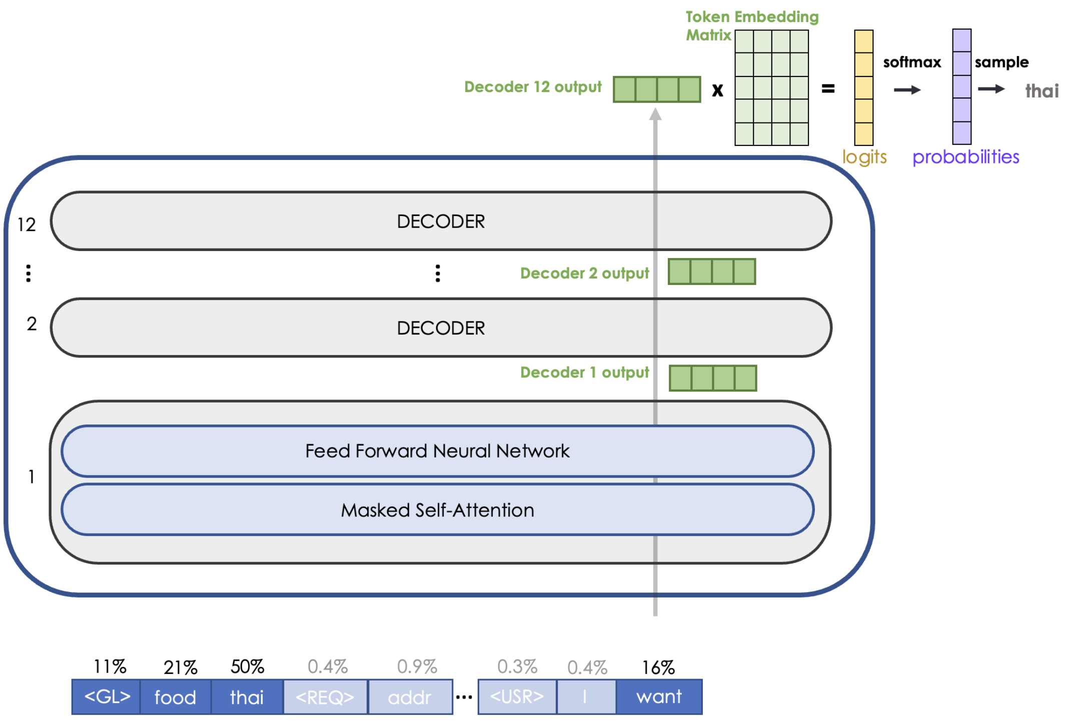


Fig. 3.4 GPT-2 architecture.

One of the model's most crucial aspects is its masked multi-head self-attention, which allows to model associations between words, enabling to focus to a higher or lesser extent on previous outputted tokens when processing the current output, as seen in Figure 3.4. Aside from the attention layer, each decoder additionally includes a feed-forward and a normalization layer.

We take this pre-trained language model from Hugginface transformers, and fine-tune it including a LM head (`GPT2LMHead`) in our task of user simulation.

Generator's Input

GPT-2 operates exclusively with inputs in the form of strings, unlike most previous simulators which tend to rely on numerical vector representations.

Our data is structured and needs to be "linearised" into sequences. Linearisation is a technique by which inputs from a datastructure are transformed into sequenced data through the inclusion of special tokens that help maintain the information.

In consequence, we design the inputs as follows:

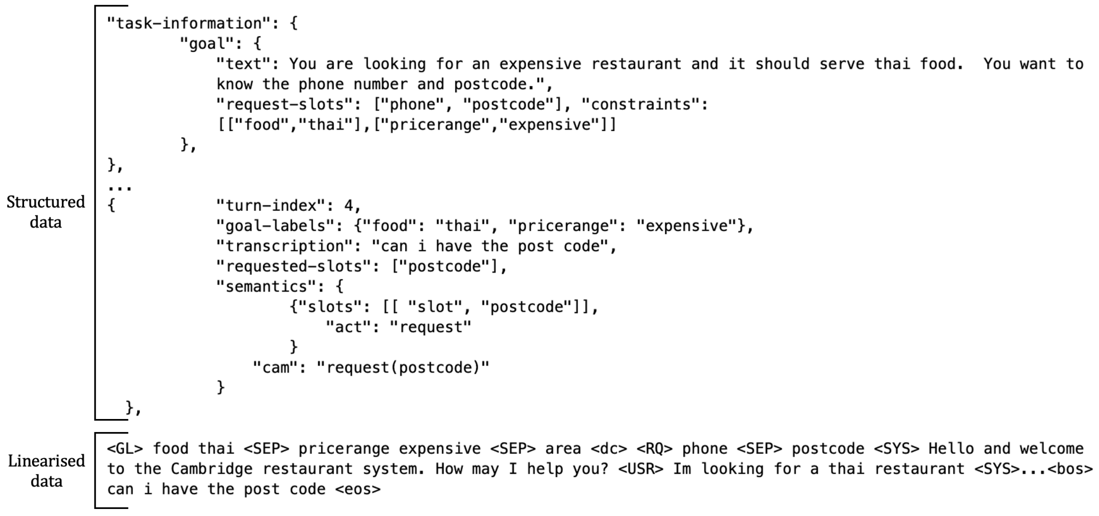
1. The goal is encoded using a special token, `<GL>`. It includes the constraints and the slots to be requested by the user, the latter identified by another special token, `<RQ>`.

Additionally, the special token `<SEP>` is created to separate different slots in the goal.

`<GL>` food indian `<SEP>` area north `<SEP>` price expensive `<RQ>` phone

2. The dialogue history is represented directly by the system and user utterances. At each turn we accumulate information from all past turns (a limit of 512 tokens is set as the maximum allowable length; if this limit is passed the history gets truncated to exclude initial turns). For system/user differentiation we include the special tokens `<SYS>` and `<USR>`. Moreover, to perform ML training, the last user utterance at each turn (i.e. what we aim to predict) is enclosed in the special tokens `<bos>` and `<eos>`, beginning and end of sequence respectively.

Figure 3.5 shows an example of data linearisation.


Structured data
Linearised data

```
"task-information": {
        "goal": {
            "text": You are looking for an expensive restaurant and it should serve thai food.  You want to
            know the phone number and postcode.",
            "request-slots": ["phone", "postcode"], "constraints":
            [["food","thai"],["pricerange","expensive"]]
        },
},
...
{           "turn-index": 4,
            "goal-labels": {"food": "thai", "pricerange": "expensive"},
            "transcription": "can i have the post code",
            "requested-slots": ["postcode"],
            "semantics": {
                    {"slots": [[ "slot", "postcode"]],
                        "act": "request"
                    }
                "cam": "request(postcode)"
            }
  },
```

```
<GL> food thai <SEP> pricerange expensive <SEP> area <dc> <RQ> phone <SEP> postcode <SYS> Hello and welcome
to the Cambridge restaurant system. How may I help you? <USR> Im looking for a thai restaurant <SYS>...<bos>
can i have the post code <eos>
```

Fig. 3.5 Linearisation of the structured data.

Once we have the input string for each turn, they are tokenized along with the labels (ground-truth responses). The labels consist of strings of the same length as the input, but where the context is padded with the ignore token.

There exists several methods of tokenization, in particular GPT-2 makes use of the Byte Pair Encoding technique, a middle ground between character and word level. It uses word-level inputs for frequent symbol sequences and character level inputs for those that are infrequent. Since we use a pre-trained model we maintain the tokenizer that was trained with it, which the authors in (Radford et al., 2019) optimized to reduce the vocabulary size to 50257 tokens.

Along with these tokens, positional embeddings are required, considering the transformer architecture does not receive input sequentially, as was the case of LSTMs. Instead, every past turn is inputted at once.

Lastly, training is performed in batches that should be of the same size, so we pad inputs to the maximum length in the batch.

3.2.3 Decoding Strategy

The final output generated by the last linear layer of the fine-tuned model can be put through a softmax layer to get a valid distribution, i.e. the conditional distribution over all possible tokens given all the previously generated tokens (see Figure 3.4). This distribution is outputted at each step of the sequence.

However, one aspect to consider is that the generated sequences will depend on the particular decoding method. As these sequences directly determine the loss we use to optimize the model (considering it is based on their tokens' log-probabilities and the reward given by the discriminator to them), making sure the decoding strategy is optimal becomes crucial.

Typical decoding methods include:

- Greedy search: only the token of highest probability is considered.
- Beam search: with a beam of size n, we expand on the n tokens of highest probability.
- Top-k sampling: top k tokens are kept and the probability is redistributed among them, sampling is then performed.
- Nucleus sampling (top-p): instead of keeping the k outputs of highest probability, we keep the smallest number of outputs whose cumulative probability surpasses p. Then the distribution is normalized.

3.2.4 Discriminator: RoBERTa

We chose a transformer of the BERT family (Devlin et al., 2019) for our discriminator, given their typical use for the task of classification. In particular, RoBERTa (Liu et al., 2019) maintains BERT's architecture, but changes its training scheme by (1) training for longer, (2) including bigger batches over more data and longer sequences, (3) excluding the next sentence prediction objective, and (4) dynamically changing the masking pattern applied to the training data. As a result, it surpassed BERT on several metrics, which is why we choose this particular variant. Moreover, it was trained with the same tokenization scheme as GPT-2 (BPE).

As any BERT-derived model, RoBERTa is composed of stacked transformer encoders (12 blocks with 12 attention heads and 125M parameters). We fine-tune a model with a binary classification head, provided by HuggingFace (`RoBERTaForSequenceClassification`), and include a softmax layer to output a valid distribution. Unlike GPT-2, which has constrained attention (only pays attention to context to the left), RoBERTa employs bidirectional self-attention.

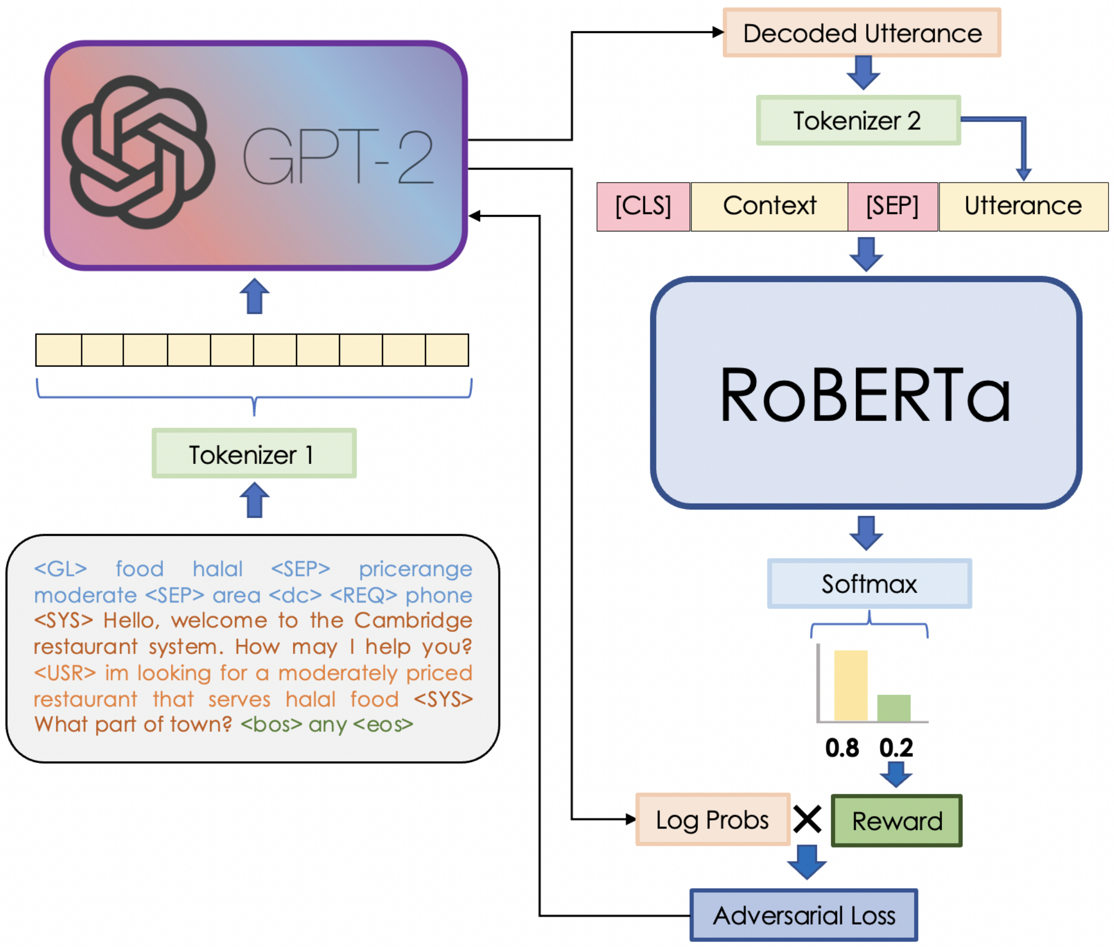

Decoded Utterance
Tokenizer 2
GPT-2
[CLS]
Context
[SEP]
Utterance
RoBERTa
Tokenizer 1
<GL> food halal <SEP> pricerange moderate <SEP> area <dc> <REQ> phone <SYS> Hello, welcome to the Cambridge restaurant system. How may I help you? <USR> im looking for a moderately priced restaurant that serves halal food <SYS> What part of town? <bos> any <eos>
Softmax
0.8
0.2
Log Probs
Reward
Adversarial Loss

Fig. 3.6 Architecture with transformers.

Discriminator's Input

Initially the input to RoBERTa will be the context and user utterance, separated by the special token SEP.

Additionally, we include at the beginning of each input sequence the special BERT token CLS, which specifies that the task of classification is to be performed, with context $\{x\}_{i=1}^{N}$, output sequence $\{y\}_{i=1}^{M}$ and label $l \in [0,1]$.

$$[\texttt{CLS}]\, x_1, x_2, ..., x_N\, [\texttt{SEP}]\, y_1, y_2, ..., y_M\, [\texttt{EOS}]$$

As in the first approach, we will use the probability of a generated utterance being real as reward for GPT-2.

3.2.5 Training

As mentioned, we train both GPT-2 with a LM head and RoBERTa with a classification head.

Sequences outputted by GPT-2 are translated to RoBERTa tokens and preprocessed for the discriminator. RoBERTa receives sequences with the generated utterance and the ground-truth (as separate inputs initially, but later we modify the architecture to input both simultaneously) and gives a score to each, which we use to compute the negative log-likelihood loss to train the discriminator. The reward given to the generated sequence is used along with the log-probabilities of the decoded tokens to optimize the generator.

Chapter 4

Experimental Methods

4.1 Dataset

Our experiments use the data from the second Dialogue State Tracking Challenge (DSTC2) (Henderson et al., 2014). This dataset constitutes a large corpus of dialogues between users (crowd-sourced through Amazon Mechanical Turk) and three telephone-based dialogue systems, and it focuses on the restaurant domain. Information for the train, test and validation sets can be seen in Table 4.1. No caller from the train set appears in the validation set. We use the test set for evaluation of the trained models and the validation set to monitor performance evolution during training. In each dialogue the users were asked to request one or several slots (among area, food, price range, name, address, phone number, postcode, and signature dish) of a restaurant that met a set of constraints related only to the area, price range and food type. This dataset also introduced changes in the goal whenever a restaurant match could not be found for the selected constraints.

For training LSTM-based user simulation models we delexicalise the specific values of the informable slots (food type, name, area, price range; e.g. italian becomes the predefined token `<foodtype>` and north is changed to `<area>`). Thus, the first model learns to generate delexicalized responses. These are later replaced by the real string values, following what is

Table 4.1 Information about the DSTC-2 train, test and validation set.

	Train	Test	Validation
Dialogue count	1612	1117	506
Turns	11677	9890	3934
Goal change (%)	40.1	37.0	44.5

stated in the current user goal. Delexicalisation is no longer needed with transformer models. We also preprocess the data to fix obvious misspellings in the annotations.

4.2 Evaluation metrics used

Due to time constraints we only evaluate our simulator through direct metrics and leave indirect evaluation as future work, as it would require significant integration effort. A wide range of metrics can be found for the task of language generation, where we differentiate between word, character or embedding-based approaches. We consider word-based metrics, particularly BLEU and ROUGE score due to their popularity (Sai et al., 2022). Moreover, these are the metrics used in (Kim and Lipani, 2022), which implements a multitask simulator based on a transformer model and capable of generating utterances.

Considering these metrics have limitations, we further conduct human evaluations to measure the models' generation quality.

4.2.1 BLEU Score

BLEU (Papineni et al., 2002) considers the ratio of the number of overlapping n-grams with the reference to the total number of n-grams in the hypothesis. This can be seen in Equation 4.1 below:

$$precision_n = \frac{\sum_{n-gram \in hypothesis} Count_{clip}(n-gram)}{\sum_{n-gram \in hypothesis} Count(n-gram)} \tag{4.1}$$

However, as observed in Equation 4.1, the total count of overlapping n-grams in the hypothesis is clipped by the maximum count of that n-gram in any of the reference sentences we compare the hypothesis against. This helps to avoid giving high scores to sentences with repeated words that overlap. For instance, if a particular n-gram appears thrice in the hypothesis, but twice in one reference and once in another reference, then we would consider the overlapping n-gram count as 2 and not 3.

In essence, this clipped count can be defined as:

$$Count_{clip}(n-gram) = min(\text{matched n-gram count}, max_{r \in references}(\text{n-gram count in r})) \tag{4.2}$$

Precision is computed separately for different values of n and then the expected value is calculated as seen in Equation 4.4. Finally, another aspect to take into consideration is the hypothesis length. Precision depends only on its overlap and not on its length, so this metric can display high scores when the simulator generates only a few matching or common words/n-grams. To circumvent short meaningless hypotheses, BLEU includes a brevity penalty term, BP, described in Equation 4.3 below:

$$BP = \begin{cases} 1 & |h| \geq |r| \\ exp(1 - \frac{|r|}{|h|}) & |h| < |r| \end{cases} \tag{4.3}$$

Where $|h|$ and $|r|$ represent the hypothesis and the reference length. The weighted BLEU score is multiplied by this term to get the final formula in Equation 4.4 below:

$$\text{BLEU-N } = BP \cdot exp(\sum_{n=1}^{N} w_n \log(precision_n)) \tag{4.4}$$

This metric's value is within the [0,1] interval, with a value close to the unit signaling the greatest similarity between the hypothesis and the reference sentences, and thus higher quality.

The training, test and validation sets are used as references, but to account for the context-dependent generation, we condition the references available based on the last dialogue context vector. If we only considered the one real answer seen in the data for each instance of the test set, we would be penalizing the simulator for not replicating the data, which is the opposite of what we want to accomplish. By considering more references aside from the one real answer given by the real user, we increase the degree of variability that is deemed acceptable, as other responses that occurred in the same context should be valid.

For each hypothesis we obtain its best cumulative BLEU-1 to BLEU-4 scores against all possible references, and we report the average across all hypotheses for the test set.

4.2.2 ROUGE

ROUGE (Lin, 2004) was originally proposed for summarization, however its use has been generalized to other NLP tasks.

It has several variants, but we focus on ROUGE-N. Like BLEU-N, ROUGE-N counts n-gram matches between the hypothesis and reference, however, it's a recall-based metric whereas BLEU is precision-based. It is defined as seen in Equation 4.5:

$$\text{ROUGE-N} = \frac{\sum_{n-gram \in reference} Count_{match}(n - gram)}{\sum_{n-gram \in reference} Count(n - gram)} \tag{4.5}$$

Where $Count_{match}(n - gram)$ is the number of n-grams co-ocurring in hypothesis and reference. In essence, BLEU measures how much the words (n-grams) in the machine generated utterances appeared in the references and ROUGE measures how much the words in the references appeared in the generated utterances.

As with BLEU, we report the average of the best ROUGE-1-R and ROUGE-2-R scores of all hypotheses[1].

4.2.3 Self-BLEU

Introduced by Zhu et al. (2018), self-BLEU measures the variability in the sequences outputted by a model, by considering each utterance as the hypothesis while the rest are taken as references. Contrarily to BLEU and ROUGE, we seek low scores in this metric, implying that there is little overlap between the generated sequences.

4.2.4 Other Direct Metrics

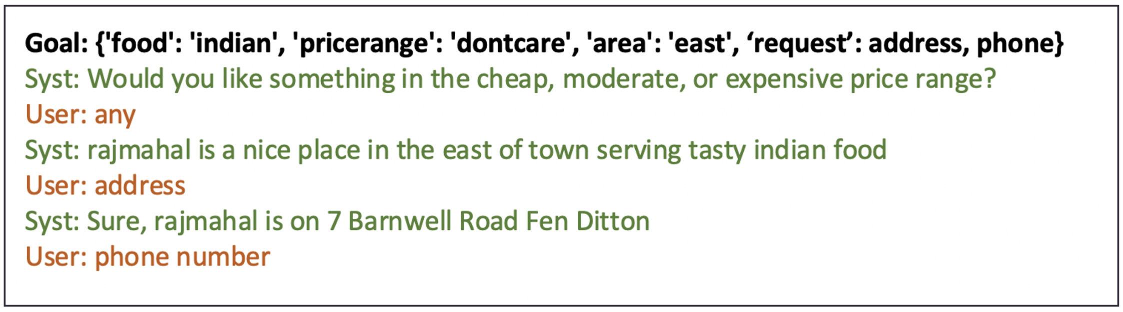
We additionally consider utterance length as a straightforward metric to measure diversity, and additionally F-score values are reported to evaluate semantics in the multitask learning approach.

4.2.5 Human Evaluation

Human evaluation aims to measure subjective metrics. Depending on the task and the goal of the evaluation, different evaluators can be considered (experts/non-experts, paid/unpaid...). Due to time constraints, we resort to a smaller-scale evaluation and leave crowd-sourcing for future work. We create a survey containing dialogues and fragments of dialogues coming from different models and real transcripts (including in all the user goal). Volunteers are unaware of the existence of different models and are asked to read the samples and give two rating for the user behaviour (see Figure 4.1 for an example). The average rating is reported. Thus, we focus the evaluation on the following two metrics:

- Coherence: aims to gain insight into the consistency of the user simulator in a dialogue against a system, i.e. determine whether the user is consistent in its actions.
- Naturalness: we aim to measure how human-like the utterances appear. This metric inherently includes aspects like diversity in the vocabulary and grammar used, fluency and repetitiveness.

[1]The final R in ROUGE-R stands for recall.



Goal: {'food': 'indian', 'pricerange': 'dontcare', 'area': 'east', ‘request’: address, phone}
Syst: Would you like something in the cheap, moderate, or expensive price range?
User: any
Syst: rajmahal is a nice place in the east of town serving tasty indian food
User: address
Syst: Sure, rajmahal is on 7 Barnwell Road Fen Ditton
User: phone number

Fig. 4.1 Sample dialogue used for human evaluation.

Chapter 5

Experiments

In this section we describe all experiments, firstly for the LSTM approach and secondly for the transformer-based. In both cases, we first train and evaluate an MLE model that acts as a baseline. Then, in an effort to alleviate the instability of adversarial training we perform several modifications to the initial model, reward and loss function, and measure their effect in performance.

5.1 LSTM System

5.1.1 MLE Baseline Systems

Firstly, experiments testing different configurations for a baseline system, trained with the traditional MLE approach, were carried out. The optimal configuration found will be used to compare the utterances generated with GAN-based models. We specifically experimented with different hidden layer sizes (we end up using a size of 32) and tested the effect of adding attention. Results in the test set of the best performing models are shown in Table 5.1.

Table 5.1 BLEU scores for different models trained with ML.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4
No attention	0.721	0.298	0.212	0.188
Attention (teacher-forcing ratio = 1)	0.760	0.489	0.414	0.399
Attention (teacher-forcing ratio = 0.5)	**0.760**	**0.491**	**0.426**	**0.405**

From the results in Table 5.1 we conclude that attention helped to increase the BLEU score and led to more sensible utterances per our judgement. This improvement can be explained by the enhancement in dialogue representation that attention brings, as the decoder can dynamically refer to all previous contexts, whereas without attention the model can be biased to the final contexts which in our case was proven to not be optimal.

Differences in the teacher-forcing ratio were not significant in the case of ML, but given the improvement of performance seen with a ratio of 0.5, which potentially relates to the reduction of exposure to the ground-truth forcing the model to learn even without it, we consider the architecture with attention and this ratio for the single-task adversarial experiments.

5.1.2 Adversarial Experiments

Subsequently, we test several settings for the adversarial approach, starting from the best configuration found in the MLE experiments. Specifically, we explore different levels of pre-training, the use of rewards at every generation step (what d'Autume et al. (2019) call dense rewards), batch size and modifications to the discriminator, reward and loss. We keep the learning rate deemed optimal in the semantic-level simulator (0.001), as we experimented with lower rates, but saw worse results.

MLE Pre-training and Dense Rewards

One common strategy in adversarial training for text generation is to start training with ML. Following this, we train the generator and discriminator via MLE for some predefined epochs. Subsequently, we switch to adversarial training for 20 epochs.

Figure 5.1 shows the effect of pre-training during adversarial training by comparing the BLEU-1 scores obtained in the validation set after each epoch. As observed, pre-training has a significant effect and, as expected, models with more pre-training start from a BLEU score much higher than those with less pre-training.

Additionally, following suggestions in (d'Autume et al., 2019; Li et al., 2017) we test the effect of using "dense rewards" during training, by giving rewards to subsequences instead of the whole sequence. While using dense rewards with pre-trained models did not appear to greatly impact performance, this method was a must when no pre-training was performed, as otherwise the model only outputted long, random utterances that received extremely low BLEU scores.

In Figure 5.2 BLEU-2 scores are shown instead. Here the reason for the low scores in the non-pre-trained model could relate to the fact that the generator is learning to output very short utterances (when tested on the test set only 7% of the utterances were longer than one word). Thus, pre-training is necessary to avoid this behaviour. However, a 10 epoch

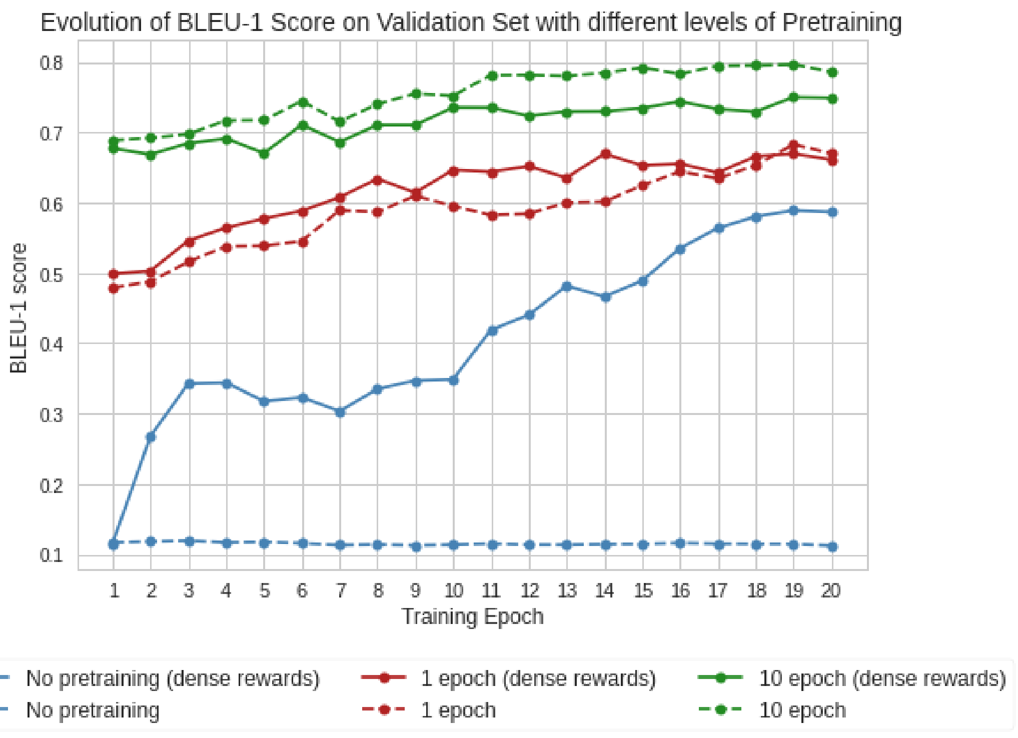


Fig. 5.1 Evolution of BLEU-1 scores on the validation set during adversarial training.

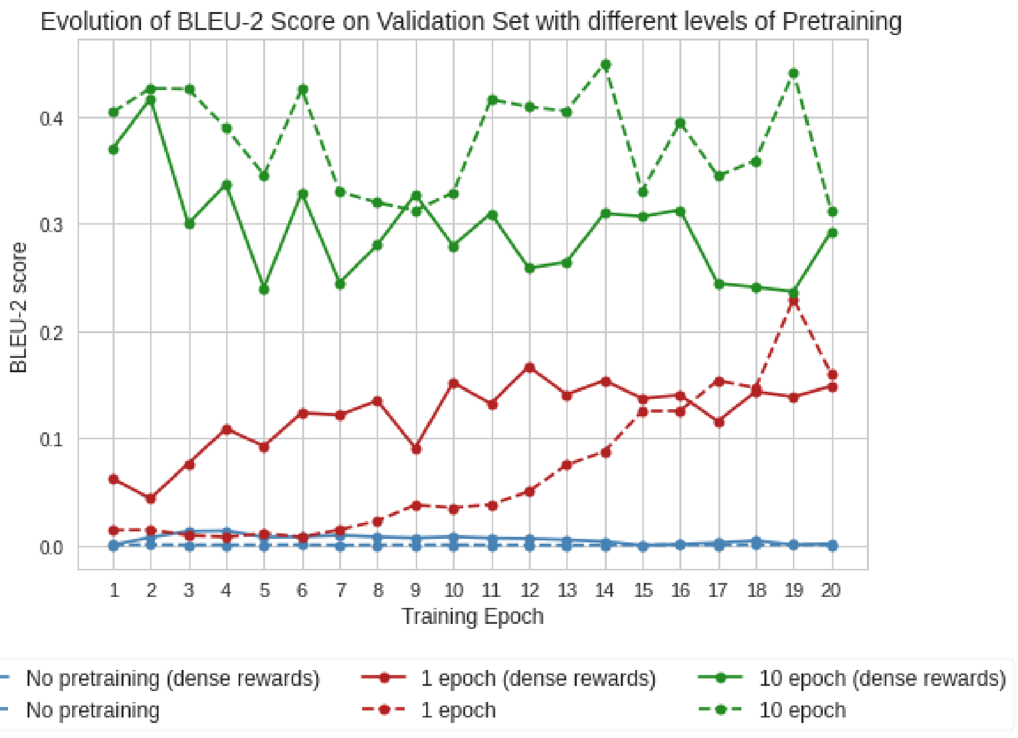


Fig. 5.2 Evolution of BLEU-2 scores on the validation set during adversarial training.

pre-trained model displayed unstable behaviour during training, and even degradation in BLEU-2 with dense rewards.

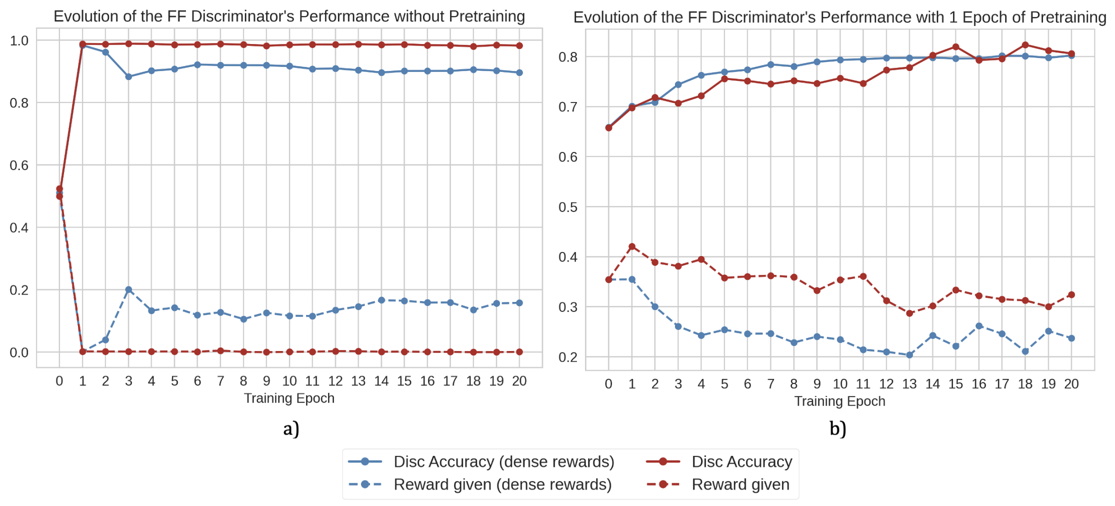


Fig. 5.3 Evolution of the discriminator's performance with and without dense rewards and a) without pre-training, b) with 1 epoch of pre-training.

Moreover, in Figure 5.3 we show the difference in the evolution of the discriminator's performance during adversarial training without pre-training and with one epoch of ML. Both discriminators tend to learn easily what distinguishes real and fake utterances, as observed in the upward trend in accuracy, but without pre-training this is more pronounced. In addition, without pre-training and with normal rewards, the discriminator directly reaches a plateau at very high accuracies, as the generator keeps outputting long, random utterances that are obviously fake.

Table 5.2 BLEU scores of adversarial models on the test set.

No. of pre-training epochs	BLEU-1	BLEU-2	BLEU-3	BLEU-4
0 (normal rewards)	0.099	0.001	-	-
0	0.534	0.003	-	-
1	0.626	0.125	0.001	-
10	**0.726**	**0.261**	**0.310**	**0.339**
MLE baseline	0.760	0.491	0.426	0.405

Table 5.2 displays the BLEU scores in the test set for the best-performing models with different pre-training. As observed, scores are significantly lower than the MLE baseline. Moreover, only the models with several epochs of pre-training have acceptable BLEU-3

and BLEU-4 scores, as the other configurations resulted in an extremely low number of utterances of three or more words, for which scores were, in addition, extremely low. This clearly signals that more than 1 epoch of pre-training is needed. Something outstanding is the high score obtained in BLEU-4 by the 10 epoch pre-trained model. Although compared to the other models this improvement is certainly meaningful, we noticed that over 50% of the utterances of more than four words were the sentence “thank you good bye”, which the system would output accurately in most cases. What we want to convey with this is that, even if pre-training helped to increase quality, the trend observed for adversarial training was that short utterances were predominant, something ultimately undesirable for our purpose of increasing output diversity (see Figure 5.5 for examples of utterances).

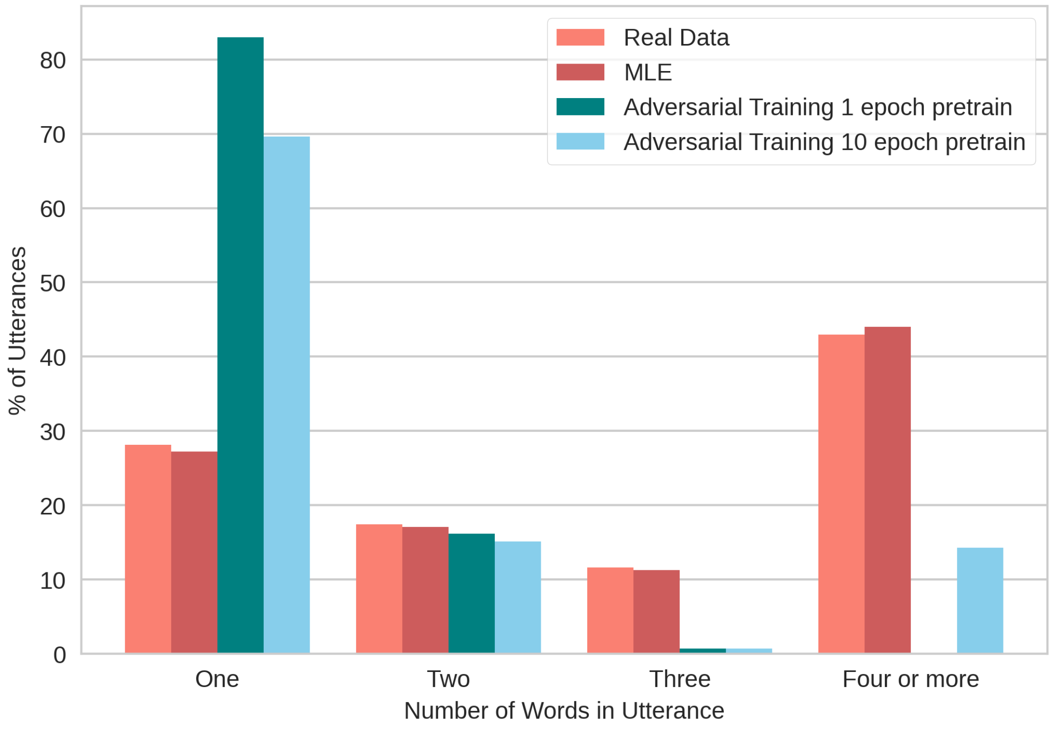


Fig. 5.4 Distribution of utterance length for the different systems and real data distribution.

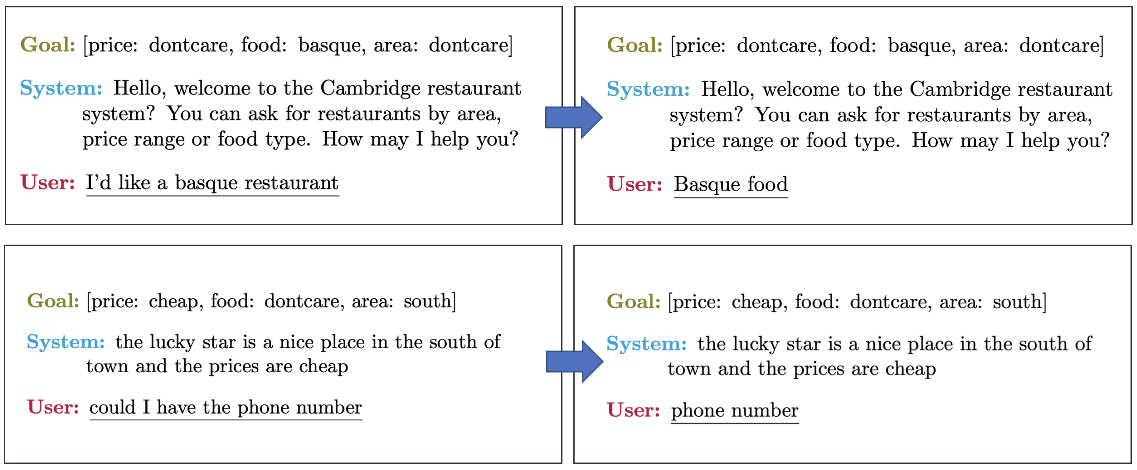
Table 5.3 displays ROUGE scores in the test set of the best-performing models. As expected, increasing pre-training approaches the models to the MLE baseline, although the difference is considerable.

Figure 5.4 includes the generated utterance length distribution for several models and the real data (all adversarial models in this case were trained with dense rewards and from now on, unless otherwise stated, this should be the configuration assumed). As observed, MLE replicates the data distribution but the adversarial models output mostly one-word responses. As mentioned previously, we hypothesize that the peak in the number of utterances in the

Table 5.3 ROUGE-N scores of adversarial models with different pre-training levels on the test set.

Model	No. of pre-training epochs	ROUGE-1	ROUGE-2
Adversarial	1	0.496	0.110
	10	**0.645**	**0.260**
MLE baseline	-	0.781	0.508

four or more category obtained with a greater amount of pre-training is essentially due to the increase in "thank you good bye".



Goal: [price: dontcare, food: basque, area: dontcare]

System: Hello, welcome to the Cambridge restaurant system? You can ask for restaurants by area, price range or food type. How may I help you?

User: I'd like a basque restaurant

Goal: [price: dontcare, food: basque, area: dontcare]

System: Hello, welcome to the Cambridge restaurant system? You can ask for restaurants by area, price range or food type. How may I help you?

User: Basque food

Goal: [price: cheap, food: dontcare, area: south]

System: the lucky star is a nice place in the south of town and the prices are cheap

User: could I have the phone number

Goal: [price: cheap, food: dontcare, area: south]

System: the lucky star is a nice place in the south of town and the prices are cheap

User: phone number

a) Transcript b) Adversarial Model Output

Fig. 5.5 Example utterances with adversarial training pre-trained for 10 epochs compared to the transcripts.

Batch size effect

Aside from experimenting with pre-training and with the use of dense rewards, d'Autume et al. (2019) also suggest increasing the batch size with the purpose of stabilizing convergence, as this encourages a reduction in variance.

Indeed, increasing the batch size to values higher than one (the one previously used) led to less fluctuation in the intermediate BLEU scores on the validation set (see Figure 5.6), but larger batch sizes did not improve performance in a meaningful way, and in fact a preference towards small batch sizes was observed. Following this we keep a batch size of 1. Another noticeable aspect is that batch size seems to have less impact when using dense rewards.

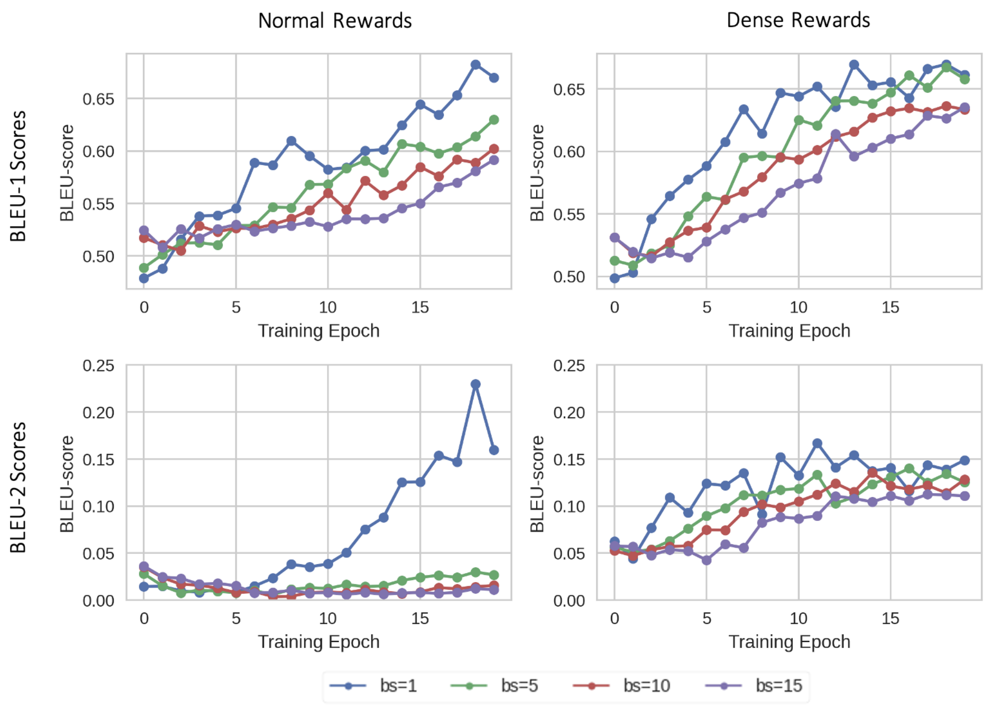


Fig. 5.6 Effect of batch size during training. Smaller batch size seems to give better results, either using normal or dense rewards. In fact, for normal rewards a batch size of 1 is the only one that achieves BLEU-2 scores significantly greater than zero.

Changing the Discriminator

As mentioned in Chapter 3, we also experimented with an LSTM discriminator. We trained the system without pre-training and using dense rewards and it also led to short utterances, more so than with the feed-forward architecture, as seen in Table 5.5. Despite the increase in BLEU-1 score (Table 5.4) this extreme behaviour is undesirable, so we keep the initial configuration for further experimentation.

Table 5.4 BLEU scores with different discriminators.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4
FF	0.534	0.003	-	-
LSTM	0.850	-	-	-

Table 5.5 Length of the generated utterances with different discriminators.

	One word	Two	Three	Four or more
FF	9218	656	16	-
LSTM	9830	60	-	-

5.1.3 Problem with Adversarial Learning from Scratch

The behaviour seen so far is that adversarial training, even with pre-training, gives the undesirable outcome of short utterances. This relates to the fact that the generator is only exposed to the ground-truth in an indirect manner, through the discriminator's reward, which promotes or discourages the generator's sequences. Then once the generator performs badly for some batches, the discriminator can easily recognize the fake sequences, giving them a low reward that results in the generator getting lost. As explained in (Li et al., 2017), the generator may know that its sequences are bad, but at the same time it is unaware of what sequences are good. The odds of escaping this situation are low as the chances of randomly generating a good response are minimal considering all the possible sequence combinations in our vocabulary. From what we observe in the output sequences, it appears the generator rapidly gets stuck at this low-reward local minimum, and learns in this case that generating shorter utterances diminishes the loss. In consequence, we derive two approaches that would increment the exposure to the data, first by mixing different types of losses, and secondly by training the generator on the ground-truth labels with a constant positive reward.

5.1.4 Modifying the Loss

As a way to alleviate the problem of short utterances we experimented with adding, to some degree, direct exposure to the target sequences, by combining the negative log-likelihood and the adversarial loss. While the adversarial loss is focused on optimizing the output to achieve good rewards from the discriminator, the ML loss aims to capture the real data distribution.

We go one step further, adding a similarity loss based on the BLEU score. However, if only the ground-truth for each sample were used as reference to compute this score, this combined loss may end up emphasizing too much the importance of replicating the data, which in our case is undesirable. To damp this effect, we instead allow for multiple references coming from the same context. In this way we aim to encourage, or at the least to not penalize, output variability. The formula for this loss can be seen in Equation 5.1. All losses are in logarithmic form and the aim is to minimize the combination. As such, $\mathscr{L}_{SIM}$ is defined as the negative logarithm of the BLEU score given to the utterance, so low BLEU scores are penalized.

$$\mathscr{L} = (1-\alpha-\beta)\mathscr{L}_{NLL} + \alpha\mathscr{L}_{ADV} + \beta\mathscr{L}_{SIM} \tag{5.1}$$

Using this combined loss we manage to avoid short utterances which, a priori, should entail a higher degree of variability. A grid search was performed to tune the hyperparameters β and α in Equation 5.1. Table 5.6, 5.7, 5.8 show results in the test set of different combinations for BLEU, ROUGE and self-BLEU scores respectively. Correlation is observed between BLEU-ROUGE scores and an inverse correlation exists between these and the self-BLEU scores. This may indicate that the variability that self-BLEU is detecting could be coming from added randomness in the utterances, instead of from an effective and accurate increase in diversity.

Table 5.6 Effect on BLEU score of different α and β values.

α	β	BLEU-1	BLEU-2	BLEU-3	BLEU-4
0	**0**	**0.760**	**0.491**	**0.426**	**0.405**
0.50	0	0.706	0.423	0.281	0.208
0.50	0.10	0.702	0.436	0.280	0.200
0.30	**0.20**	**0.717**	**0.443**	**0.290**	**0.210**
0.33	0.33	0.674	0.372	0.231	0.163

Out of all these models, that with $\alpha = 0.3$ and $\beta = 0.2$ provides a good trade-off between correctness (BLEU/ROUGE) and variability (self-BLEU) when compared against the baseline MLE ($\alpha = 0$ and $\beta = 0$).

Table 5.7 ROUGE scores with different α and β values.

α	β	ROUGE-1	ROUGE-2
0	**0**	**0.781**	**0.508**
0.5	0	0.690	0.328
0.5	0.1	0.692	0.331
0.3	**0.2**	**0.711**	**0.345**
0.33	0.33	0.683	0.316

Table 5.8 Effect on self-BLEU score of different α and β values.

α	β	Self-BLEU-1	Self-BLEU-2	Self-BLEU-3	Self-BLEU-4
0	0	0.992	0.922	0.862	0.798
0.50	0	0.989	0.890	0.651	0.462
0.50	0.10	0.991	0.885	0.655	0.453
0.30	0.20	0.992	0.886	0.675	0.486
0.33	**0.33**	**0.989**	**0.883**	**0.646**	**0.445**
Real Data		0.995	0.980	0.960	0.927

These models show similar BLEU and ROUGE scores to the 10 epoch pre-trained adversarial model, but this approach manages to stop short utterances, as seen in the utterance length distribution in Figure 5.9. To test whether the utterances generated with the combined loss and this hyperparameter configuration can be considered acceptable in terms of naturalness and coherence, we perform human evaluation, with results shown in Section 5.1.6.

5.1.5 Increasing Exposure to the Ground-truth: Using Constant Positive Rewards

In this section we explore another possibility to improve the model performance through a more direct exposure to the ground-truth. Previously when we were applying teacher-forcing we would feed the decoded sequence to the discriminator to get a reward for it. Now we train the generator with the correct labels, giving it a fixed positive reward whenever we apply teacher-forcing. This is a way to guide the generator to the correct distribution, considering we have seen previously that it is very easy for it to get lost. We set the positive reward to a value of 2 and consider an initial exposure to the ground-truth of 30%, which we decay continuously per epoch up to a minimum of 10%, similarly to the approach taken in (Wu et al., 2021).

Results on the test set for both BLEU (Table 5.9) and ROUGE (Table 5.10) reach values comparable to the MLE baseline, while a decrease in self-BLEU is still observed (Table 5.11). Table 5.11 also shows results in the self-BLEU scores for the *real* data, which gives a sense of the variability observed in it.

Comparing this approach to the previous one (combined loss) we observe that we obtain better results in BLEU-3 and BLEU-4, with a consequent increase in self-BLEU-3 and self-BLEU-4. Moreover, despite the lower BLEU-1 and BLEU-2 compared to the model combining losses, the ROUGE-1 and ROUGE-2 scores increase with this approach, which signals that this model has enhanced recall.

Figure 5.7 additionally shows the discriminator's accuracy decreasing during training, what signals that the generator is indeed managing to improve itself.

Table 5.9 BLEU scores of the models with different adversarial configurations.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Combined Loss	**0.717**	**0.443**	0.290	0.210
Guided + constant rewards	0.710	0.397	**0.373**	**0.360**
MLE baseline	0.760	0.491	0.426	0.405

Table 5.10 ROUGE scores with different adversarial configurations.

	ROUGE-1	ROUGE-2
Combined Loss	0.711	0.345
Guided + constant rewards	**0.726**	**0.420**
MLE baseline	0.781	0.508

Table 5.11 Effect on self-BLEU score of different adversarial configurations.

	Self-BLEU-1	Self-BLEU-2	Self-BLEU-3	Self-BLEU-4
Combined loss	**0.992**	0.886	**0.675**	**0.486**
Guided + constant rewards	0.998	**0.862**	0.760	0.683
MLE baseline	0.992	0.922	0.862	0.798
Real Data	0.995	0.980	0.960	0.927

What is most interesting about this approach is the fact that no MLE is used (neither in a pre-training phase nor in combination in the loss) and yet this purely adversarial model was able to compete against the ML baseline in terms of direct metrics. Moreover, this method represents another way to control the amount of exposure to the labels. If we consider other initial teacher-forcing ratios or decide to not decay it we obtain similar but slightly different results. The design decisions made in our case aimed to reach a balance between BLEU/self-BLEU.

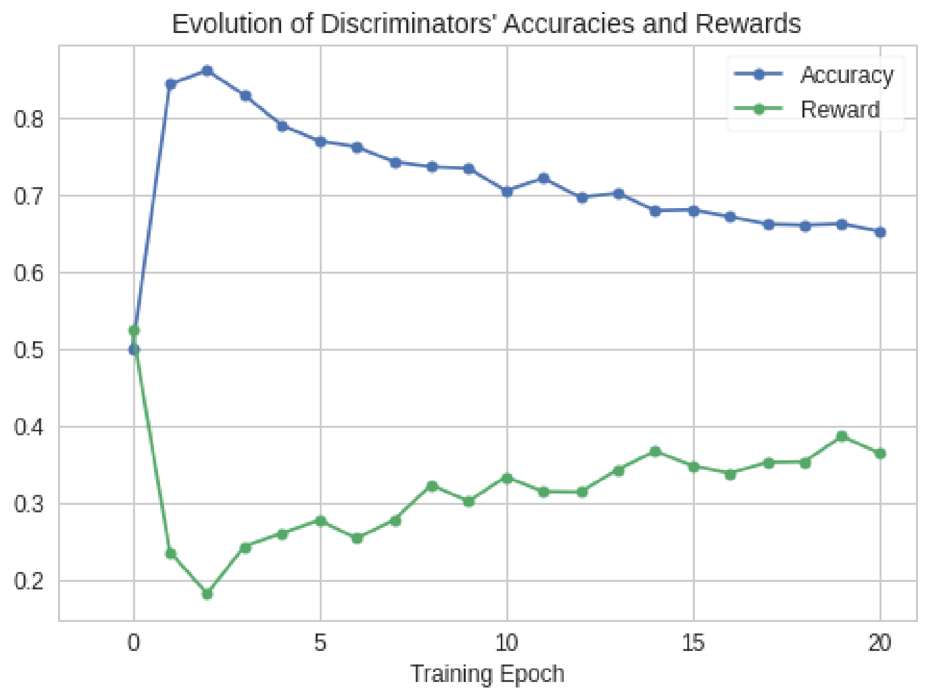


Fig. 5.7 Evolution of the discriminator's accuracy and given rewards.

Goal: [price: moderate, food: dontcare, area: north]

System: Hello, welcome to the Cambridge restaurant system? You can ask for restaurants by area, price range or food type. How may I help you?

User: I want a moderately priced restaurant in the north part of town

Goal: [price: moderate, food: european, area: north]

System: Hello, welcome to the Cambridge restaurant system? You can ask for restaurants by area, price range or food type. How may I help you?

User: Moderate and north of town european

Goal: [price: dontcare, food: basque, area: dontcare]

System: Hello, welcome to the Cambridge restaurant system? You can ask for restaurants by area, price range or food type. How may I help you?

User: Looking for a restaurant, basque food

Goal: [price: dontcare, food: thai, area: dontcare]

System: Hello, welcome to the Cambridge restaurant system? You can ask for restaurants by area, price range or food type. How may I help you?

User: Im looking for thai food

Fig. 5.8 Example utterances.

These utterances are also tested with human evaluation in the following section and the utterance length distribution of this approach is also included in Figure 5.9. Some examples showing the user responding differently to the system's first prompt are shown in Figure 5.8.

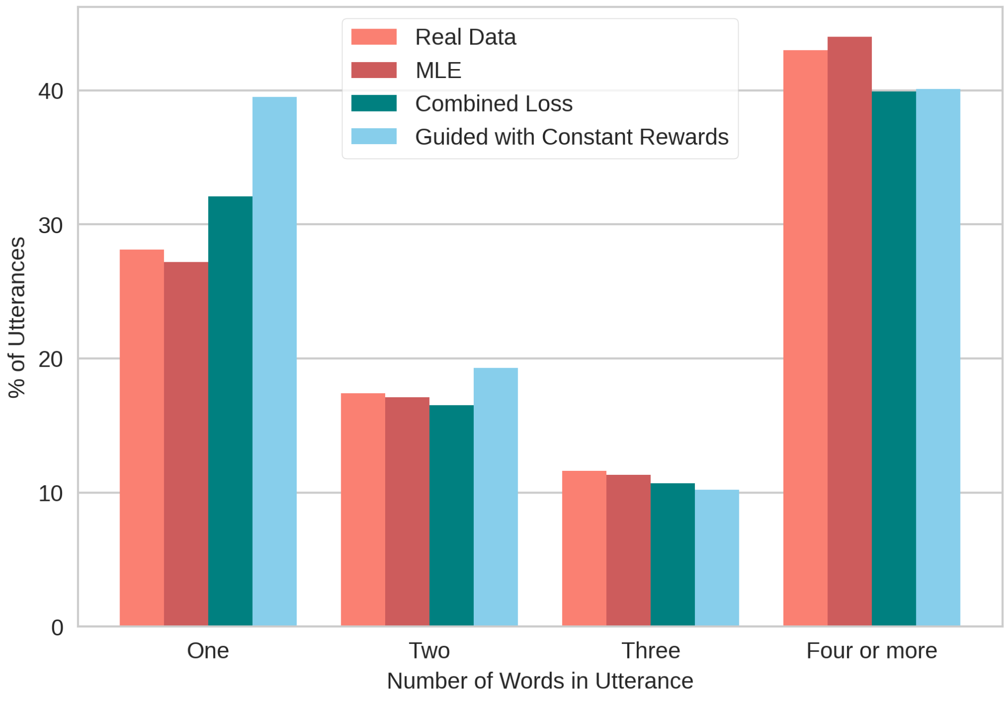


Fig. 5.9 Utterance length distribution for the two modifications proposed, along with that of the MLE baseline and real data for comparison purposes. As observed, both of the proposed approaches led to more reasonable distributions than those obtained through the standard adversarial framework.

5.1.6 Human Evaluation Results

Human evaluation was performed to measure the degree of coherence and naturalness of the outputs. This would determine whether the decrease in self-BLEU related entirely to added randomness.

We sample dialogues from the combined-loss model, the model guiding the generator with constant positive rewards, the baseline MLE model and the real data. There are a total of 36 samples in the survey, 9 in each category. A total of 6 users rated each dialogue in terms of coherence and naturalness, on a 6 point Likert scale. Table 5.12 displays the average ratings.

As expected the real data received the highest ratings in both categories but as can be seen, all models show an acceptable level of coherence and naturalness. Interestingly, the two

Table 5.12 Average ratings from the human evaluation (54 ratings per model).

	Coherence	Naturalness	BLEU-1	BLEU-4	ROUGE-1	ROUGE-2
Real data	**5.15**	**4.53**	-	-	-	-
MLE	4.31	3.55	**0.760**	**0.405**	**0.781**	**0.508**
Combined Loss	**4.45**	3.35	0.717	0.210	0.711	0.345
Guided + const rewards	4.35	**3.71**	0.710	0.360	0.726	0.420

models that were trained adversarially achieved similar ratings to the MLE baseline. Figure 5.10 further visualizes the results for the samples evaluated. For each model, it shows the distribution of the ratings of the sample dialogues. In this plot we observe that coherence in the generated dialogues tends to be higher than naturalness. Moreover, models' naturalness seems to be more variable, which makes sense considering the greater subjectivity of this metric. One outstanding aspect is the presence of several outliers in the coherence of the combined loss model. As we have little data points it would be required to get more data to confirm this, but this plot could be signalling that coherence is very consistent in this model, while naturalness is still variable (although shifted towards greater values). If more data proved this, this model's characteristic would be helpful, as having consistently-coherent utterances that can be more or less natural could give a beneficial variability for training the dialogue system. Nevertheless, the coherence of the guided adversarial model is greater (even greater than MLE) and we still see variability (resembling a normal distribution) in naturalness, always within acceptable levels. As a result, this initial evaluation seems to convey that this model is preferable.

In addition, the final goal of the user simulator is to push the dialogue system to explore the policy state-space by presenting to it diverse prompts to decipher, and possibly this could be achieved with generally sensible but occasionally noisy utterances, like the ones that are likely being achieved by the models presented.

Lastly, we added a subset of the direct metrics (BLEU/ROUGE) to check consistency between them and human evaluation. We observe that MLE and the guided model achieved higher values in these metrics, and also seem to receive similar coherence and naturalness ratings (see Figure 5.10 and Table 5.12). The combined loss model had lower BLEU/ROUGE scores and also lower naturalness. Surprisingly it obtained high coherence, although the increased presence of outliers may be influencing this.

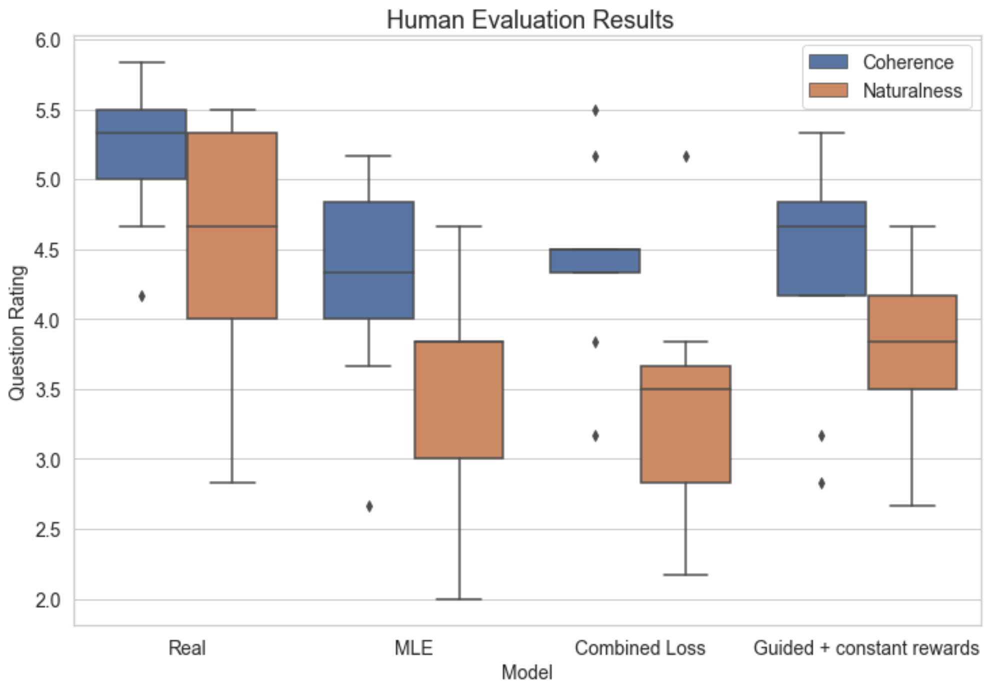


Fig. 5.10 Boxplots of the results of each model's samples (for coherence and naturalness).

5.1.7 Multitask Learning (MTL) Experiments

The experiments presented so far have been with models trained for the single task of utterance generation. In this section, we explore a multitask scheme which also involves dialogue act generation.

MTL-MLE

We again begin training with ML, with results for the two proposed MTL methods (combined vs alternating loss) shown in Table 5.13.

As observed, combining the losses led to small increases in the BLEU scores, although we did not observe the same positive transfer from word to semantic generation, as given by the decrease in F-score. On the other hand, the alternating approach led to performance degradation in both tasks, and increasing the batch size damaged performance even further. The word-focus model aimed to increase the focus on the word task by dedicating more batches to it than to semantic generation. However, this approach did not improve the BLEU scores, and made the F-scores significantly worse.

On the whole, performing multitask learning with ML showed evidence that improvement in performance can potentially be achieved. We now test if this can also aid the adversarial framework, without the need to modify the reward or the training procedure as done in previous sections.

Table 5.13 Semantic and word evaluation with MTL-MLE.

Model	Batch Size	F-score	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Combined Loss	1	0.440	**0.772**	0.478	**0.454**	**0.453**
Alternating	1	0.415	0.707	0.430	0.381	0.360
Alternating (word focus)	1	0.009	0.700	0.420	0.384	0.324
Alternating	15	0.276	0.614	0.292	0.167	0.145
Single-task baseline	1	**0.530**	0.760	**0.491**	0.426	0.405

MTL-ADV

Table 5.14 Semantic and word evaluation with MTL-ADV.

Model	Batch Size	F-score	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Combined Loss	1	**0.365**	0.099	0.0006	-	-
Alternating	1	0.313	**0.585**	**0.007**	-	-
MTL-MLE baseline	1	0.440	0.772	0.478	0.454	0.453

Results for both approaches with adversarial training can be seen in Table 5.14. Combining losses from both tasks gave the best results with ML training, but with adversarial training we find that, while improvement is observed in the semantic task, no learning seems to occur for utterance generation. Checking the train losses we do observe a downward trend for both tasks but, interestingly, Figure 5.11 b) suggests that learning may not be occurring, as the rewards from the utterance discriminator are consistently small.

We then switched to the alternating approach, which gave more reasonable results (see Figure 5.11 a) and Table 5.14). For this case, we tested with batch sizes of 1 and 10 and found that increasing batch size again damaged performance. Nevertheless, the same short-utterance trend was observed, which explains the low scores in BLEU-2. Finally, performance in the semantic task was also negatively affected, as given by the decrease in F-scores with respect to the adversarial semantic baseline in (Dockes, 2021).

Figure 5.12 sums up the best results obtained in the multitask experiments, for ML and adversarial training.

In conclusion, although the slight increase in the direct metrics tested for the ML model seemed promising, the same improvement was not observed with adversarial training.

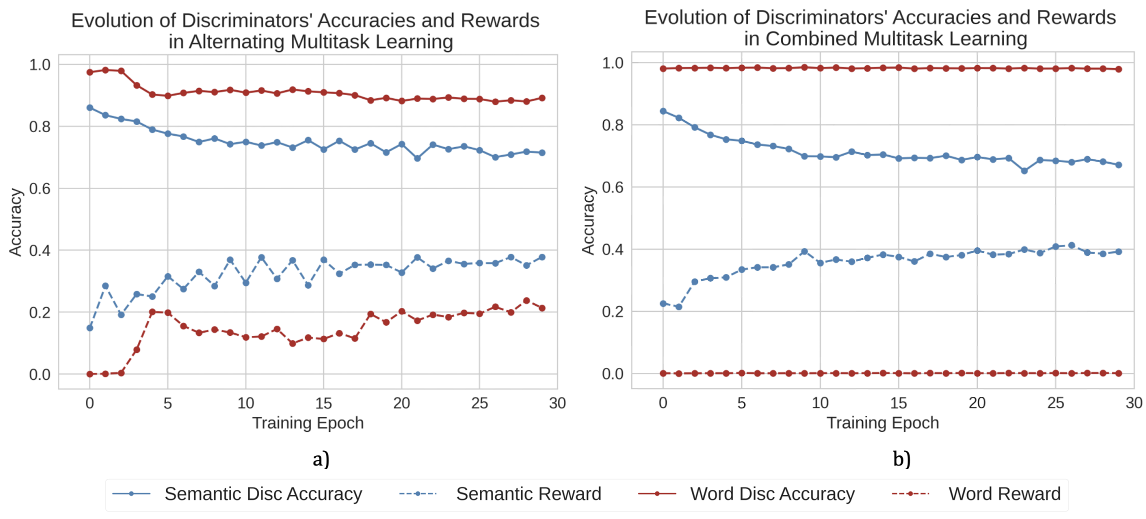


Fig. 5.11 Evolution of the discriminator's accuracy and reward for each task in a) alternating loss MTL and b) combined loss MTL.

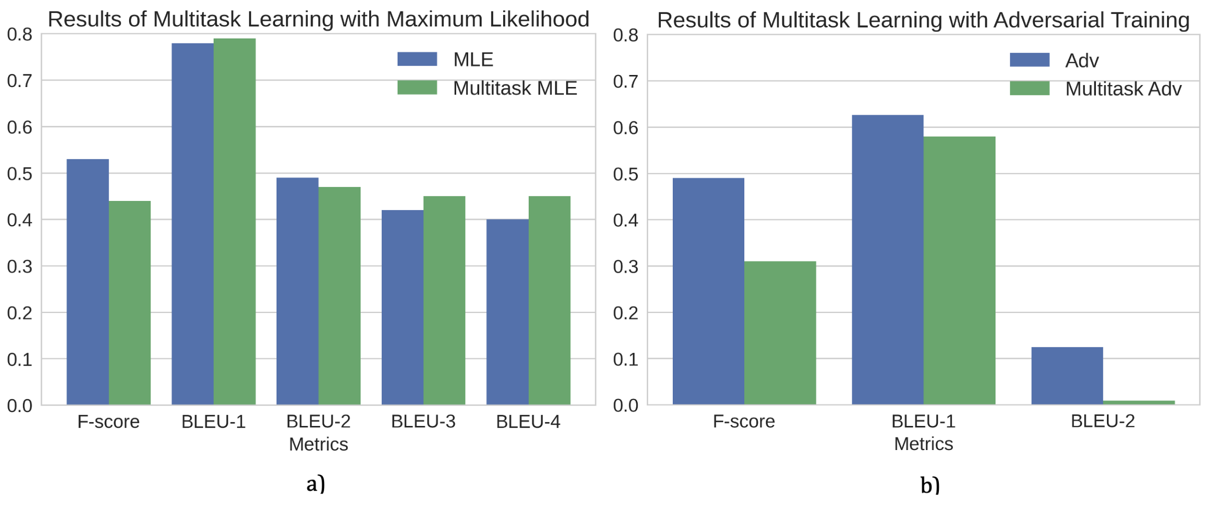


Fig. 5.12 Result for multitask learning with a) MLE vs b) adversarial training approach. Despite the increase in direct metrics with MLE, adversarial training does not appear to benefit from it.

Lastly, given that the modified adversarial loss (with configuration $\alpha = 0.3$ and $\beta = 0.2$) led to improvements in performance and longer utterances than pre-trained adversarial models, we carried out multitask experiments with this loss for utterance generation, and maximum likelihood loss in the semantic task. Results are shown in Table 5.15.

Interestingly, this configuration led to higher F-scores than the MTL-MLE baseline, but BLEU scores still decreased. The same behaviour was observed when using the other successful adversarial approach (guided generator with constant positive rewards) to train

Table 5.15 Results for Alternating-MTL with batch size of 1, ML loss in the semantic task and combined loss ($\alpha = 0.3$, $\beta = 0.2$) in word task

	F-score	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Multitask	**0.463**	0.710	0.299	0.227	0.186
Word Task	-	0.717	0.443	0.290	0.210
Multitask MLE	0.440	**0.772**	**0.478**	**0.454**	**0.453**

word generation and ML for semantics. Nevertheless, these results suggest that with further exploration and hyperparameter fine-tuning, multitask learning could potentially be applied successfully, which would make the integration of the LSTM simulator with a dialogue system more straightforward (although one caveat that would remain a challenge would be to ensure consistency between word and semantic outputs).

5.1.8 Discussion LSTM approach

As our approach to user simulation is new to the best of our knowledge, we lack direct state-of-the-art comparisons. The closest work to ours is the NUS (Kreyssig et al., 2018), considering we use the same form of input and output and generator architecture. However, they train with ML and only report indirect metrics.

Our results are aligned with what has been previously observed in text generation with GANs. The suggestions made by d'Autume et al. (2019) proved to stabilise training, which was also observed in the semantic GAN-based simulator, but their impact on performance seemed arbitrary.

Table 5.16 Summary of results for LSTM models.

	BLEU				Self-BLEU				ROUGE	
	1	2	3	4	1	2	3	4	1	2
MLE baseline	**0.76**	**0.49**	**0.43**	**0.41**	0.99	0.92	0.86	0.80	**0.78**	**0.51**
Adversarial 10 epoch pt	0.73	0.26	0.31	0.34	0.99	**0.83**	0.84	0.82	0.65	0.26
Combined loss	0.72	0.44	0.29	0.21	**0.99**	0.89	**0.68**	**0.49**	0.71	0.35
Guided + const reward	0.71	0.40	0.37	0.36	1.0	0.86	0.76	0.68	0.73	0.42

Overall, allowing the free interaction between generator and discriminator during training resulted in a GAN-based simulator capable of outputting coherent utterances given a context, but the short length of the responses (typically one word) did not allow the output variability desired. The discouragement to generate long utterances that was observed with the adversarial approach is likely due to the dependence of the loss on this length. Despite the efforts carried out to modify the training, we did not observe significant growth of the utterance length. Pre-training aided, but short utterances were still typical (which could explain the low ROUGE scores obtained by the 10 epoch pre-train model seen in Table 5.16, entailing low recall), and it seems plausible that for some contexts, like for instance the closing line "thank you good bye", the model simply learns a formula to use in all cases. This would explain the high self-BLEU of this model for the 3 and 4-gram cases in Table 5.16. Additionally, one possible reason why pre-training is not as helpful as one could expect may relate to the abrupt change from the ML loss to the REINFORCE one, which confuses the generator and can even damage performance.

From what we observed, we believe the root cause of this behaviour comes from the generator-discriminator interaction, as both of them showed the expected performance when individually tested (the generator is successfully trained with MLE and when pre-training the discriminator against a fixed generator we observe that it increases its accuracy, what entails that learning is indeed taking place). It is challenging to balance their individual learning, and, as explained in section 5.1.3, it is easy for the generator to lose track of what would be considered correct utterances if the learning is imbalanced and the discriminator achieves a high accuracy, giving low rewards. With this in mind, we tried guiding the generator more directly, modifying the loss by combining it with MLE loss and our particular similarity loss (based on the BLEU score), and by training it on ground-truth labels with a constant positive reward. Both methods seemed to avoid short utterance generation and gave acceptable BLEU scores in the test set. Furthermore, a small-scale human evaluation signals that the utterances are of adequate quality in terms of coherence and naturalness, comparable (or even better) to those from the MLE model.

Due to time constraints and the lack of availability of a dialogue system fitted to be trained directly with utterances instead of semantics, we were not able to test our system in an indirect manner, i.e. through comparing the performance of a TOD system trained against the user simulator. As a consequence, we can only rely on direct metrics to compare the GAN-based simulator to the MLE one. The latter performed better in these (except on self-BLEU, see Table 5.16), as we expected and as previously seen in the semantic simulator (Dockes, 2021).

Referring back to what was discussed in Chapter 2, the text generation task is particularly tricky, more so if trained in an adversarial style, as the model needs to learn how words connect between themselves aside from the relationship between dialogue context and user response, and all of this added to the need to alleviate typical issues related to GANs in general. We hypothesize that such problems could potentially be solved with a more robust pre-learning step, in which pre-trained embedding models substitute MLE pre-training.

5.2 Transformer models

In this section we move on to experiments performed with the transformer-based approach.

5.2.1 MLE baseline

Following the structure from the LSTM approach, we first include the results obtained when training GPT-2 with ML for 4 epochs (we stopped training once convergence was reached in the validation set). We use a batch size of 1, accumulating gradients for 8 steps, and a learning rate of 10^{-5}. Table 5.17 contains results for the BLEU scores in the test set.

Table 5.17 Results for MLE baseline with transformers.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4
GPT-2 MLE	**0.842**	**0.950**	**0.821**	**0.906**
LSTM MLE	0.760	0.491	0.426	0.405

As observed, using GPT-2 as generator leads to substantially higher scores than the LSTM model, which proves the greater power of the new design to model the data distribution. We now move to the adversarial framework.

5.2.2 Adversarial Experiments

Initially, we tested adversarial training without any modification, but this led to empty utterances being outputted, in line with the short utterances that we saw in the LSTM models. In consequence, we transfer the knowledge gained from this previous approach and experiment with mixing the different losses and forcing the model to generate user responses closer to the ground-truth.

Combined Loss

We employ the same combined loss with the best values for α and β found in the LSTM model (α= 0.3, $\beta = 0.2$). We additionally train a model with the same values for α but no similarity loss (β = 0), to check whether adding the latter is indeed improving the model. Table 5.18 shows the results.

In line with our findings with the LSTM architecture, the models no longer generate very short utterances that resulted in very low BLEU scores (see Figure 5.13). Moreover, from the results in Table 5.18 we can conclude that adding the BLEU-derived similarity loss can help improve the training while reducing MLE impact (and, thus, potentially replication, as

Table 5.18 Effect of combining losses in transformer models.

α	β	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2
0	0	**0.842**	**0.950**	0.821	**0.906**	**0.903**	**0.618**
0.30	0	0.789	0.920	0.813	0.895	0.874	0.601
0.30	0.20	0.795	0.919	**0.824**	0.890	0.867	0.570

supported as well by the slight decrease in self-BLEU seen in Table 5.19). Although BLEU also encourages replication, it does so in a less constrained way, as we compute it against multiple, variable utterances (as described in Chapter 4). In fact, we obtain a model that outperforms the MLE baseline for BLEU-3, although the difference is almost negligible.

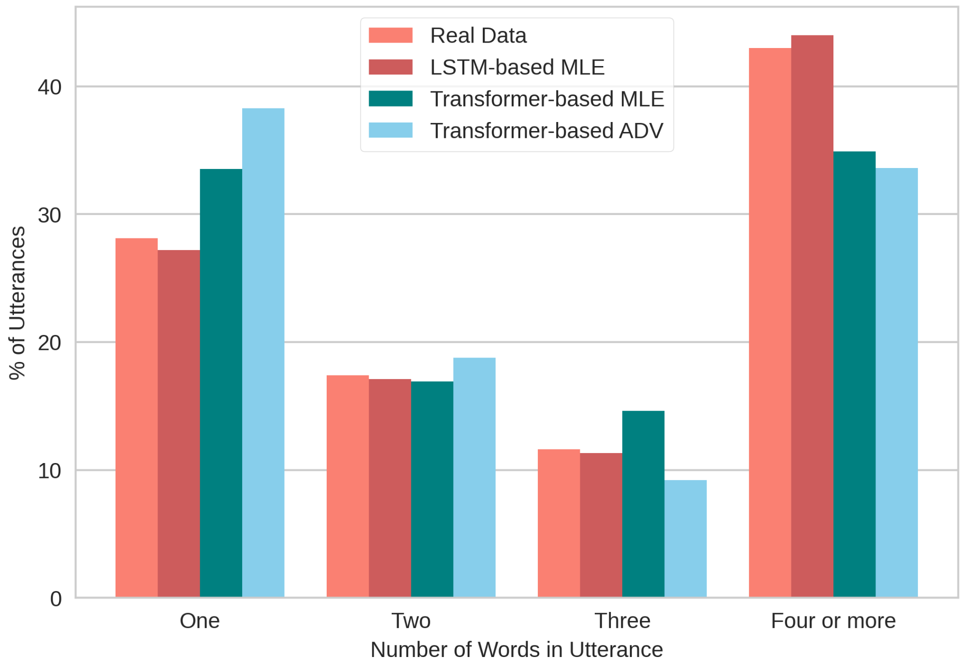


Fig. 5.13 Length distribution for transformer-based models compared against the real data and the LSTM-MLE model. The adversarial model implements the combined loss.

Table 5.19 Effect of loss combination on the self-BLEU scores for transformer models.

α	β	Self-BLEU-1	Self-BLEU-2	Self-BLEU-3	Self-BLEU-4
0	0	0.999	0.998	0.996	0.992
0.30	0	0.999	0.998	0.995	0.995
0.30	**0.20**	**0.999**	**0.995**	**0.994**	**0.992**

The similarity in BLEU and self-BLEU scores leads us to believe that these training methods are likely reaching the same local optima. Nevertheless, although arguably negligible, our model with the loss combination reduced self-BLEU scores compared to the baseline, a small indicator that this method could potentially help to diversify the output. We checked if this was due to the adversarial and similarity loss being of significantly smaller magnitude compared to the ML loss, but found that this was not the case (see Figure 5.14).

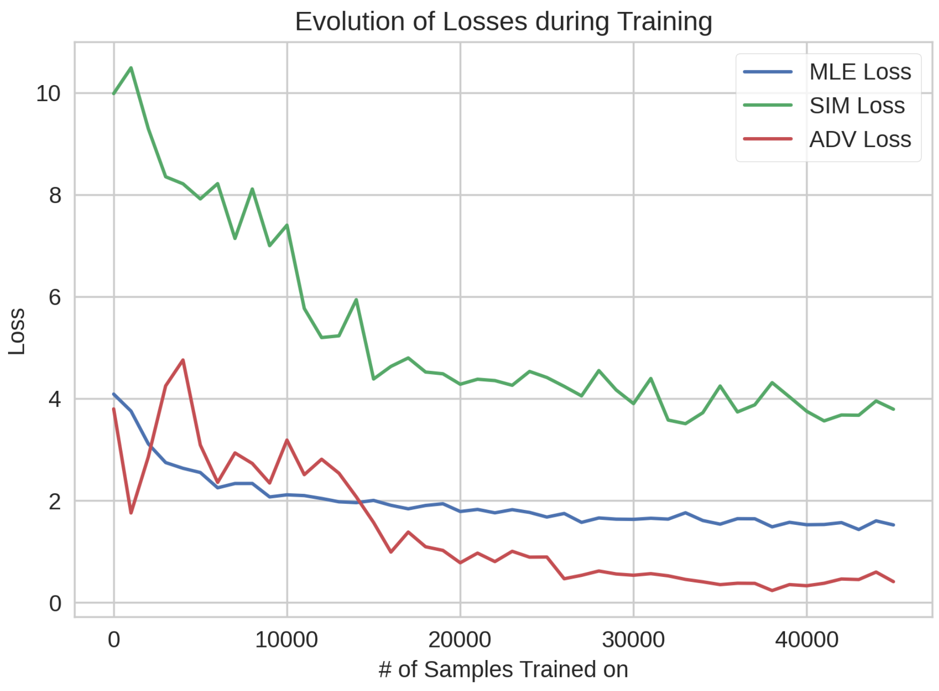


Fig. 5.14 Evolution of the different losses during training, averaged every 1000 samples and using the combined loss with α= 0.3, $\beta = 0.2$.

Alternate training

We maintain the combined loss and experiment with a different approach to training in which, instead of updating discriminator and generator after each batch, we first train the generator with multiple batches while keeping the discriminator fixed, and then switch to training the discriminator with the contexts from the same batches and the corresponding utterances from the updated generator. Specifically we train each for 24 samples (accumulating gradients every 8 samples). As seen in Table 5.20 and Table 5.21 we did not observe meaningful changes, neither in the BLEU and ROUGE nor in the self-BLEU scores.

Table 5.20 Effect of training scheme (combined loss with $\alpha = 0.3$, $\beta = 0.2$).

Training Updates	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2
Simultaneous	0.795	0.918	**0.824**	0.890	0.866	0.570
Alternating	0.784	0.880	0.809	0.880	0.867	0.599
MLE Baseline	**0.842**	**0.950**	0.821	**0.906**	**0.903**	**0.618**

Table 5.21 Effect of training scheme on self-BLEU scores (combined loss with $\alpha = 0.3$, $\beta = 0.2$).

	Self-BLEU-1	Self-BLEU-2	Self-BLEU-3	Self-BLEU-4
Simultaneous	**0.999**	**0.995**	0.994	**0.992**
Alternating	0.999	0.995	**0.991**	0.994
MLE Baseline	0.999	0.998	0.996	0.992

Decoding Strategy

We also explore different decoding methods to generate the sequences during training. Particularly we perform nucleus sampling ($p = 0.9$) and beam search with a beam size of 5. As seen in Table 5.22, we find that, while performing nucleus sampling is detrimental, augmenting the beam size gives enhanced BLEU scores and does not lead to a consistent increase in self-BLEU. In fact, as shown in Table 5.23, self-BLEU scores are reduced with respect to the MLE baseline.

Table 5.22 Effect of decoding strategy on BLEU and ROUGE scores (combined loss with $\alpha = 0.3$, $\beta = 0.2$).

Training Updates	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2
Nucleus Sampling	0.781	0.884	0.799	0.865	0.854	0.562
Greedy	0.795	0.918	0.824	0.890	0.866	0.570
Beam search (n = 5)	**0.866**	0.925	**0.841**	0.901	0.903	0.560
MLE Baseline	0.842	**0.950**	0.821	**0.906**	**0.903**	**0.618**

Table 5.23 Effect of decoding strategy on self-BLEU (combined loss with $\alpha = 0.3$, $\beta = 0.2$).

	Self-BLEU-1	Self-BLEU-2	Self-BLEU-3	Self-BLEU-4
Nucleus	0.999	0.995	0.988	0.990
Greedy	0.999	**0.995**	0.994	0.992
Beam search	**0.999**	0.998	**0.986**	**0.990**
MLE Baseline	0.999	0.998	0.996	0.992

Contrastive Discriminator

We lastly move to a contrastive discriminator. This architecture proved to give good results in the TextGAIL (Wu et al., 2021), as they argue that the sigmoid loss in the binary classifier saturates early on (i.e. the discriminator learns to differentiate samples easily), creating bad rewards for the generator. They propose instead to feed both real and generated user response to the discriminator at the same time, along with the context, for it to model which one is more realistic.

Table 5.24 BLEU and ROUGE scores for different discriminators (combined loss with $\alpha = 0.3$, $\beta = 0.2$).

Training Updates	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2
Binary Classifier	**0.795**	**0.918**	**0.824**	**0.890**	**0.866**	0.570
Contrastive	0.775	0.869	0.801	0.857	0.849	**0.580**

Table 5.25 Self-BLEU (variability) scores for different discriminators (combined loss with $\alpha = 0.3$, $\beta = 0.2$).

Training Updates	Self-BLEU-1	Self-BLEU-2	Self-BLEU-3	Self-BLEU-4
Binary Classifier	0.999	**0.995**	0.994	0.992
Contrastive	**0.998**	0.997	**0.990**	**0.990**

As seen in Table 5.24, using a contrastive discriminator did not lead to enhanced results as in the TextGAIL, which could indicate that the reason of weight behind their results resides in the use of PPO. However, we do notice some decrease in self-BLEU (Table 5.25).

5.2.3 Discussion Transformer Model

The experiments in this section represent one of the first cases of using GPT-2 for word-level user simulation. By training the simulator with ML we achieved very high BLEU scores, considerably greater than the LSTM model (see Table 5.26), but also very high self-BLEU scores, which indicates little diversity in the outputs. The significant reduction on ROUGE-2 could also be signalling this.

Adversarial experiments with transformers were limited due to time constraints. While we did not have time to thoroughly explore different hyperparameter configurations to increase variability, we were able to develop a user simulator that correctly captures the data distribution, as given by the high BLEU scores, but this had to be done through combining the adversarial loss with MLE and our similarity loss, as training adversarially from scratch or using the teacher-forcing technique with constant rewards deployed in the LSTM model led to empty utterances.

Table 5.26 Summary of results for Transformer models.

	BLEU				Self-BLEU				ROUGE	
	1	2	3	4	1	2	3	4	1	2
LSTM MLE	0.760	0.491	0.426	0.405	0.992	0.922	0.862	0.798	0.781	0.508
GPT MLE	0.842	**0.950**	0.821	**0.906**	0.999	0.998	0.996	0.992	0.903	**0.618**
Alternate Training	0.784	0.880	0.809	0.880	0.999	0.995	0.991	0.994	0.867	0.599
Greedy	0.795	0.918	0.824	0.890	**0.999**	**0.995**	0.994	0.992	0.866	0.570
Beam search	**0.866**	0.925	**0.841**	0.901	0.999	0.998	**0.986**	**0.990**	**0.903**	0.560
Contrastive disc	0.775	0.869	0.801	0.857	0.998	0.997	0.990	0.990	0.849	0.580

The closest model that we can compare it with is the multitask user simulator presented by Kim and Lipani (2022), who use the T5 transformer model and present results for the single task of utterance generation in terms of BLEU and ROUGE scores. They only report BLEU-1 and BLEU-4 for this task, showing results considerably lower than what we obtained in our experiments. We believe such a big gap could be due to the different nature of the data (they use other datasets like MultiWOZ), considering DSTC-2 has a smaller vocabulary and less variation.

Overall, these models reached a behaviour very similar to MLE and in all cases output diversity was restricted, as given by the high self-BLEU scores, although for some configurations we manage to slightly decrease them compared to the MLE baseline.

Nevertheless, we have established a system that allows to regulate the amount of adversarial training to include, and with further exploration it could be possible to further increase its output diversity.

Chapter 6

Conclusion

6.1 Summary

In this dissertation we have described the implementation of a word-level neural user simulator. Starting from a semantic-level simulator, we have tested the suitability of the previous LSTM model to the word generation task. To enhance its performance, we introduced several modifications, including most importantly an attention mechanism and changes to the discriminator (particularly in the input encoding). The MLE-simulator with attention was proven to perform significantly better in terms of direct metrics. Moreover, an MTL-scheme merging semantic and word generation was proposed, and in the case of MLE it increased BLEU scores.

In the adversarial setting, we tested several strategies shown to tackle the typical problems in text generation with GANs, and although they seemed to help to some extent, we found that jointly training generator and discriminator was still challenging. Although convergence could be reached, in most cases it led to a model generating generally coherent but extremely short utterances. We believe the most challenging part resides in the loss used to optimize the generator, which is directly dependent on the reward. To diminish this dependence we combined the loss with MLE, which gave promising results. Moreover, we tried guiding the generator further by training it on the ground-truth with a constant positive reward. Both these approaches led to acceptable BLEU and ROUGE scores and to more natural utterances (as established through human evaluation), while a decrease in self-BLEU was achieved.

Moreover,we established the basis for a transformer-based user simulator that could potentially entail more promising results in terms of output diversity. Experiments carried out allowed us to see that training in an adversarial manner from scratch is challenging, as was the case with the LSTM-based models. Better utterances (with comparable or even improved BLEU scores to the MLE baseline) can be obtained when applying the techniques designed to increase the exposure to the real data distribution, but in the case of transformers these utterances tended to be invariable, seeming like the model learnt a formula for each context. Although this model did not achieve our diversity expectations, we obtained significant increases in terms of BLEU and ROUGE scores, and with some configurations we observed a decrease in self-BLEU scores compared to the MLE baseline.

In essence, after seeing the effect in both models, we can conclude that some exposure to the ground-truth can be beneficial, or even necessary in some cases, to achieve good results with conditional adversarial training in text generation. Applying this, we were able to generate relatively coherent and natural utterances that had slightly increased variability which we believe could be beneficial to enhance training of systems' policies.

6.2 Future Work

Due to the advantages that the transformer-based model represents it would be more significant to, in the future, focus on advancing this particular architecture.

In this case, it would be interesting to evaluate the system's utterances using a method similar to BERTscore, where the information contained in word embeddings is used to measure similarity between reference and hypothesis. It would also be interesting to base our similarity loss in this metric rather than BLEU (to account not only for grammatical and syntactical variation, but also consider synonymy).

Augmenting the dataset by including synonyms to increase variability, using for instance WordNet-based synonym replacement, would be of interest, as perhaps the utterances in the DSTC-2 dataset are not the most appropriate for this task, considering their high self-BLEU scores. Using other datasets such as MultiWOZ could also be meaningful.

Additionally, due to memory constraints, we were not able to use large pre-trained transformer models (GPT-2 medium or large) which could potentially lead to more diverse vocabulary being used, so it would be interesting to explore this. Moreover, a more in-depth hyperparameter tuning could give enhanced results.

Although the results obtained with direct evaluation are insightful, it is likely that MLE performs better in these corpus-based metrics like BLEU or ROUGE, as it is aiming to replicate the data, while the purpose of adversarial training is to introduce some level of variability into the outputs. As a result, what is perhaps the most meaningful step to perform, would be to interface the current user simulator with a dialogue system to corroborate if it, indeed, leads to better policies, as observed in the semantic-level simulator.

References

Ai, H., Tetreault, J., and Litman, D. (2007). Comparing user simulation models for dialog strategy learning. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 1–4, Rochester, New York. Association for Computational Linguistics.

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (GANs). In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 224–232. PMLR.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1171–1179, Cambridge, MA, USA. MIT Press.

Bingel, J. and Søgaard, A. (2017). Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain. Association for Computational Linguistics.

Budzianowski, P. and Vulić, I. (2019). 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. Association for Computational Linguistics.

Caruana, R. (1993). Multitask learning: A knowledge-based source of inductive bias. In *ICML*.

Chandramohan, S., Geist, M., Lefèvre, F., and Pietquin, O. (2011). User simulation in dialogue systems using inverse reinforcement learning. In *INTERSPEECH*.

Che, T., Li, Y., Zhang, R., Hjelm, R. D., Li, W., Song, Y., and Bengio, Y. (2017). Maximum-likelihood augmented discrete generative adversarial networks.

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

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA. Association for Computing Machinery.

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

Cuayahuitl, H., Renals, S., Lemon, O., and Shimodaira, H. (2005). Human-computer dialogue simulation using hidden markov models. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 290–295.

d'Autume, C. d. M., Rosca, M., Rae, J., and Mohamed, S. (2019). *Training Language GANs from Scratch*. Curran Associates Inc., Red Hook, NY, USA.

de Rosa, G. H. and Papa, J. a. P. (2021). A survey on text generation using generative adversarial networks. *Pattern Recogn.*, 119(C).

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

Dockes, C. (2021). Building a conversational user simulator using generative adversarial networks. Master's thesis, University of Cambridge.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China. Association for Computational Linguistics.

Eckert, W., Levin, E., and Pieraccini, R. (1997). User modeling for spoken dialogue system evaluation. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 80–87.

El Asri, L., He, J., and Suleman, K. (2016). A sequence-to-sequence model for user simulation in spoken dialogue systems.

Eshky, A. (2014). Generative probabilistic models of goal-directed users in task-oriented dialogs. Master's thesis, University of Edinburgh.

Gašić, M. and Young, S. (2014). Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.

Georgila, K., Henderson, J., and Lemon, O. (2005). Learning user simulations for information state update dialogue systems. pages 893–896.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA. MIT Press.

Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., and Wang, J. (2017). Long text generation via adversarial training with leaked information.

Gur, I., Hakkani-Tür, D. Z., Tür, G., and Shah, P. (2018). User modeling for task oriented dialogues. *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 900–906.

Henderson, M., Thomson, B., and Williams, J. D. (2014). The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hou, Y., Fang, M., Che, W., and Liu, T. (2019). A corpus-free state2seq user simulator for task-oriented dialogue. *ArXiv*, abs/1909.04448.

Hu, Z., Turki, T., and Wang, J. T. L. (2020). Generative adversarial networks for stochastic video prediction with action control. *IEEE Access*, 8:63336–63348.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org.

Jabbar, A., Li, X., and Omar, B. (2021). A survey on generative adversarial networks: Variants, applications, and training. *ACM Comput. Surv.*, 54(8).

Jung, S., Lee, C., Kim, K., Jeong, M., and Lee, G. G. (2009). Data-driven user simulation for automated evaluation of spoken dialog systems. *Computer Speech and Language*, 23(4):479–509.

Ke, P., Huang, F., Huang, M., and Zhu, X. (2019). ARAML: A stable adversarial training framework for text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4271–4281, Hong Kong, China. Association for Computational Linguistics.

Keizer, S., Gašić, M., Jurčíček, F., Mairesse, F., Thomson, B., Yu, K., and Young, S. (2010). Parameter estimation for agenda-based user simulation. In *Proceedings of the SIGDIAL 2010 Conference*, pages 116–123, Tokyo, Japan. Association for Computational Linguistics.

Kim, T. E. and Lipani, A. (2022). A multi-task based neural model to simulate users in goal-oriented dialogue systems.

Kreyssig, F., Casanueva, I., Budzianowski, P., and Gašić, M. (2018). Neural user simulation for corpus-based policy optimisation of spoken dialogue systems. pages 60–69.

Kusner, M. and Hernández-Lobato, J. (2016). Gans for sequences of discrete elements with the gumbel-softmax distribution.

Levin, E., Pieraccini, R., and Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies. *Speech and Audio Processing, IEEE Transactions on*, 8:11 – 23.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., and Jurafsky, D. (2017). Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169, Copenhagen, Denmark. Association for Computational Linguistics.

Li, X., Lipton, Z., Dhingra, B., Li, L., Gao, J., and Chen, Y.-N. (2016). A user simulator for task-completion dialogues.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Lin, H.-c., Lubis, N., Hu, S., van Niekerk, C., Geishauser, C., Heck, M., Feng, S., and Gasic, M. (2021). Domain-independent user simulation with transformers for task-oriented dialogue systems. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 445–456, Singapore and Online. Association for Computational Linguistics.

Lin, K., Li, D., He, X., Zhang, Z., and Sun, M.-T. (2017). Adversarial ranking for language generation. *Advances in neural information processing systems*, 30.

Liu, B. and Lane, I. R. (2017). Iterative policy learning in end-to-end trainable task-oriented neural dialog models. *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 482–489.

Liu, C.-W., Lowe, R., Serban, I., Noseworthy, M., Charlin, L., and Pineau, J. (2016). 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. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Martínez Alonso, H. and Plank, B. (2017). When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 44–53, Valencia, Spain. Association for Computational Linguistics.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *ArXiv*, abs/1411.1784.

Ng, A. Y. and Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML ’00, page 663–670, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Nie, X., Lin, Z., Huang, X., and Zhang, Y. (2019). *Graph Neural Net-Based User Simulator*, pages 638–650.

Norouzi, M., Bengio, S., Chen, z., Jaitly, N., Schuster, M., Wu, Y., and Schuurmans, D. (2016). Reward augmented maximum likelihood for neural structured prediction. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Papangelis, A., Wang, Y.-C., Molino, P., and Tür, G. (2019). Collaborative multi-agent dialogue model training via reinforcement learning. In *SIGdial*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Pietquin, O. (2006). Consistent goal-directed user model for realisitc man-machine task-oriented spoken dialogue simulation. In *2006 IEEE International Conference on Multimedia and Expo*, pages 425–428.

Pietquin, O. and Hastie, H. (2013). A survey on metrics for the evaluation of user simulations. *The Knowledge Engineering Review*, 28(1):59–73.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ratliff, L. J., Burden, S. A., and Sastry, S. S. (2013). Characterization and computation of local nash equilibria in continuous games. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 917–924.

Rieser, V. and Lemon, O. (2006). Cluster-based user simulations for learning dialogue strategies. In *Proc. Interspeech 2006*, pages paper 1127–Wed2WeS.1.

Rossignol, S., Pietquin, O., and Ianotto, M. (2011). Training a BN-based user model for dialogue simulation with missing data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 598–604, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks.

Sai, A. B., Mohankumar, A. K., and Khapra, M. M. (2022). A survey of evaluation metrics used for nlg systems. *ACM Comput. Surv.*, 55(2).

Schatzmann, J., Stuttle, M., Weilhammer, K., and Young, S. (2005). Effects of the user model on simulation-based learning of dialogue strategies. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 220–225.

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

Schatzmann, J., Weilhammer, K., Stuttle, M. N., and Young, S. J. (2006). A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21:97 – 126.

Schatzmann, J. and Young, S. (2009). The hidden agenda user simulation model. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):733–747.

Scheffler, K. and Young, S. (2000). Probabilistic simulation of human-machine dialogues. volume 2, pages II1217 – II1220 vol.2.

Scheffler, K. and Young, S. (2002). Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. pages 12–19.

Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., and Staiano, J. (2020). Coldgans: Taming language gans with cautious sampling strategies. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18978–18989. Curran Associates, Inc.

Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., and Bengio, Y. (2017). A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3295–3301. AAAI Press.

Shi, W., Qian, K., Wang, X., and Yu, Z. (2019). How to build user simulators to train RL-based dialog systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1990–2000, Hong Kong, China. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*.

Tseng, B.-H., Dai, Y., Kreyssig, F., and Byrne, B. (2021). Transferable dialogue systems and user simulators. In *ACL/IJCNLP (1)*, pages 152–166.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256.

Wu, Q., Li, L., and Yu, Z. (2021). Textgail: Generative adversarial imitation learning for text generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14067–14075.

Xu, J., Ren, X., Lin, J., and Sun, X. (2018). Diversity-promoting GAN: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3940–3949, Brussels, Belgium. Association for Computational Linguistics.

Xu, P. and Fung, P. (2019). A novel repetition normalized adversarial reward for headline generation.

Xu, R., Tao, C., Jiang, D., Zhao, X., Zhao, D., and Yan, R. (2021). Learning an effective context-response matching model with self-supervised tasks for retrieval-based dialogues. In *AAAI*.

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

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

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017a). Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 2852–2858. AAAI Press.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017b). Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D., and Carin, L. (2017). Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 4006–4015. JMLR.org.

Zhang, Y. and Yang, Q. (2017). A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, PP.

Zhang, Z., Guo, T., and Chen, M. (2021). *DialogueBERT: A Self-Supervised Learning Based Dialogue Pre-Training Encoder*, page 3647–3651. Association for Computing Machinery, New York, NY, USA.

Zhu, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J., and Yu, Y. (2018). Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research &amp; Development in Information Retrieval*, SIGIR '18, page 1097–1100, New York, NY, USA. Association for Computing Machinery.