Thesis

Towards Generalization in Dialog through Inductive Biases

Shikib Mehri

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Maxine Eskenazi (Chair)	Carnegie Mellon University
Graham Neubig	Carnegie Mellon University
Yonatan Bisk	Carnegie Mellon University
Dilek Hakkani-Tur	Amazon
Tiancheng Zhao	Zhejiang University

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies.

Abstract

Generalization is imperative in dialog research. Data-driven models have been shown to be capable of performing specific tasks in constrained contexts, given ample data. However, the complexity of human communication necessitates that models of dialog be capable of generalizing beyond the limitations of any finite corpus. Models of dialog must be able to generalize to unseen and unforeseen phenomena. This thesis studies four classes of generalization: (1) generalization to new inputs, (2) generalization to new problems, (3) generalization to new outputs and (4) generalization to new dialog tasks. Inductive biases are studied in order to facilitate these four classes of generalization. An inductive bias is motivated by prior knowledge (e.g., domain knowledge, knowledge of the desired generalizations) and aims to influence the abstractions learned by a model in order to induce generalization. Four categories of inductive biases are studied: (1) through self-supervised learning, (2) inductive biases in the model architecture, (3) inductive biases in the problem formulation and (4) the task specification as an inductive bias.

This thesis consists of four chapters, each corresponding to one class of inductive bias. Chapter 3 studies self-supervised learning as an inductive bias and validates the use of the self-supervised training data and the self-supervised objectives as a mechanism for facilitating generalization to new inputs, new problems and new outputs. Chapter 4 incorporates inductive biases into the model architecture thereby *prescribing* a specific procedure by which the model infers the output from a given input. Through inductive biases in the model architecture, models are shown to generalize to new inputs/domains and to new outputs. Chapter 5 studies inductive biases in the problem formulation, wherein a problem is reformulated to better align with the capabilities of a pre-trained model. Both dialog evaluation and slot filling are reformulated to the task of response generation, which facilitates zero-shot generalization to new inputs and new outputs. Chapter 6 explores the most challenging class of generalization in dialog: generalization to new tasks. To transfer to unseen tasks (e.g., restaurant reservations) in a zero-shot setting, the task specification is used as an inductive bias. The task specification is a minimal expression of the task-specific properties that must be learned by a data-driven model. This thesis studies several mechanisms for using the task specification as an inductive bias: as an input, during the creation of synthetic data and as part of the model architecture. Leveraging the task specification as an inductive bias results in significant performance gains in zero-shot generalization to unseen tasks.

Contents

Chapter 1

Introduction

1.1 Overview

The need for generalization is a long-standing challenge in artificial intelligence research [Thrun and Mitchell, 1994; Mitchell et al., 1997; Torrey and Shavlik, 2010]. The inherently complex nature of many areas of artificial intelligence, for instance natural language processing [Lebowitz, 1983], necessitate that intelligent systems be able to *generalize* to unseen and unforeseen inputs. The need for generalization is further exacerbated by the dominant paradigm of data-driven machine learning. While data-driven models excel at learning patterns from a training corpus [Krizhevsky et al., 2012; Sutskever et al., 2014], they struggle to generalize beyond the observed data [Marcus, 2018].

Generalization is especially imperative in dialog research. The complexity of human communication dictates that no finite dialog corpus can contain every phenomenon necessary for modelling dialog. Data-driven models of dialog have been shown to be capable of performing specific tasks in specific contexts [Gao et al., 2018]. However, they struggle in low-resource settings and in transferring to unobserved phenomena. In order to mature beyond this narrow scope, models of dialog must be capable of generalizing beyond their training data. Concretely, these are the four types of generalization necessary in dialog modelling:

- **Generalization to New Inputs**: The space of inputs for all natural language tasks is unbounded, due to the compositional nature of language [Szabó, 2004]. Data-driven models must be able to generalize to unobserved linguistic patterns and terminology, which generally arise due to a domain shift or a stylistic difference. A dialog model which is trained on a variety of domains (e.g., restaurants, attractions), should have the capacity to generalize to new domains (e.g., hotels).
- **Generalization to New Problems**: Dialog modelling consists of a variety of problems, including intent prediction, slot filling, state tracking, response generation and evaluation. It is desirable to have a general purpose model of dialog that can be flexibly adapted to a variety of different problems. A dialog model capable of intent prediction, has learned skills pertaining to language understanding that should be transferable to other dialog problems, for instance slot filling.

- **Generalization to New Outputs:** Dialog corpora are generally collected with fixed ontologies (e.g., *slots:* restaurant name, date, time), however real-world usage often requires generalization beyond a specific ontology. Generalizing beyond a specific ontology is required when transferring to new tasks and domains (e.g., transfer from restaurants to hotels) or simply handling a constantly-evolving task (e.g., a banking system wants to ask users for their date of birth). As such, models of dialog should be able to generalize to new slots, new intents and new evaluation qualities (fluency, relevance, etc.).
- **Generalization to New Tasks:** There are many possible tasks that can be performed by a dialog system (e.g., providing transit directions, booking flights, resetting the life support on a spaceship, etc.). It is impossible to know all of the tasks *a priori* and dialog corpora are generally limited to a few different tasks. To handle a new task, or even to update the dialog policy for an existing task, data-driven models require additional dialog training data. Since collecting additional data is expensive, both in terms of time and cost, it is imperative that models be able to generalize to unseen tasks without additional data.

This thesis explores these four types of generalization in dialog modelling and aims to facilitate generalization through the use of *inductive biases* [Mitchell, 1980]. **An inductive bias is the set of assumptions used by an intelligent system to predict outputs for unobserved inputs, i.e., to generalize. Inductive biases, motivated by prior knowledge (e.g., domain knowledge, knowledge of the desired generalization, etc.), influence the abstractions learned by a data-driven model in order to induce generalization.** For example, the use of convolutional layers [LeCun et al., 1995] enforces spatial invariance and therefore allows generalization to shifted images (e.g., training data has all objects in bottom left, generalize to images with objects in top right). Multi-tasking is another inductive bias, motivated by knowledge of the target problem, which improves the performance and facilitates generalization by simultaneously learning multiple tasks [Caruana, 1997]. An inductive bias allows a system developer, with *knowledge* (i.e., of the domain, the problem or the desired generalization), to *modify* a component of a data-driven model in order to *prescribe* specific behavior and thereby *induce the desired generalization*.

A data-driven model, particularly a neural network, is often seen as a black-box algorithm that implicitly learns abstractions from a given corpus. In contrast, incorporating inductive biases into the design of a data-driven model allows the explicit enforcing of certain behaviors. This thesis presents several mechanisms for incorporating various forms of inductive biases into data-driven models of dialog. These inductive biases enforce specific learned abstractions and thereby facilitate the four aforementioned classes of generalization.

In introducing the concept of *learning machines*, Turing and Haugeland [1950] describe three components that influence the abilities of a human mind: (1) the initial state of the mind (i.e., at birth), (2) the education to which it is subjected and (3) other experiences, not described as education, to which it has been subjected. This characterization of a human mind can be adapted to describe a data-driven model. A data-driven model is the result of three components: (1) the design of the model architecture, which influences the mechanisms by which it processes inputs and generates predictions, (2) the supervised learning it undergoes, which encompasses the data, the training algorithm and the problem formulation and (3) the

self-supervised learning which it undergoes, which includes the data and the self-supervised objectives. Together, these three components produce a data-driven model. As such, to influence the resulting model in order to facilitate generalization, this thesis incorporates inductive biases into each of these three components.

Self-supervised learning is a long-standing concept in artificial intelligence research [Dayan et al., 1999]. Large-scale self-supervised training [Devlin et al., 2018; Radford et al., 2018] has brought about a paradigm shift in natural language processing research. Concretely, self-supervised training objectives learn to interpret and represent the input through reconstruction tasks such as predicting the next word [Radford et al., 2018] or predicting a masked word [Devlin et al., 2018]. Data for self-supervised training is widely available (i.e., on the internet), therefore allowing self-supervised learning to be performed at unprecedented scale [Radford et al., 2019; Brown et al., 2020]. Self-supervised learning has been shown to facilitate generalization [Wang et al., 2018] with BERT-based models. Devlin et al. [2018] achieve state-of-the-art results on a variety of different problems, without necessitating hand-crafted architectures.

Self-supervised learning is itself a form of inductive bias. The choice of data and the self-supervised objectives influence *what is learned* by a data-driven model, and can therefore be used to facilitate generalization. This thesis explores the use of self-supervised learning as an inductive bias for inducing generalization in models of dialog. Concretely, two studies are carried out:

- **Section 3.2** presents CONVBERT, a model that was pre-trained on open-domain dialog data, and carries out a comprehensive study of task-adaptive self-supervised pre-training and multi-tasking on a number of different language understanding problems in task-oriented dialog. This work demonstrates the efficacy of self-supervised training as a mechanism for facilitating generalization to new domains. By using self-supervised training as an adaptation mechanism, a dialog model can generalize to new problems (intent prediction, slot filling and state tracking) and to new domains in few-shot settings.
- **Section 3.3** designs self-supervised objectives that approximate different qualities for open-domain dialog evaluation. The USR metric relies on three self-supervised objectives that act as a proxy for five different qualities of dialog. Without any annotated data or supervised learning, the resulting models effectively approximate different qualities and attain improved correlations with human judgments, relative to existing automatic evaluation metrics. This work demonstrates that the choice of objectives acts as an inductive bias that facilitates the learning of dialog evaluation models without any labeled training data.

The design of a model architecture is another important component that defines the mechanisms by which a data-driven model processes inputs and generates outputs. The architecture plays an important role in dictating the abstractions learned by the resulting model. Hand-crafted architectures for dialog models achieved strong performance on a variety of problems [Bhargava et al., 2013; Hakkani-Tür et al., 2016; Serban et al., 2017]. Researchers incorporate knowledge about the domain and the specific problem into the design of a hand-crafted architecture, in order to influence the abstractions learned by the model. For

example, the Hierarchical Encoder Decoder (HRED) [Serban et al., 2017] is designed to first encode every utterance in a dialog and then produce a dialog-level representation by considering a sequence of utterance-level representations. This hierarchical structure is an inductive bias, motivated by the inherent hierarchical structure in dialog, that forces a data-driven model to independently consider each utterance prior to producing a dialog-level representation. These inductive biases allowed the model to learn higher-level abstractions from a given corpus, in order to achieve better performance on a held-out test set.

This thesis incorporates inductive biases into the model architecture with the goal of inducing generalization. Through modifications to the architecture, and by extension the training algorithm, the resulting data-driven models are forced to learn certain high-level abstractions that facilitate different types of generalization. The design of these inductive biases is motivated by knowledge of the problem and the desired generalizations. Two studies are carried out pertaining to incorporating inductive biases into neural models of dialog:

- **Section 4.2** presents Structured Fusion Networks (SFN), which incorporate the traditional dialog pipeline into neural response generation models. By designing an architecture and a multi-tasking setup that mimics pipeline-based dialog systems, the resulting model possesses the generalizability and robustness of pipeline-based dialog systems while still maintaining the benefits of neural models. Structured Fusion Networks prescribe a specific procedure by which the model must perform the task of response generation. In doing so, the resulting model performs better in low-resource settings and can generalize to unseen domains in few-shot settings.
- **Section 4.3** introduces a model for the problem of intent prediction, which is capable of generalization to *new domains* and to *new outputs*. This work leverages example-driven training, in combination with self-supervised training (building on the work in Chapter 3), to facilitate generalization. In example-driven training, the output class is predicted by measuring the similarity of a given input with a set of examples. This modification to the architecture and training algorithm acts as an inductive bias that (1) reduces the task to sentence similarity and relieves the model of having to learn an implicit representation-to-intent mapping, (2) maintains consistency with the capabilities of the pre-trained model, i.e. generating a similar encoding for similar inputs. By making the representation-to-intent mapping an explicit non-parametric process (rather than an implicitly learned set of weights in a classification layer), the resulting model is able to generalize to new outputs and can predict unseen intents, given the corresponding examples. Furthermore, by maintaining consistency with the capabilities of the pre-trained model, this approach avoids catastrophic forgetting and is capable of generalizing to new domains and new outputs without any additional training.

The problem formulation strongly influences what is learned by the resulting data-driven model. This is especially true when working with large-scale pre-trained models. Pre-trained models have certain capabilities that can be effectively leveraged with the right problem formulation. In this sense, the problem formulation is an inductive bias motivated by knowledge of the pre-existing capabilities of the pre-trained

model. In this thesis, we leverage generative pre-trained models that are trained on open-domain dialog data. These models perform well on the task of open-domain response generation, however they have also implicitly learned specific skills that can be leveraged for different dialog problems:

- **Section 5.2** introduces fine-grained evaluation of dialog (FED), which reformulates the problem of dialog evaluation to be better aligned with the capabilities of large-scale pre-trained generative language models. Concretely, FED is an unsupervised evaluation metric which evaluates the language model likelihood of certain follow-up utterances as a mechanism as an estimator of different evaluation qualities (e.g., fluency, relevance, consistency, engagingness, etc.). This formulation of dialog evaluation acts as an inductive bias that leverages the capabilities of the pre-trained model and leads to zero-shot generalization to (1) new outputs (i.e., 18 different dialog qualities) and (2) new inputs (i.e., different linguistic patterns).
- **Section 5.3** similarly leverages the capabilities of pre-trained language models for the task of slot filling. The Generative Slot Filling model (GENSF) formulates slot filling as a response generation task. This formulation is motivated by the knowledge that pre-trained language models have implicitly learned some notion of certain slots (e.g., time, name, etc.). This formulation facilitates better performance in low-resource settings and zero-shot generalization to new slots.

These three types of inductive biases (i.e., (1) through self-supervised learning, (2) in the architecture, and (3) in the problem formulation) allow us to enforce *what is learned* by data-driven models, thereby facilitating three of the four aforementioned classes of generalization (i.e., to new inputs, to new outputs and to new problems). In Chapters 3 - 5, this thesis demonstrates how well-motivated inductive biases can yield improved performance and sample-efficient data-driven models of dialog. At a high level, this work demonstrates how *knowledge* can *motivate* various forms of inductive biases to *prescribe* specific behavior in order to facilitate generalization. Chapter 6 leverages and extends the aforementioned forms of inductive bias to facilitate generalization to *new tasks*.

Generalizing to a new task is significantly more challenging, as it requires data-driven models to generalize to both new inputs (caused by domain shift), to new outputs (new tasks require new output classes), and to an unseen task-specific *dialog policy*. The task-specific dialog policy defines a particular task. The policy describes the desired behavior of a system, i.e. how it should respond to specific user intents. For example, the policy may indicate that after the user provides their name – the system should ask for their date of birth. When generalizing to an unseen task in a zero-shot setting, a data-driven model has no notion of the dialog policy for the new task. As such, to facilitate zero-shot transfer to an unseen task, the unseen dialog policy must be provided to a data-driven model through an inductive bias.

This thesis addresses this by introducing the *schema-guided paradigm*, wherein a *task specification* (i.e., a schema) is leveraged as an inductive bias to induce zero-shot generalization to an unseen task. The schema defines a task through a generalized representation of the task-specific dialog policy and dialog ontology (i.e., the set of intent and slot classes). Since a data-driven model does not and can not have prior

knowledge of an unseen task (i.e., the policy and ontology), the schema is fundamentally *necessary* for zero-shot generalization. Through three different studies, this thesis aims to demonstrate that the schema is *sufficient* for facilitating zero-shot generalization. These studies use the task specification as an inductive bias through three different approaches:

- **Section 6.3** provides the task specification as an *input* to the Schema Attention Model (SAM). Concretely, SAM attends to a graph-based representation of the task-specific dialog policy. SAM builds on the previous chapters of this thesis and relies on (1) self-supervised pre-training, (2) inductive biases in the model architecture, (3) a problem formulation that facilitates generalization to new outputs and (4) a graph-based representation of the task-specific dialog policy which acts as an inductive bias during zero-shot generalization. When provided with the schema for an unseen task, SAM attains significant gains in zero-shot generalization.

- **Section 6.4** leverages the task specification, in tandem with a large-scale pre-trained language model, to generate synthetic data. Language Models as Data (LAD) uses a domain-agnostic process to create accurate and diverse synthetic data, that can effectively convey the necessary properties of a task to a downstream data-driven model of dialog. By expressing the task specification through the synthetic training data, LAD facilitates significant gains in zero-shot generalization in intent prediction, slot filling and next-action prediction.

- **Section 6.5** uses the task specification as an inductive bias in the model architecture. Schema-Guided Fusion (SGF) represents the task-specific dialog policy in the architecture of a neural dialog model. SGF uses the task specification to explicitly define the relationship between different compontents of a dialog model, specifically intent prediction, slot filling and next action prediction. By explicitly defining the relationship between different components, SGF enforces adherence to the task specification and leverages the strengths of zero-shot intent prediction and slot filling to facilitate performance gains in zero-shot next action prediction. Furthermore, SGF is able to *learn* a task-specific dialog policy from human dialogs.

This thesis presents four categories of inductive biases (i.e., (1) through self-supervised training, (2) in the model architecture, (3) in the problem formulation and (4) through the task-specification) to address the four classes of generalization necessary in dialog (i.e., to new inputs, to new outputs, to new problems and to new tasks). These inductive biases are motivated by knowledge of the problem, the domain, or the desired generalization. Data-driven models are effective at learning to address specific problems in specific contexts. This thesis takes an important step towards progressing models of dialog beyond this narrow scope and to facilitate generalization to unseen phenomena through inductive biases.

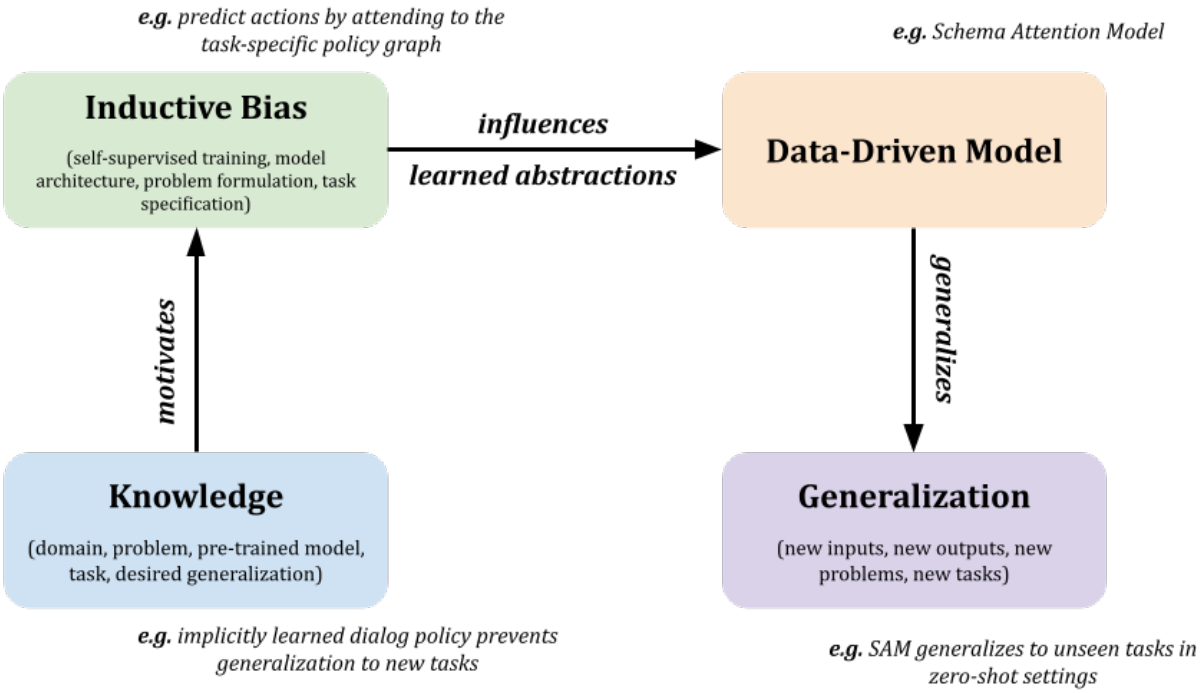


Figure 1.1: A visualization of the thesis statement. Knowledge motivates inductive biases which influence learned abstractions in data-driven models and thereby facilitate generalization. An example of how a particular section (Section 6.3) of this thesis aligns to this image is shown.

1.2 Thesis Statement

Thesis Statement: This thesis advocates for the use of inductive biases to facilitate generalization in dialog. Inductive biases can be leveraged to influence the high-level abstractions that are learned by a data-driven model. By designing inductive biases, motivated by knowledge of the domain, problem or the desired generalization, the work in this thesis achieves superior generalization. Four different types of inductive biases are explored: (1) through self-supervised training, (2) inductive biases in the model architecture, (3) inductive biases in the problem formulation, (4) the task specification as an inductive bias. These four types of inductive biases facilitate four classes of generalization in dialog: (A) generalization to new inputs, (B) generalization to new problems, (C) generalization to new outputs and (D) generalization to new tasks. This thesis addresses the problem of generalization and carries out studies to justify the claim that *if inductive biases are leveraged to incorporate domain knowledge into a data-driven model, then models of dialog can progress beyond the limitations of the training corpora and successfully generalize to unseen and unforeseen phenomena.*

Figure 1.1 visualizes the thesis statement, with an example of how this framework is applied to facilitate generalization to unseen tasks in Section 6.3. Table 1.1 characterizes the different inductive biases used in this thesis, their corresponding motivation and resulting generalizations.

Section	Inductive Biases	Motivation	Generalizes to	Problems
3.2	self-supervised data; training algorithm	adapt to relevant domains	new inputs; new problems	intent prediction; state tracking; semantic parsing; slot filling
3.3	self-supervised objectives; self-supervised data	approximate evaluation qualities	new inputs; new outputs (qualities)	dialog evaluation
4.2	model architecture; training algorithm	structured modelling resembling pipeline-based systems	new inputs	response generation
4.3	model architecture; training algorithm	facilitate generalization to new outputs; leverage capabilities of pre-trained models	new inputs; new outputs	intent prediction
5.2	problem formulation	leverage capabilities of pre-trained models	new inputs; new outputs	dialog evaluation
5.3	problem formulation; model architecture	leverage capabilities of pre-trained models	new inputs; new outputs	slot filling
6.3	task specification; model architecture; problem formulation; self-supervised data	disentangle the task-specific dialog policy to facilitate generalization to new tasks	new inputs; new outputs; new tasks	next action prediction
6.4	task specification; model architecture; problem formulation; self-supervised data	generate synthetic data that conveys the task specification	new problems; new inputs; new outputs; new tasks	next action prediction; slot filling; intent prediction
6.5	task specification; model architecture; problem formulation; self-supervised data	leverage the task specification as an inductive bias in the model architecture	new problems; new inputs; new outputs; new tasks	next action prediction

Table 1.1: Characterization of the different studies carried out in this thesis, the inductive biases used, their corresponding motivations and the resulting generalizations.

Chapter 2

Background and Related Work

2.1 Neural Models of Dialog

2.1.1 Natural Language Understanding

Language understanding is a critical component of dialog systems. The objective of natural language understanding is to extract semantic concepts from natural language utterances, through intent prediction and slot filling [Tur and De Mori, 2011]. Through the ability to learn meaningful patterns from large volumes of data, neural models provide a scalable framework for language understanding [Mesnil et al., 2013].

Intent prediction is the problem of classifying a natural language utterance into one of several pre-defined intent classes [Hemphill et al., 1990; Coucke et al., 2018]. For example, given the utterance *'I'm leaving from CMU'*, the appropriate intent class may be *'inform-departure-location'*. Intent prediction is a vital component of dialog systems, as determining the goals of the user is the first step to producing an appropriate response [Raux et al., 2005; Young et al., 2013].

Slot filling is the problem of identifying values for pre-defined attributes, or slots, in a natural language utterance [Tur and De Mori, 2011]. For example, given the utterance *'I'm leaving from CMU at 10 AM'*, the resulting slots should be *'departure-location': CMU* and *departure-time: 10 AM*. Slot filling is a crucial component of task-oriented dialog systems, as it allows a dialog system to ground its response generation in the information provided by the user [Young, 2002, 2010]. The majority of early work on language understanding leverages the Airline Travel Information System (ATIS) corpus [Hemphill et al., 1990], which consists of slot-labeled utterances pertaining to flight-related information (e.g., *'I want to fly to Boston from New York next week'*).

The earliest use of neural models for language understanding was in the context of call-routing, wherein the domain and the intent of a natural language utterance had to be determined in order to appropriately route the call. Sarikaya et al. [2011] leveraged Deep Belief Nets (DBNs) to learn a multi-layer generative model from unlabeled data. The features discovered by the DBN were then used to initialize a feed-forward neural network, which was fine-tuned on labeled data. The resulting neural model was shown to outperform

traditional classifiers (e.g., Support Vector Machines, Maximum Entropy, etc.) on the problem of intent prediction, especially when trained on larger quantities of data.

Deoras and Sarikaya [2013] leverage a similar approach for the task of slot filling. By initializing a feed-forward neural network with features learned by a DBN, the resulting neural model outperforms conditional random fields (CRF), which were at the time the state-of-the-art for sequence classification. Mesnil et al. [2013] train several classes of recurrent neural networks (RNNs) on the ATIS corpus and outperform CRF classifiers. Mesnil et al. [2014] carries out additional experiments demonstrating that RNN classifiers outperform feed-forward neural networks. Yao et al. [2013] improve the use of RNNs for slot filling by incorporating lexical, named entity and bag-of-words features into the model. By incorporating additional features, Yao et al. [2013] were using inductive biases to influence the abstractions learned by the model in order to better model the ATIS corpus. Yao et al. [2014] use long short-term memory (LSTM) networks [Hochreiter and Schmidhuber, 1997; Hochreiter, 1998] and demonstrate that the improved sequence modelling abilities of LSTMs results in stronger performance on the problem of slot filling.

Shi et al. [2015] jointly train a model for domain identification, intent prediction and slot filling. Their model first encodes an utterance using an RNN and classifies the utterances using a convolutional neural network (CNN). Similarly [Xu and Sarikaya, 2013] jointly trains a CNN model on the tasks of intent detection and slot filling. The model of Xu and Sarikaya [2013] uses CNNs to construct a neural version of a triangular CRF, which jointly models the intent and slot sequence and exploits their dependencies. Guo et al. [2014] jointly trains on domain classification, intent prediction and slot filling using a recursive neural network. The recursive neural network encodes an utterance recursively, conditioned on the parse tree of the utterance. By jointly training on multiple natural language understanding problems, these approaches are leveraging domain knowledge about the inherent similarity of these tasks as an inductive bias that influences the abstractions learned by the model. Furthermore, domain knowledge about the problem motivates the design of the model architectures, as exemplified by the CNN triangular CRF of Xu and Sarikaya [2013] and the recursive neural network of Guo et al. [2014].

Motivated by the success of early neural approaches for language understanding, the dialog research community began developing custom architectures and training paradigms to improve neural models. Kurata et al. [2016] propose the encoder-labeler LSTM, which first encodes an input utterance into a fixed length vector that is then used as the initial state of another LSTM for sequence labeling. In this manner, the second LSTM can fill slots with knowledge of the entire sequence. Liu and Lane [2016] proposes an RNN encoder-decoder network with attention for the tasks of slot filling and intent prediction. The use of the attention mechanism allows the encoder-decoder network to better identify supporting information when when predicting a slot tag for a particular word. To mitigate the difficulty of memorization in RNNs, Peng and Yao [2015] propose RNNs with external memory which can dynamically read from/write to an external memory, in order to better leverage patterns observed in previous utterances. Goo et al. [2018] augment an attention RNN with a slot-gating mechanism to better model the relationship between the intent classes and slot tags. Zhao and Feng [2018] present a generative model for slot filling which combines a sequence-to-sequence network with a pointer network. Zhang et al. [2020] represent the utterance as a graph and

leverage a graph LSTM architecture to encode the utterance and predict the appropriate slots and intents.

Neural models for language understanding were progressively improved to better model the problems of intent prediction and slot filling. These improvements to the architecture design were motivated by domain knowledge (e.g., the relationship between intents and slots; the importance of external knowledge in slot filling). Changes to the model architectures influenced the abstractions learned by the model, and therefore allowed better generalization to unseen samples in the test set. Nonetheless, these neural models are largely incapable of generalization to out-of-domain samples [Chen et al., 2019a] and are constrained to a pre-defined ontology of slots and intents [Wu et al., 2019].

2.1.2 Response Generation

The problem of response generation requires a dialog system to generate a natural language response to a given dialog context. Generating a relevant, fluent and appropriate response is a challenging task that necessitates (1) understanding a natural language conversation, (2) producing a natural language utterance, and (3) progressing the dialog according to some pre-defined dialog policy. The problem of response generation is central to dialog; 'solving' response generation would be equivalent to solving dialog [Turing and Haugeland, 1950].

It is important to make the distinction between response generation and natural language generation. The objective of natural language generation is to generate an appropriate response conditioned on a desired system action or a set of slot values [Rambow et al., 2001]. In contrast, response generation is the end-to-end task of mapping a dialog history to a natural language response.

Vinyals and Le [2015] train a sequence-to-sequence (seq2seq) network [Sutskever et al., 2014] for both open-domain and goal-oriented dialog. A response is generated using an LSTM to first encode the conversation history and subsequently generate the natural language response. The seq2seq network is trained on the OpenSubtitles dataset [Tiedemann, 2009] and internal dataset of IT conversations.

Since the work of Vinyals and Le [2015], there has been a large body of work addressing the problem of response generation. Li et al. [2015] use a Maximum Mutual Information (MMI) training objective to encourage informative responses. Serban et al. [2016] propose the hierarchical encoder decoder (HRED) to better model the hierarchical nature of dialog. HRED uses utterance encoder to produce utterance-level encodings and subsequently reasons over the utterance encodings using a conversational context encoder. Li et al. [2016b] incorporate a number of heuristics into a reinforcement learning reward function, to encourage useful conversational properties such as informativity, coherence and forward-looking. Li et al. [2016a] encodes a speaker's persona as a distributed embedding and uses it to improve dialog generation. Zhao et al. [2017a] enable task-oriented systems to handle out-of-domain conversation, through the use of augmented training data and an entity indexer. Li et al. [2017] train a discriminator to predict whether an utterance was generated by the dialog system or a human, and leverage this discriminator to train a response generation model. Eric and Manning [2017] present introduce key-value retrieval networks, which generates a response by reasoning over underlying knowledge bases through a key-value retrieval mechanism.

The MultiWOZ corpus [Budzianowski et al., 2018] is a useful resource for research on response generation for task-oriented dialog. MultiWOZ consists of dialogs between a tourist and a clerk at an information centre. MultiWOZ consists of multi-domain dialogs spanning seven different domains (restaurants, hotels, attractions, trains, hospital, taxi, police). In addition to dialogs, the corpus consists of belief span, user intent and dialog act annotations. These additional annotations allow research into structured modelling of task-oriented response generation. Furthermore, the presence of multiple domains facilitates an assessment of the generalizability of dialog models.

The baseline response generation model presented by Budzianowski et al. [2018] augments a seq2seq network, by incorporating the ground-truth belief span annotation using a feed-forward layer between the encoder and the decoder. Chen et al. [2019b] augment a Transformer network [Vaswani et al., 2017] with a gating mechanism that conditions on a dialog act graph. In this manner, different self-attention heads are used depending on the predicted dialog act graph for a given dialog context. Zhao et al. [2019] introduces a latent action framework that treats the action spaces of seq2seq models as latent variables. Through this latent action framework, it is possible to optimize an end-to-end response generation model with reinforcement learning while mitigating the degradation in the quality of the natural language responses.

Neural networks for response generation aim to learn a semantically meaningful mapping between a dialog context and a system response. To better accomplish this, research has augmented architectures and training paradigms with inductive biases that influence certain behaviors in the resulting models. In modelling open-domain dialog, the objective has generally been to produce more engaging and relevant responses. In contrast, research in task-oriented response generation has addressed the challenges of producing system responses that progress the dialog according to some task-specific policy (i.e., help the user achieve certain goals) and to effectively leverage knowledge.

End-to-end neural networks for response generation have been shown to suffer from a number of shortcomings. Li et al. [2016b] introduced the dull response problem, which describes how neural dialog systems tend to produce generic and dull responses (e.g., *"I don't know"*). Zhao and Eskenazi [2018] describes generative dialog models as being data-hungry, and difficult to train in low-resource environments. Mo et al. [2018]; Zhao and Eskenazi [2018] both demonstrate that dialog systems have difficulty generalizing to new domains. Madan et al. [2018]; Zhao et al. [2018] highlight the inherent lack of interpretability in standard generative dialog models, both proposing unsupervised mechanisms of making models more interpretable. Hu et al. [2017] work on the problem of controllable text generation, which is inherently difficult in seq2seq architectures, including generative models of dialog.

2.2 Pre-Trained Models

The advent of large-scaled pre-trained models has brought about a paradigm shift in natural language processing. Rather than designing sophisticated architectures consisting of well-motivated inductive biases, strong performance can be obtained on a variety of problems by fine-tuning pre-trained models. This section describes large-scale pre-trained models, their applications to dialog, their achievements in facilitating

generalization and the shortcomings of the fine-tuning paradigm.

2.2.1 Large-Scale Pre-Trained Models

Neural networks perform best when trained on large quantities of data. However, natural language processing suffers from a data scarcity problem. It is infeasible to construct a annotated corpus that contains every phenomenon necessary for modelling language. This problem is mitigated to some degree through the careful design of model architectures and training paradigms, which learn higher level abstractions that can generalize to unseen samples in the test set. However, this is still insufficient for generalizing to new domains or in low-resource settings. Large-scale pre-training aims to tackle the problem of generalization. Concretely, models are pre-trained in a self-supervised manner on large quantities of unlabeled data. These pre-trained models are then fine-tuned on an annotated downstream corpus. In this manner, the model learns to produce meaningful representations during pre-training which are then leveraged during fine-tuning.

The first instance of large-scale pre-training in NLP was word2vec [Mikolov et al., 2013a] which trains log-linear classifiers in a self-supervised manner on the Google News corpus (6 billion tokens). Two versions of word2vec were trained, CBOW word2vec (continuous bag-of-words) which predicts the current word based on the surrounding context and skip-gram word2vec which predicts the surrounding words conditioned on the current words. The word2vec model learns representations of words such that semantically similar words have similar vector representations. The word2vec model was then used to initialize embeddings in models for downstream tasks.

Skip-thought vectors [Kiros et al., 2015] were pre-trained on the BookCorpus dataset [Zhu et al., 2015], a collection of text from books totalling over 984 billion words. Kiros et al. [2015] train skip-thought vectors to auto-regressively generate the previous sentence and the next sentence conditioned on the current sentence. Through this self-supervised training, skip-thought vectors learn to produce better contextual representations of text. The off-the-shelf use of skip-thought vectors resulted in better performance on several downstream tasks.

Peters et al. [2018] trained a deep bidirectional LSTM language model on a large text corpus (5.5 billion words) to predict both the next and previous word. The intermediate layers of the deep LSTM were then used to produce ELMo (Embeddings from Language Models). Rather than using the top-level representation produced by the biLSTM, ELMo learns a linear combination of the different layers for each downstream task. ELMo achieved state-of-the-art results on several downstream language understanding tasks.

Radford et al. [2018] introduce GPT, a unidirectional language model using a transformer architecture [Vaswani et al., 2017] that was pre-trained on the BookCorpus [Zhu et al., 2015]. GPT achieved strong performance on several downstream tasks after pre-training with a simple language modelling objective.

Devlin et al. [2018] introduce BERT, a large transformer model that was trained with two self-supervised objectives: masked language modelling and next sentence prediction. In masked language modelling, similar to the the CBOW word2vec model [Mikolov et al., 2013b] the objective is to predict a *masked* token conditioned on the surrounding context. Through the masked language modelling objective, BERT learns to

produce contextually meaningful representations of words. Next sentence prediction tasks BERT with predicting whether a sentence is the appropriate next sentence. The next sentence prediction objective forces BERT to produce meaningful sentence-level representations. Through pre-training on English Wikipedia and the BookCorpus (totalling over 3 billion words), BERT achieves state-of-the-art results on a number of natural language understanding tasks and the GLUE benchmark [Wang et al., 2018].

The impressive performance of BERT inspired several follow-up studies in large-scale pre-training. Liu et al. [2019b] present RoBERTa which trains a BERT-based model on additional data, for a longer duration and with better hyperparameter tuning. RoBERTa strongly outperforms BERT, demonstrating the importance of scale in pre-training. Yang et al. [2019] introduce XLNET which uses a modified self-supervised objective which learns conditional distributions for all permutations of tokens in a sequence. BERT is time-consuming and expensive to pre-train and difficult to deploy in production settings due to its large parameter size. To address this problem, there have been efforts to make BERT more efficient through parameter reduction techniques such as knowledge distillation [Sanh et al., 2019] and cross-layer parameter sharing [Lan et al., 2019].

GPT-2 [Radford et al., 2019] improves upon the GPT, by training a larger language model (1.5B parameters) on over 8 million webpages from WebText (40GB of text). GPT-2 demonstrates strong gains on a variety of language understanding and generation tasks. Zhang et al. [2019c] introduces DialoGPT, which further pre-trains the GPT-2 model on 147 million open-domain dialogs extracted from Reddit. Through human evaluation, DialoGPT is shown to generate high quality responses for open-domain dialog, on par with human-generated responses.

GPT-3 [Brown et al., 2020] is a 175B parameter language model trained on the CommonCrawl data, which consists of over 400B tokens. Despite training with a simple language modelling objective, similar to GPT and GPT-2, GPT-3 achieves impressive results on a variety of language understanding and language generation tasks *without any additional fine-tuning*. Through large-scale pre-training on a massive collection of text from the internet, GPT-3 was able to learn high-level abstractions that facilitate few-shot generalization through prefix tuning.

2.2.2 Fine-Tuning on Dialog

Large-scale pre-training has attained significant performance gains across many tasks within NLP [Devlin et al., 2018; Radford et al., 2018]. Through self-supervised pre-training on large natural language corpora, these models gain generalized language understanding capabilities that transfer effectively to downstream tasks [Wang et al., 2018]. Generally, prior to fine-tuning, the pre-trained models are adapted to the specifics of the downstream task through minor architectural modifications (e.g., adding a classification layer) [Chen et al., 2019a]. By avoiding major task-specific changes to the models, it is assumed that the underlying pre-trained models possess a degree of generality that allows transfer to a variety of tasks.

Chen et al. [2019a] fine-tune BERT for the tasks of intent prediction and slot filling, obtaining state-of-the-art results by simply adding a linear classification layer on top of the BERT architecture. Castellucci

et al. [2019] concurrently carry out similar studies and further assess the performance of BERT on non-English language understanding corpora. Heck et al. [2020] fine-tune BERT for dialog state tracking, using a triple copy mechanism that can copy values from (1) the user input, (2) the system utterances, and (3) different slots.

Pre-trained models have also been leveraged for task-oriented response generation. Large-scale pre-trained models excel at sampling from a distribution of potential responses, to generate a relevant and appropriate response to a dialog context. This makes them especially effective for open-domain dialog, which is not necessarily goal-driven. In contrast, this is detrimental to task-oriented dialog wherein system responses must progress the dialog according to a task-specific policy. To mitigate this problem, the belief states, database outputs, and predicted dialog acts are incorporated as part of the input sequence when fine-tuning pre-trained models, such as GPT-2, on task-oriented dialog corpora [Budzianowski and Vulić, 2019; Peng et al., 2020b; Yang et al., 2020].

To better adapt pre-trained language models for task-oriented dialog, there has been considerable effort in further pre-training on task-oriented dialog data. Wu et al. [2020] further pre-trains BERT with self-supervised objectives on multiple task-oriented dialog datasets, prior to supervised fine-tuning. Peng et al. [2020a] further trains GPT-2 on a number of task-oriented dialog data for the task of response generation, conditioned on structured intermediate annotations (belief span, database output) incorporated into the sequence. Rather than further pre-training an existing model, ConveRT [Henderson et al., 2019] pre-trains only on open-domain dialog with a light-weight model and demonstrates superior transfer to task-oriented dialog relative to BERT on both intent prediction [Casanueva et al., 2020] and slot filling [Coope et al., 2020]. By further pre-training on dialog data, these approaches achieve better alignment between the capabilities of the pre-trained model and the requirements of the downstream problems. Henderson and Vulić [2020] take this a step further by designing a custom pre-training paradigm specifically for the task of slot filling. Concretely, the ConVEx model uses open-domain dialogs from Reddit with a pairwise cloze pre-training objective. Through this pre-training objective, the ConVEx model learns representations that result in better slot filling performance on several downstream corpora.

2.3 Dialog Evaluation

Evaluation metrics often define the research direction of a field. As dialog systems begin to demonstrate human-level performance [Zhang et al., 2019c; Adiwardana et al., 2020], the development and adoption of meaningful and interpretable automatic evaluation measures is essential. Since standard language evaluation metrics (e.g., BLEU, METEOR) have been shown to be ineffective for dialog [Deriu et al., 2019; Liu et al., 2016], human evaluation is often used for dialog evaluation. However, human evaluation is costly and is typically only used as a final evaluation. During development, systems are generally optimized for poorly correlated automatic metrics which can result in sub-par performance [Dinan et al., 2019]. Automatic metrics must be meaningful and interpretable so that they can be used to compare dialog systems, understanding their respective strengths and weaknesses, and effectively guide dialog research. This section provides an

overview of research on automatic evaluation of dialog.

2.3.1 Standard Evaluation Metrics

Until recently, dialog systems research used standard language evaluation metrics. These metrics were initially developed for machine translation and compute word overlap between the generated text and a ground-truth reference.

BLEU [Papineni et al., 2002] is a word overlap metric that is often used to benchmark natural language generation. BLEU computes the n-gram precision between the generated response and a human reference. METEOR [Banerjee and Lavie, 2005] and ROUGE Lin [2004] were proposed to improve upon BLEU. METEOR incorporates stems and synonyms when calculating word overlap. Instead of computing precision, ROUGE opts to measure n-gram recall. Though these two metrics improve upon BLEU, they remain ineffective for dialog evaluation [Liu et al., 2016].

Dialog evaluation is difficult for several reasons: (1) The one-to-many nature of dialog [Zhao et al., 2017b] makes word-overlap metrics ineffective for scoring valid responses that deviate from the ground-truth [Liu et al., 2016; Deriu et al., 2019]. To avoid comparing to a single reference response, several papers have proposed the use of multiple reference responses. Multiple reference responses can be obtained with retrieval models [Galley et al., 2015; Sordoni et al., 2015] or through data collection [Gupta et al., 2019]. These multi-reference metrics show improvement in performance, but it is infeasible to thoroughly cover the space of potential responses for a given dialog context. (2) Dialog quality is inherently multi-faceted [Walker et al., 1997; See et al., 2019] and an interpretable metric should measure several qualities (e.g., *interesting*, *relevant*, *fluent*). The relative importance of different qualities may differ for different domains and applications. As such, it is unreasonable for a dialog evaluation metric to *prescribe* the criteria that makes a response or a dialog good. (3) Dialog systems have begun to be evaluated in an interactive setting [Ram et al., 2018; Adiwardana et al., 2020] where a real user has a back-and-forth conversation with a system. Interactive evaluation is not constrained to a static corpus and better captures the performance of a system in a realistic setting. However, standard language evaluation metrics compare to a ground-truth response, making them unsuitable for assessing interactive conversations. Furthermore, it is imperative that dialog evaluation metrics measure *dialog-level* qualities such as consistency, cohesiveness and topic depth.

2.3.2 Embedding-Based Metrics

To address some of the shortcomings of standard evaluation metrics, a new class of metrics was proposed. Embedding-based metrics compute the similarity between the generated text and the ground-truth by computing similarity in embedding space.

Greedy Matching [Rus and Lintean, 2012] is an embedding-based metric that greedily matches each word in the generated sequence to a reference word based on the cosine similarity of their embeddings. The final score is then an average over all the words in the generated sequence. Embedding Average [Wieting et al., 2015] computes a sentence embedding for both the generated sequence and the ground-truth response

by taking an average of word embeddings. The score is then a cosine similarity of the average embedding for both the generated and reference sequence. Vector Extrema [Forgues et al., 2014] follows a similar setup to Embedding Average, where the score is the cosine similarity between sentence embeddings. Rather than taking an average over word embeddings, this method identifies the *maximum* value for each dimension of the word embedding. Taking the maximum is motivated by the idea that common words will be de-emphasized as they will be closer to the origin. Vector Extrema has been shown to perform better on dialog tasks than other metrics [Gupta et al., 2019; Liu et al., 2016]. Skip-Thought [Kiros et al., 2015] uses a recurrent neural network to produce a sentence-level embedding for the generated and reference sequences. A cosine similarity is then computed between the two embeddings.

BERTScore [Zhang et al., 2019b] uses a pre-trained BERT [Devlin et al., 2018] model to greedily match each word in a reference response with one word in the generated sequence. By doing so, it computes the recall of the generated sequence. BERTScore was shown to have strong system-level and segment-level correlation with human judgment on several machine translation and captioning tasks. However, although it is a more sophisticated metric, it still compares word similarity between a reference and a generated sequence. While this method may work well for tasks where there is a limited space of outputs for each input (e.g., captioning, translation), it is ineffective at dealing with the one-to-many nature of dialog.

2.3.3 Model-Based Evaluation

To better address the one-to-many problem in dialog, several approaches were proposed for model-based evaluation. Model-based evaluation relies on a model to predict the quality of a particular response in the context of the dialog history.

Lowe et al. [2017] train ADEM to produce a quality score conditioned on the dialog context, the reference response and the generated response. Venkatesh et al. [2018] present a framework for evaluation of Alexa prize conversations, which attains moderate correlation with user ratings. Both of these methods are trained on explicit quality annotations. This allows these metrics to progress beyond the limitations of the reference response, however they are still constrained by the limitations of their training corpora. Evaluation metrics must be able to generalize to new domains, topics and systems. Model-based metrics that train on specific corpora may be incapable of this type of generalization, and therefore insufficient for evaluating dialog beyond the domains and systems present in their training data.

Li et al. [2017] proposes a reference-free dialog evaluator which is trained to discriminate between human and generated responses. This approach evaluates the quality of a response without a reference or quality annotation training data. Li et al. [2017] use this evaluation model as a reward during reinforcement learning and demonstrate strong performance. However, correlation with human judgement was not evaluated. It may be insufficient to rely on a discriminator as a meaningful evaluation of dialog since this assumes that all human responses are perfect and all generated responses are imperfect.

Tao et al. [2018] present RUBER which consists of both a referenced metric and an unreferenced metric. The referenced metric calculates the cosine similarity of word embeddings between system response and a

human reference. The unreferenced metric is a model-based metric trained with a triplet ranking loss. While the referenced metric measures similarity to the ground-truth response, the unreferenced metric of RUBER predicts whether whether a generated response is appropriate in the context of the dialog history. The unreferenced metric of RUBER is an especially important contribution as it demonstrates the efficacy of using self-supervised training to evaluate dialog without comparing to a reference response. This type of evaluation progresses beyond the limitations of both the referenced response and the specific training data used in model-based evaluation.

2.3.4 Evaluation with Pre-Trained Models

The advent of large-scale pre-trained models [Devlin et al., 2018] brought about a new class of dialog evaluation metrics. Similar to the unreferenced metric of Tao et al. [2018], this class of metrics relies on self-supervised training to evaluate dialog without relying on a reference response.

BERT-RUBER [Ghazarian et al., 2019] replaces the RNN in RUBER with a pre-trained BERT model to further improve the performance using contextualized word embeddings. Building on BERT-RUBER, PONE [Lan et al., 2020] improves the training of the unreferenced metric by improving the negative sampling and training the evaluation model on a dataset augmented by NLG models. MAUDE [Sinha et al., 2020] further improves upon the unreferenced metric in RUBER by training with Noise Contrastive Estimation (NCE) [Gutmann and Hyvärinen, 2010] which requires the model to differentiate between a correct response and randomly sampled negative responses.

DEB [Sai et al., 2020] constructs a dialog dataset with manually-created relevant and adversarial irrelevant responses. DEB by first pre-trains BERT on a large-scale dialog corpus and subsequently fine-then on the proposed task-specific dataset with only the next sentence prediction objective. GRADE [Huang et al., 2020] models topic transition dynamics in dialog by constructing a graph representation of the dialog history. This graph is then passed as input to a model that is trained with the same triplet loss as RUBER. Through the graph representation of the dialog history, GRADE better models turn-level topic transition dynamics in dialog. While GRADE is focused on turn-level topic transition dynamics in dialog, DynaEval [Zhang et al., 2021a] uses a graph structure to model the dialog-level interaction between users and systems.

USL-H [Phy et al., 2020] combines three models trained with different objectives: valid utterance prediction (VUP), next utterance/sentence prediction (NSP), and MLM. VUP model decides whether the response is valid in terms of its grammatical correctness. NSP model and MLM model are trained on a dialog corpus to evaluate sensibleness and likelihood of responses respectively. Similarly, the Deep AM-FM metric [Zhang et al., 2021b] measures two aspects of dialog quality through the Adequacy Metric (AM) and the Fluency Metric (FM). AM assesses the semantic similarity of system responses and human references by comparing their BERT embeddings. FM compares the similarity of the language model probabilities for both the system response and the human reference, and produces a higher score if the probabilities are similar.

HolisticEval [Pang et al., 2020] uses GPT-2 [Radford et al., 2019] and pre-trained models natural lan-

guage inference model to assess several qualities of dialog: context coherence, language fluency, response diversity, and logical self-consistency. PredictiveEngage [Ghazarian et al., 2020] incorporates an utterance-level engagement classifier to better assess the overall quality of a response. FlowScore [Li et al., 2021] uses a large pre-trained model to construct dynamic information flow from the dialog history to evaluate the quality of a dialog.

This class of dialog evaluation metrics has achieved promising results, especially in terms of correlation with human judgments. By leveraging large-scale pre-trained models and performing self-supervised training, this class of evaluation metrics is capable to generalize and therefore better able to evaluate dialog.

2.4 Generalization in Dialog

The primary goal of this thesis is to facilitate generalization in models of dialog. The concept of generalization is central to machine learning [Thrun and Mitchell, 1994], and has been an important consideration in the design of data-driven models of dialog. Early neural dialog models, such as those described in Section 2.1, designed architectures and training paradigms in order to induce higher level abstractions that could better generalize to unseen examples in the test set. Though this is a very limited form of generalization, it exemplifies the means by which inductive biases can be constructed to induce certain behaviors in neural models. This section provides an overview of recent work for facilitating generalization to new problems, new inputs and new outputs.

The advent of large-scale pre-training [Devlin et al., 2018; Radford et al., 2019] has been imperative in facilitating generalization in dialog. Pre-trained models allow generalization both to new inputs and to new problems. Pre-trained models can effectively represent a wide distribution of language, and therefore facilitate generalization to inputs beyond those observed in a problem-specific training corpus. Better few-shot generalization has been observed with pre-trained models in intent prediction [Casanueva et al., 2020], slot filling [Coope et al., 2020], state tracking [Heck et al., 2020] and response generation [Peng et al., 2020b]. Furthermore, a single pre-trained model can be adapted to many different dialog problems, without significant modification to the model architecture. BERT [Devlin et al., 2018] has been used for intent prediction [Chen et al., 2019a; Castellucci et al., 2019], slot filling [Chen et al., 2019a; Castellucci et al., 2019], dialog state tracking [Heck et al., 2020] and dialog evaluation [Ghazarian et al., 2019; Sai et al., 2020; Zhang et al., 2021b]. Similarly, the ConveRT model [Henderson et al., 2019], which was trained on dialog data, has been shown to perform well on both intent prediction [Casanueva et al., 2020] and slot filling [Coope et al., 2020]. Pre-trained models achieve strong performance on a variety of different downstream problems and perform much better in few-shot settings.

While pre-trained models have exceled at few-shot generalization to new problems and domains, it is considerably more difficult to generalize to new outputs (i.e., new intents, new slots, etc.) and to generalize in a zero-shot setting. Chen et al. [2016] present a zero-shot approach for learning embeddings for unseen intents, thereby allowing a model to be extended to predict an intent that was unseen at training time. Bapna et al. [2017] show that slot names and their corresponding descriptions can be leveraged to implicitly

align slots across domains and achieve better cross-domain generalization. Wu et al. [2019] similarly use slot names, in combination with a generative model for state tracking, to obtain strong zero-shot results for new slots. Shah et al. [2019] leverage examples for zero-shot slot filling and cross-domain generalization. Generally, approaches for zero-shot generalizability leverage the similarity across different domains (e.g., *restaurant-area* and *hotel-area* are conceptually similar). These approaches can be combined with large-scale pre-trained models, to achieve generalization of language understanding across dissimilar domains [Ruan et al., 2020]. Rastogi et al. [2020b] address zero-shot domain adaptation in state tracking by leveraging BERT [Devlin et al., 2018] with a domain-specific API specification.

Zhao and Eskenazi [2018] present an approach for zero-shot end-to-end dialog that leverages the Action Matching framework to learn a cross-domain latent action space. This cross-domain latent action space serves as an alignment across different domains, thereby facilitating zero-shot generalization to new domains. Qian and Yu [2019] use model-agnostic meta learning to attain stronger results in zero-shot dialog. There is considerably less work in zero-shot generalization to new dialog tasks. This is in part a consequence of the task-specific dialog policies (i.e., how should a system progress the conversation in order to achieve a particular goal) that preclude generalization to new tasks in dialog. For example, it is challenging to transfer an end-to-end dialog model to a new task if the model has no notion of the corresponding dialog policy. This thesis studies this problem and aims to address it, in order to facilitate zero-shot generalization to new tasks.

Chapter 3

Self-Supervised Training as an Inductive Bias

3.1 Introduction

Self-supervised learning is a long-standing concept in artificial intelligence research [Dayan et al., 1999]. Self-supervised learning aims to produce meaningful representations of un-annotated data through reconstruction tasks, such as predicting the next word [Radford et al., 2018] or predicting a masked word [Devlin et al., 2018]. By leveraging vast quantities of un-annotated data that is readily available (i.e., on the internet), large-scale self-supervised pre-training [Devlin et al., 2018; Radford et al., 2018] has brought about a paradigm shift in natural language processing research. Self-supervised training has been shown to facilitate generalization [Wang et al., 2018], with pre-trained models achieving strong performance on a variety of problems and in few-shot settings.

Self-supervised learning is in itself a form of inductive bias. The choice of data and the self-supervised objectives influence *what is learned* by a data-driven model and can therefore be used to facilitate generalization. Self-supervised training can be leveraged to define *the set of assumptions* used by an intelligent system to predict outputs for unseen inputs (i.e., to generalize). Through self-supervised training, data-driven models can learn more meaningful representations which thereby facilitate both generalization to new inputs and generalization to new problems. A model that undergoes self-supervised pre-training may be capable of generalizing to inputs beyond the limitations of the annotated corpus used for supervised training, by leveraging meaningful representations learned during pre-training. Likewise, a model that undergoes self-supervised training can learn sufficiently general representations that improve the performance of problem-agnostic model architectures.

This chapter studies self-supervised learning as an inductive bias and as a means of facilitating generalization in dialog. Section 3.2 examines the role of the self-supervised training data as a mechanism for generalizing to new problems and new domains in few-shot settings. Experiments are carried out with both large-scale pre-training and smaller-scale self-supervised pre-training/multi-tasking. Section 3.3 explores

the use of self-supervised training objectives to model different phenomena for dialog evaluation. By constructing self-supervised training objectives, Section 3.3 demonstrates generalization to new outputs without any annotated training data.

3.2 Self-Supervised Training Data as an Inductive Bias

In an effort to facilitate generalization, this work studies the use of the self-supervised *training data* as an inductive bias. The choice of the self-supervised training data influences the representations learned during self-supervision. Knowledge of the desired generalization (e.g., the downstream data that we want to generalize to) can be used to determine the most appropriate choice of self-supervised training data. Furthermore, by learning sufficiently general representations – the same pre-trained model can be leveraged for multiple downstream problems without many task-specific architectural modifications.

This work aims to address two research questions:

1. Does large-scale pre-training on **open-domain dialog data** result in better performance on downstream dialog tasks?

2. Does self-supervised training on **specific dialog domains** help to induce generalization to those domains?

To address the first question, a BERT-base model is trained in a self-supervised manner on a large open-domain dialog corpus to produce CONVBERT. By pre-training on *dialog data*, CONVBERT should be better suited to modelling dialog, therefore attain improved performance in few-shot settings and better generalize to different downstream dialog problems. Experiments are carried out with CONVBERT on DialoGLUE [Mehri et al., 2020a], which is a benchmark consisting of 7 task-oriented dialog datasets covering 4 distinct natural language understanding tasks.

To address the second question, this work explores the use of task-adaptive self-supervised training as a mechanism for inducing generalization to specific downstream domains and corpora. Rather than pre-training on a large corpus, task-adaptive self-supervised training learns to model the data from the downstream corpus in a self-supervised manner. This work studies both task-adaptive pre-training and multi-tasking. Furthermore, the effect of pre-training with multiple task-oriented corpora is studied.

CONVBERT in combination with task-adaptive self-supervised training matches or exceeds state-of-the-art results on five of the seven DialoGLUE datasets [Mehri et al., 2020a]. Most notably, a **+2.98** improvement is attained in the joint goal accuracy over the best dialog state tracking models on the MultiWOZ corpus. The methods and the experiments presented in this section demonstrate that the choice of self-supervised training data can serve as an inductive bias. The choice of data and self-supervised training algorithm can influence what is learned by the model and thereby facilitate generalization both to new problems and new inputs.

3.2.1 Methods

This section describes the methods employed throughout this work. First, the architectures for the 4 different DialoGLUE tasks (intent prediction, slot filling, dialog state tracking and semantic parsing) are introduced. These architectures are built around an underlying BERT-like model, with minor task-specific modifications. A sufficiently general model of dialog should work well for multiple different downstream problems, without significant modification. Next, the CONVBERT model is introduced. CONVBERT is a BERT model that was further pre-trained on a large open-domain dialog corpus. Finally, task-adaptive self-supervised training is described. Task-adaptive training performs self-supervised training on the data in DialoGLUE to better adapt pre-trained models to the downstream problems and domains.

Task-Specific Architectures

For each of the four different dialog tasks present in DialoGLUE, a task-specific architecture is defined. These architectures are built around an underlying BERT-like model, with minor task-specific architectural modifications.

Intent Prediction: A BERT-like model is fine-tuned to encode an utterance and predict its intent. Specifically, the pooled representation produced by a BERT-like model is passed through a linear layer to predict the intent class.

Slot Filling: For slot filling, the problem is formulated as IOB tagging [Ramshaw and Marcus, 1999] wherein every token in the utterance is labeled as either being the beginning of a slot value (B-), inside a slot value (I-) or not belonging to a slot value (O). A BERT-like model is used to produce a representation of each token, which is then passed through a linear layer that predicts the appropriate tag (e.g., "B-time", "I-people").

Semantic Parsing: The hierarchical representations of the TOP dataset are transformed into (i) a top-level intent for the utterance which corresponds to the root of the tree and (ii) a label for each word of the utterance which is the path from the root to each leaf node (which is always a word). Given this representation, a BERT-like model is trained to simultaneously predict the top-level intent for the utterance using the pooled representation and the labels for each token using the token representation.

Dialog State Tracking: The state tracking architecture is inspired by TripPy Heck et al. [2020] which uses an underlying BERT model and a triple copy strategy to perform state tracking. The TripPy model uses (i) span prediction and a copy mechanism to extract values from a user utterance, (ii) a copy mechanism over concepts mentioned by the system utterance and (iii) a copy mechanism over the *DS memory*, the existing dialog state.

These architectures are held consistent throughout the experiments. An arbitrary BERT-like encoder can be plugged into all of the aforementioned architectures. In this manner, the capabilities of a particular pre-trained model can be evaluated on all of the DialoGLUE tasks, irrespective of the architecture. As such, performance improvements are guaranteed to be a consequence of the improved representations derived through self-supervised training.

ConvBERT

Though large-scale pre-trained models (e.g., BERT) have exhibited strong language understanding capabilities, recent work has suggested that they may be insufficient for modelling dialog, due to the intrinsically goal-driven, linguistically diverse, multi-turn and often informal/noisy nature of dialog [Henderson et al., 2019; Zhang et al., 2019c]. The unique challenges of modelling dialog have been addressed by training on large amounts of *conversational data*, from online forums. This work extends these efforts by fine-tuning BERT on a large open-domain dialog corpus consisting of nearly 700 million conversations to produce CONVBERT.

By training CONVBERT with large amounts of open-domain dialog, it is hypothesized that the resulting model is better able to produce semantically meaningful representations of utterances and multi-turn dialogs. Specifically, an uncased BERT-base model is further pre-trained for 4 epochs on a large-open domain dialog corpus scraped from Reddit, using a masked language modelling objective. During this pre-training, the input is the last 3 turns of dialog context followed by the [SEP] token and the dialog response. The entire input is truncated to have a sequence length of 72, and an Adam optimizer is used with an initial learning rate of $3 \cdot 10^{-4}$.

Task-Adaptive Training

Task-adaptive training is the process of adapting a pre-trained model to a specific task or domain, by performing self-supervised training on the downstream corpus. While large-scale pre-trained models should learn sufficiently general representations – differences between the pre-training data and the downstream data requires generalization to new inputs and may therefore result in lower performance. Performing self-supervised training on the downstream corpus used for fine-tuning has been shown to help with performance in few-shot settings and domain adaptation [Mehri et al., 2019; Gururangan et al., 2020]. In order to adapt BERT-like models to the various DialoGLUE tasks and domains, task-adaptive self-supervised training is performed with a masked language modelling (MLM) objective on each dataset. Several methods are explored for leveraging task-adaptive self-supervised training: (i) self-supervised pre-training prior to supervised fine-tuning on the specific task, (ii) multi-tasking by simultaneously performing self-supervised training and supervised fine-tuning on the task and (iii) both pre-training and multi-tasking. An example experimental setting is as follows: (1) begin with the pre-trained CONVBERT model, (2) conduct self-supervised pre-training on the utterances of HWU, (3) fine-tune on HWU using the intent prediction architecture and simultaneously perform self-supervised training on the utterances of HWU.

To further study the benefits of self-supervised training and the effects of the self-supervised training data, both BERT and CONVBERT are further trained with a masked language modelling over the combination of all the DialoGLUE datasets to produce BERT-DG and CONVBERT-DG. With this intermediate self-supervised training, the resulting models should be better adapted to handle task-oriented dialog and therefore better generalize to the downstream problems and domains in DialoGLUE.

3.2.2 Experiments

Experimental Setup

The experiments are carried out with four BERT-like models: **(1) BERT**-base, **(2) CONVBERT** which is BERT pre-trained on a large corpus of open-domain dialogs, **(3) BERT-DG** which is BERT trained on the full DialoGLUE data in a self-supervised manner and **(4) CONVBERT-DG** which is CONVBERT trained on the full DialoGLUE data in a self-supervised manner.

Experiments are carried out with these four models in four different settings: (1) directly fine-tuning on the target task, (2) self-supervised pre-training with MLM on the target dataset prior to fine-tuning, (3) self-supervised multi-tasking with MLM on the target dataset during fine-tuning and (4) both pre-training and multi-tasking with MLM.

Self-supervised MLM pre-training is performed for 3 epochs prior to fine-tuning. Fine-tuning on a target task is carried out until the performance on the validation set does not improve for 10 epochs. During multi-tasking, the training alternates each epoch between fine-tuning on the target task and self-supervised training with MLM.

To assess the effectiveness of the pre-trained models in low-resource settings, few-shot experiments carried out. In such experiments, self-supervised training is performed only on the few-shot data, which is 10% of the full data. The self-supervised pre-training and multi-tasking is performed with only the few-shot versions of each dataset. However, both BERT-DG and CONVBERT-DG are trained with the full DialoGLUE data, albeit in a self-supervised manner, meaning that they see more in-domain data than either BERT or CONVBERT in the few-shot experiments. For all the few-shot experiments, each model is trained five times with different random seeds and the average performance across the five runs is reported.

The evaluation metrics are consistent with prior work on these datasets. Intent prediction (BANKING77, CLINC150, HWU64) is evaluated with accuracy. The slot filling tasks (RESTAURANT8K, DSTC8), are evaluated with macro-averaged F-1 score as defined by Coope et al. [2020]. Exact-match is used for TOP, which measures how often the hierarchical semantic representation is exactly reconstructed. Dialog state tracking on MULTIWOZ is evaluated with joint goal accuracy.

Results

The results of the full data experiments are shown in Table 3.1. The proposed approaches attain a performance gain over the vanilla BERT model [Devlin et al., 2018] across all seven datasets. These results highlight the efficacy of both the CONVBERT model and the task-adaptive self-supervised training. Across four datasets, the best results are attained by CONVBERT with both self-supervised pre-training and multi-tasking. Table 3.2 compares to prior work, wherein it is shown that CONVBERT in combination with task-adaptive training, matches or exceeds state-of-the-art performance across five out of seven of the datasets. On the dialog state tracking task of MultiWOZ, the proposed approach attains a **+2.98** improvement in the joint goal accuracy over the prior state-of-the-art. These strong results, which hold true across several

Model	Average	BANKING77	HWU64	CLINC150	RESTAURANT8K	DSTC8	TOP	MULTIWOZ
BERT	86.08	93.02	89.87	95.93	95.53	90.05	81.90	56.30
+ Pre	86.18	92.34	91.82	96.27	95.78	89.48	81.54	56.07
+ Multi	85.97	92.27	90.99	96.22	95.61	89.93	81.46	55.30
+ Pre, Multi	85.92	93.20	90.99	95.67	95.04	89.96	82.08	55.06
CONVBERT	86.01	92.95	90.43	97.07	**95.90**	87.58	82.13	56.00
+ Pre	86.19	93.25	92.84	97.09	95.33	87.02	82.00	55.67
+ Multi	85.97	93.20	91.36	97.09	95.39	90.02	**82.63**	56.48
+ Pre, Multi	**86.89**	**93.44**	92.38	**97.11**	95.44	**91.20**	82.08	56.56
BERT-DG	86.11	91.75	90.89	95.98	95.23	90.24	81.16	57.54
+ Pre	86.16	92.01	91.26	96.20	94.61	89.79	81.29	58.00
+ Multi	86.38	92.53	90.61	95.89	95.44	90.81	81.04	58.34
+ Pre, Multi	86.18	92.57	91.26	96.22	95.11	88.69	80.89	58.53
CONVBERT-DG	82.9	93.21	91.64	96.96	93.44	74.54	72.22	58.57
+ Pre	84.1	93.05	**92.94**	**97.11**	95.38	90.88	60.68	58.65
+ Multi	82.78	93.02	91.73	**97.13**	**95.93**	88.97	53.97	**58.70**
+ Pre, Multi	85.34	92.99	91.82	**97.11**	94.34	86.49	76.36	58.29

Table 3.1: Full data experiments on DialoGLUE. The average score on the DialoGLUE benchmark is shown in the leftmost column. The evaluation metrics are intent prediction: accuracy, slot filling: macro-averaged F-1, TOP: exact match: MultiWOZ: joint goal accuracy.

datasets, suggest that large-scale pre-training on open-domain dialog data in combination with task-adaptive self-supervised training transfers effectively to several task-oriented dialog tasks.

When looking at the aggregate performance across all the DialoGLUE tasks, neither CONVBERT nor task-adaptive training independently attain improvements over BERT. However by combining these two approaches, there is a **+0.81** improvement in the average score. This suggests that through large-scale pre-training on open-domain dialog, CONVBERT learns skills that are valuable to DialoGLUE, however it is only through task-adaptive training that these skills are transferred effectively to the downstream problems.

A noteworthy outcome of these experiments is the fact that the BERT model with task-adaptive self-supervised training sometimes outperforms CONVBERT. This indicates that in certain settings, it is more beneficial to perform self-supervised training on the *the downstream dataset* rather than a much larger, but less relevant, dialog corpus. This particularly highlights the importance of the self-supervised data as a mechanism for controlling what is learned by the model, and facilitating generalization to different tasks and domains.

Performing self-supervised training across the combination of the DialoGLUE datasets gives mixed results. CONVBERT-DG attains a significant performance gain on MultiWOZ, suggesting that self-supervised training on other task-oriented dialog corpora helps significantly in modelling MultiWOZ dialogs. Across other datasets, it is only marginally better than the CONVBERT model and sometimes worse. Aside from the unique case of MultiWOZ, it appears that self-supervised training with additional task-oriented dialog

BANKING77 (accuracy) ✓	
USE [Casanueva et al., 2020]	92.81
ConveRT [Casanueva et al., 2020]	93.01
USE + ConveRT [Casanueva et al., 2020]	**93.36**
CONVBERT + Pre + Multi	**93.44**
HWU64 (accuracy) ✓	
USE [Casanueva et al., 2020]	91.25
ConveRT [Casanueva et al., 2020]	91.24
USE + ConveRT [Casanueva et al., 2020]	92.62
CONVBERT-DG + Pre	**92.94**
CLINC150 (accuracy) ✓	
USE [Casanueva et al., 2020]	95.06
ConveRT [Casanueva et al., 2020]	**97.16**
USE + ConveRT [Casanueva et al., 2020]	**97.16**
CONVBERT-DG + Multi	**97.13**
RESTAURANT8K (F-1) ✓	
Span-BERT [Coope et al., 2020]	93.00
V-CNN-CRF [Coope et al., 2020]	94.00
Span-ConveRT [Coope et al., 2020]	**96.00**
CONVBERT-DG + Multi	**95.93**
DSTC8 (F-1)	
Span-BERT [Coope et al., 2020]	91.50
V-CNN-CRF [Coope et al., 2020]	91.25
Span-ConveRT [Coope et al., 2020]	**94.00**
CONVBERT + Pre + Multi	91.20
TOP (Exact Match)	
RNNG [Gupta et al., 2018]	78.51
SR + ELMo [Einolghozati et al., 2019]	**87.25**
SEQ2SEQ-PTR [Rongali et al., 2020]	86.67
CONVBERT + Multi	82.63
MULTIWOZ (Joint Goal Accuracy) ✓	
DST-Picklist [Zhang et al., 2019a]	53.30
TripPy [Heck et al., 2020]	55.30
SimpleTOD [Hosseini-Asl et al., 2020]	55.72
CONVBERT-DG + Multi	**58.70**

Table 3.2: Comparison to prior work on all seven datasets. The proposed models match or exceed state-of-the-art results on five out of seven datasets (marked with checkmarks), with significant improvements (**+3**) on the MultiWOZ corpus.

Model	Average	BANKING77	HWU64	CLINC150	RESTAURANT8K	DSTC8	TOP	MULTIWOZ
BERT	66.07	79.87	81.69	89.52	87.28	45.05	74.38	4.69
+ Pre	66.57	80.72	83.05	89.73	86.37	47.17	74.41	4.55
+ Multi	66.11	79.89	82.32	89.69	87.53	44.92	74.45	3.95
+ Pre, Multi	66.87	81.49	82.70	90.53	86.34	48.55	74.17	4.29
CONVBERT	68.03	83.63	83.77	92.10	86.90	49.08	74.86	5.90
+ Pre	67.36	83.68	83.77	92.10	86.90	45.20	74.92	5.09
+ Multi	68.16	83.15	82.32	92.33	86.71	**50.49**	75.21	5.48
+ Pre, Multi	68.22	83.99	84.52	92.75	86.17	48.40	**78.84**	6.87
BERT-DG	72.70	81.47	83.23	90.57	85.31	43.85	74.80	49.70
+ Pre	72.80	81.79	83.74	90.44	86.66	43.45	74.34	49.40
+ Multi	73.00	81.60	83.18	90.43	86.48	44.86	74.79	49.67
+ Pre, Multi	72.90	81.08	83.40	90.09	86.26	46.32	73.56	**49.86**
CONVBERT-DG	73.75	84.42	85.17	92.87	**87.65**	41.94	75.27	48.94
+ Pre	74.10	84.74	85.63	93.16	86.95	43.61	75.32	49.26
+ Multi	**74.35**	84.09	85.74	93.14	87.48	45.31	75.37	49.35
+ Pre, Multi	73.80	**85.06**	**85.69**	**93.06**	87.58	44.36	72.01	48.89

Table 3.3: Few-shot data experiments on DialoGLUE. The values in this table are averaged across five runs, with different random seeds. The evaluation metrics are intent prediction: accuracy, slot filling: macro-averaged F-1, TOP: exact match: MultiWOZ: joint goal accuracy.

data, beyond just the dataset in question, does not provide significant improvements. For two datasets, DSTC8 and TOP, there is a decrease in performance which may be indicative of catastrophic forgetting. Namely, the CONVBERT-DG model may have lost the language understanding capabilities captured by the CONVBERT model through the additional self-supervised training, and only partially recovers this through the task-specific self-supervised training.

While the proposed models achieve state-of-the-art performance across five of the seven tasks, they underperform on TOP and DSTC8. On the TOP dataset, the best models use sophisticated architectures which have been tailored to the task of semantic parsing [Einolghozati et al., 2019; Rongali et al., 2020]. In this work, the objective is to improve the underlying language encoders in a manner that results in consistent performance gains across all of the tasks. The primary goal is to improve the aggregate improvement across the DialoGLUE benchmark, rather than the performance on a single task. As such, this work tries to avoid complex task-specific architectures when simpler models achieve competitive results.

The results of the few-shot experiments are shown in Table 3.3. The few-shot experiments are particularly important for assessing the generalizability of the methods and their ability to transfer to different downstream problems and domains. In low data environments, self-supervised training on the entirety of the DialoGLUE datasets results in performance gains – with BERT-DG and CONVBERT-DG doing better than BERT and CONVBERT respectively. However, this is not entirely surprising as these models are exposed to more utterances from every dataset, albeit without any of the labels.

Most significantly, on MultiWOZ there is a 40 percent difference between BERT-DG and CONVBERT-DG over BERT and CONVBERT. For state tracking in particular, it appears that observing additional dialog data in a self-supervised setting, results in significant improvements. This may suggest that dialog state tracking is more dependent on having semantically meaningful representations of dialog.

Self-supervised training on the *same* dataset also helps significantly in few-shot environments. Across almost every dataset, the best result is obtained through some form of task-adaptive MLM training. Especially in settings with fewer training examples, adapting the pre-trained models to the domains of the dataset is necessary for good performance on the downstream problems.

CONVBERT is also far more effective in the few-shot experiments than it is in the full data experiments, with a **+1.96** point improvement in the aggregate score over BERT. While the full datasets may be sufficient to effectively transfer BERT to task-oriented dialog, with only 10% of the data, the benefits of the large-scale open-domain pre-training are far clearer.

3.2.3 Discussion

The work described in this section validates the use of self-supervised training data as an inductive bias. The experiments carried out aim to address two research questions about the efficacy of both open-domain dialog data and domain-specific dialog data as a means of inducing generalization.

The strong performance of CONVBERT, particularly in the few-shot experiments, provides an answer to the first research question. Pre-training on **open-domain dialog data** results in better performance on downstream dialog tasks. However, as demonstrated by the results of the full-shot experiments, this effect is diminished by a large downstream corpus. When combined with CONVBERT, task-adaptive self-supervised training facilitates better results in both the full-shot and few-shot settings. Pre-training on the entirety of the DialoGLUE data, unsurprisingly, attains strong results in the few-shot setting. This suggests that using a larger un-annotated corpus for self-supervised training from the same domains can drastically improve performance on the downstream problem.

The few-shot results, especially, address the second research question. Self-supervised training on **specific dialog domains** can induce generalization, both to new inputs (i.e., in the few-shot settings) and to new problems (i.e., same model works for multiple downstream corpora/tasks). The performance of BERT-DG and CONVBERT-DG, particularly demonstrate the effectiveness of leveraging *additional* self-supervised training data from the same domains. However, even in the absence of additional data, task-adaptive self-supervised training on the same domains achieves strong performance gains especially when coupled with CONVBERT.

This work studies the role of the self-supervised *training data* as an inductive bias. Choosing the data used for self-supervised pre-training or task-adaptive self-supervised training influences the representations learned by the model. The experiments in this section demonstrate that choosing self-supervised training data that is *similar* to the downstream corpus can often induce better performance, especially when this data is not already observed during downstream fine-tuning (e.g., the BERT-DG and CONVBERT-DG few-shot

experiments). In this manner, a developer that is aware of the desired type of generalization can choose the data used for self-supervised learning in order to best facilitate the necessary generalization.

3.3 Self-Supervised Training Objectives as an Inductive Bias

The lack of meaningful automatic evaluation metrics is a significant impediment for open-domain dialog generation research. Standard language generation metrics have been shown to be ineffective for dialog evaluation [Deriu et al., 2019; Liu et al., 2016]. Without well-accepted, meaningful automatic metrics, open-domain dialog researchers have come to rely on human evaluation. Due to its time- and cost-intensive nature, human evaluation is typically only used for the final dialog model. As such, during development dialog systems are generally optimized with poorly-correlated automatic metrics (e.g., F-1, BLEU, PPL) which can result in sub-par human evaluation scores [Dinan et al., 2019]. To facilitate development of open-domain dialog models with meaningful automatic metrics, this section presents the **U**n**S**upervised and **R**eference free (**USR**) [Mehri and Eskenazi, 2020b] evaluation metric for dialog.

Standard automatic metrics for evaluating dialog generation (e.g., BLEU, F-1, METEOR, ROUGE) have several shortcomings that make them unsuitable for dialog evaluation: (1) The one-to-many nature of dialog [Zhao et al., 2017b] makes word-overlap metrics ineffective for scoring valid system output that deviates from the ground-truth response [Liu et al., 2016; Gupta et al., 2019]. (2) Human evaluation of dialog typically measures multiple properties (e.g., appropriate, interesting, consistent). Automatic metrics on the other hand, condense the multi-faceted nature of dialog quality to a single uninterpretable metric. (3) There are many definitions of what a *good dialog* is and, as such, it is not feasible to construct a *"one size fits all"* metric. Depending on the task and the data, the desired qualities of a dialog system may differ Walker et al. [1997]; Deriu et al. [2019].

The problem of dialog evaluation is inherently one of generalization. To estimate the quality of generated responses for open-domain dialog, an evaluation metric must be able to model language it has not observed at training time. In part, this is because there is limited quality-annotated data (i.e., dialog context, response, and quality score) for training evaluation metrics. Further, the domain and the topic of the dialogs is generally unbounded in open-domain dialog – therefore an effective dialog evaluation metric must be able to generalize to a wide variety of inputs. In addition to requiring generation to new inputs, dialog evaluation requires generalization to new outputs. Dialog evaluation is multi-faceted and consists of multiple different qualities (e.g., relevance, fluency, engagingness). Without annotated training data for these various qualities, a dialog evaluation metric must generalize to new outputs in an unsupervised or self-supervised manner.

This work studies the role of self-supervised *training objectives* as a mechanism for facilitating generalization to new inputs and new outputs. USR consists of self-supervised training objectives that approximate different qualities for open-domain dialog evaluation. Three self-supervised objectives are designed as a proxy for five different qualities of dialog (relevance, fluency, interestingness, etc.). The task of open-domain dialog evaluation fundamentally requires generalization, both to new inputs (topics, domains, etc.) and new outputs (different qualities). Without any annotated data, the proposed USR metric *must* rely on

self-supervised training objectives that aim to model different dialog qualities. Through self-supervised training with specific objectives, the resulting models are able to model different qualities and achieve reasonable correlations with human judgments. USR (1) alleviates the one-to-many issue of standard metrics through its reference-free nature, (2) produces interpretable measures for multiple desirable properties of dialog, and (3) provides a configurable mechanism for combining several sub-metrics into an overall quality score.

To evaluate the performance of USR, human quality annotations were collected for models trained on the Topical-Chat corpus [Gopalakrishnan et al., 2019] and the PersonaChat corpus [Zhang et al., 2018]. USR is shown to strongly correlate with human judgment on both Topical-Chat (turn-level Spearman: **0.42**, system-level Spearman: **1.0**) and PersonaChat (turn-level Spearman: **0.48** and system-level Spearman: **1.0**). The strong correlation with human judgments across two datasets and a variety of model types demonstrates that well-motivated self-supervised training objectives can facilitate generalization to new inputs and new outputs.

3.3.1 Methods

This section describes the USR metric, a self-supervised reference-free evaluation metric for dialog. USR leverages pre-trained language models, specifically RoBERTa [Liu et al., 2019b], to measure various qualities of dialog. USR is designed to be reference-free because there is no *one right answer* due to the inherent one-to-many nature of dialog [Zhao et al., 2017b].

Several sub-metrics were developed for five different qualities of dialog: understandable, natural, interesting, maintains context and uses knowledge[1]. Each of these sub-metrics is trained in a self-supervised manner with a specific training objective, which aims to approximate a specific quality. Each of USR's sub-metrics assess specific dialog qualities, and USR aggregates the outputs of these sub-metrics to produce an overall assessment of a response. By aggregating several sub-metrics to form an overall score, USR facilitates better understanding the performance of a response generation model.

Masked Language Modelling

The masked language modelling (MLM) sub-metric aims to approximate whether an utterance is understandable and natural. The sub-metric fine-tunes RoBERTa [Liu et al., 2019b] in a self-supervised manner with a masked language modelling training objective. By fine-tuning on a specific corpus, either TopicalChat [Gopalakrishnan et al., 2019] or PersonaChat [Zhang et al., 2018], the resulting sub-metric is better suited to evaluate whether a particular response matches the language of the corpus. By computing the likelihood of the response estimated by the fine-tuned RoBERTa model, the MLM sub-metric can identify erroneous and potentially non-fluent responses.

The RoBERTa-base model [Liu et al., 2019b] is fine-tuned in a self-supervised manner with a masked language modelling objective (MLM) on the training set of a dialog corpus, either TopicalChat [Gopalakr-

[1] A thorough definition of these qualities can be found in Mehri and Eskenazi [2020b].

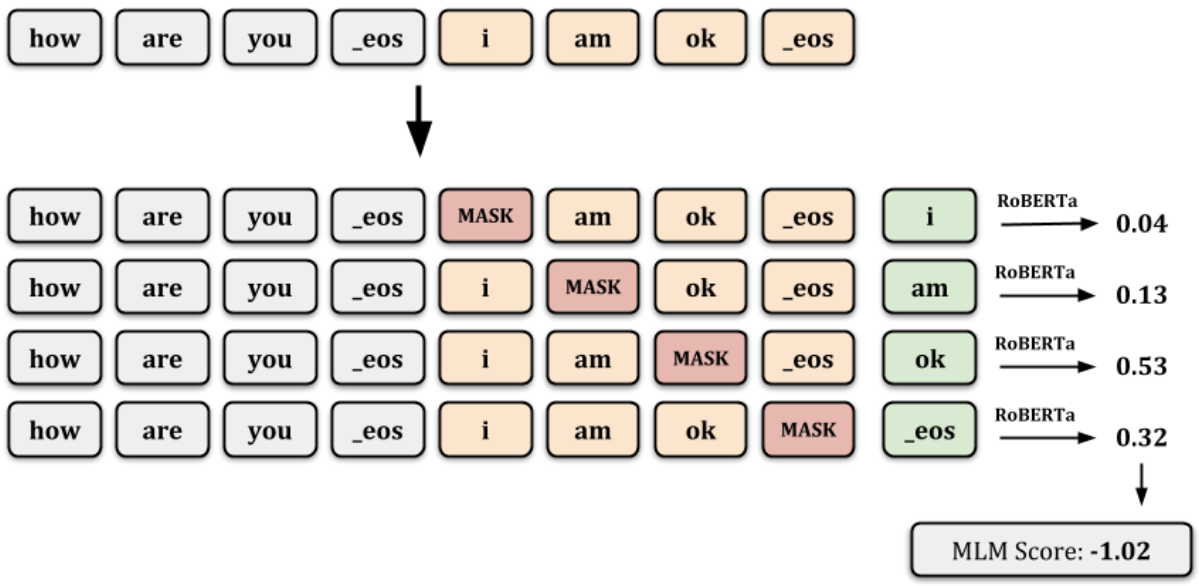


Figure 3.1: Visualization of the masked language modelling (MLM) metric. Context words are in grey; response words are in red. The red words are masked, and RoBERTa must predict the likelihood of their true value (shown in green).

ishnan et al., 2019] or PersonaChat [Zhang et al., 2018], using the implementation open-sourced by Wolf et al. [2019]. The MLM fine-tuning is performed on only the dialog, without any of the facts, for a single epoch.

RoBERTa uses both past and future context to predict a probability distribution for a masked word. The input sequence to MLM is a concatenation of a dialog context, c, and a response, r. One word at a time, each word in r is masked and its log likelihood is computed. Given the masked log-likelihood for the i-th word of r as l_i, the value of the metric is then computed to be $-\sum_i^{|r|} l_i$. Figure 3.1 visualizes this process.

Dialog Retrieval Metrics

Two self-supervised training objectives are defined in order to approximate three dialog qualities: *maintains context*, *interesting* and *uses knowledge*. Both training objectives are based on dialog retrieval. Conditioned on some context x, a response r and a binary label $y \in \{0, 1\}$ indicating whether r is the true response or a negatively sampled one, both training objectives model the probability $P(y = 1|\, x, r)$.

To measure the qualities *maintains context* and *interesting*, the value of x is defined to be the concatenation of the dialog history, c, and any knowledge that the response may have been conditioned on, f. This is intuitive for maintains context, as during self-supervision the sub-metric learns to score relevant responses higher than negatively sampled responses. As such, a generated response that maintains context should have a higher $P(y = 1|\, c, f, r)$. The same model is also a good estimator if whether a response is interesting, because a dull or generic response (e.g., *'ok'* or *'that is cool'*) will occur more frequently in the training corpus

and is therefore more likely to be a negatively sampled response. In contrast, an interesting/unique response is less likely to be a negatively sampled response. As such, responses that are frequent in the dataset and therefore dull/generic will have a lower $P(y = 1|\ c, f, r)$ score.

To measure *uses knowledge*, x is defined to be the knowledge the response is conditioned on. The model measures $P(y = 1|\ f, r)$. During training, the model learns to produce higher scores for responses that leverage the knowledge, and a lower score for negatively sampled responses. As such, when used for evaluation, the model will produce a higher score for responses that effectively use the provided knowledge.

USR

Given the three sub-metrics which approximate the five dialog qualities, USR combines the scores into an overall measure that correlates well with *overall quality* ratings for each response.

Given a dataset of human annotations for the five dialog qualities and an *overall quality* score, a regression model is trained to predict the overall score from the quality scores. The predictions of this regression model attained a 0.9654 Spearman correlation using the annotated quality scores. To combine the outputs of the three sub-metrics into an overall score for the response, the same regression model is leveraged by USR.

USR combines its sub-metrics into one measure of overall quality. This combination is configurable, adaptable to different datasets or tasks. For example, if a specific application prefers natural responses over interesting ones, the weights of the regression model can be adjusted. In this manner, USR could potentially be adapted to different settings or to the preferences of specific users.

3.3.2 Experiments

To assess the performance of the USR metric, as well as the three sub-metrics (MLM, USR-DRc and USR-DRf), several experiments are carried out. The objective of these experiments is to measure the correlation of various automatic evaluation metrics to human judgments.

Human Quality Annotations

To measure the correlation of various automatic metrics to human judgments, a dataset of human quality annotations was collected. For both PersonaChat and TopicalChat, several generation models were trained and used to produce multiple responses. In addition to the model-generated responses, human produced an additional response. These responses were then annotated by three individuals each, with each response being annotated for the five specific dialog qualities (understandable, natural, interesting, maintains context, uses knowledge) as well as a measure of overall quality. More details about the collection of this dataset can be found in Mehri and Eskenazi [2020b].

Metric	Spearman	Pearson
Understandable		
BERTScore (base)	0.2502	0.2611
USR - MLM	**0.3268**	**0.3264**
USR	0.3152	0.2932
Natural		
BERTScore (base)	0.2094	0.2260
USR - MLM	**0.3254**	**0.3370**
USR	0.3037	0.2763
Maintains Context		
METEOR	0.3018	0.2495
USR - DR (x = c)	0.3650	0.3391
USR	**0.3769**	**0.4160**
Interesting		
BERTScore (base)	0.4121	0.3901
USR - DR (x = c)	**0.4877**	0.3533
USR	0.4645	**0.4555**
Uses Knowledge		
METEOR	0.3909	**0.3328**
USR - DR (x = f)	**0.4468**	0.2220
USR	0.3353	0.3175

Table 3.4: Turn-level correlations on Topical-Chat. We show: (1) best non-USR metric, (2) best USR sub-metric and (3) USR metric. All measures in this table are statistically significant to $p < 0.01$.

Metric	Spearman	Pearson
Understandable		
BERTScore (base)	*0.0685*	*0.0672*
USR - MLM	*0.1186*	**0.1313**
USR	**0.1324**	*0.1241*
Natural		
VectorExtrema	0.1375	0.1458
USR - DR (x = c)	0.2291	0.1733
USR	**0.2430**	**0.1862**
Maintains Context		
METEOR	0.2564	0.2500
USR - DR (x = c)	**0.5625**	0.6021
USR	0.5280	**0.6065**
Interesting		
BERTScore (base)	*0.0491*	*0.0325*
USR - DR (x = c)	**0.2634**	*0.0606*
USR	*0.0171*	*0.0315*
Uses Knowledge		
METEOR	0.1719	0.1678
USR - DR (x = c)	**0.6309**	**0.4508**
USR	0.3177	0.4027

Table 3.5: Turn-level correlations on Persona-Chat. We show: (1) best non-USR metric, (2) best USR sub-metric and (3) USR metric. All values with $p > 0.05$ are italicized.

Metric	Spearman	Pearson
Word-Overlap Metrics		
F-1	0.1645	0.1690
BLEU-1	0.2728	0.2876
BLEU-2	0.2862	0.3012
BLEU-3	0.2569	0.3006
BLEU-4	0.2160	0.2956
METEOR	0.3365	0.3908
ROUGE-L	0.2745	0.2870
Embedding Based Metrics		
Greedy Matching	0.1712	0.1943
Embedding Average	0.1803	0.2038
Vector Extrema	0.2032	0.2091
Skip-Thought	*0.1040*	*0.1181*
BERTScore (base)	0.3229	0.3540
BERTScore (large)	0.2982	0.3252
Reference Free Metrics		
USR - MLM	0.3086	0.3345
USR - DR (x = c)	0.3245	0.4068
USR - DR (x = f)	0.1419	0.3221
USR	**0.4192**	**0.4220**

Table 3.6: Turn-level correlations between all automatic metrics and the *Overall Quality* ratings for the Topical-Chat corpus. All values with $p > 0.05$ are italicized.

Metric	Spearman	Pearson
Word-Overlap Metrics		
F-1	0.1422	*0.1241*
BLEU-1	*0.0434*	*0.0469*
BLEU-2	*0.1122*	*0.0943*
BLEU-3	*0.1202*	*0.0924*
BLEU-4	0.1353	*0.0899*
METEOR	0.2527	0.2713
ROUGE-L	*0.0659*	*0.0385*
Embedding Based Metrics		
Greedy Matching	*0.0916*	*0.0625*
Embedding Average	*0.1182*	0.1428
Vector Extrema	0.1570	0.1410
Skip-Thought	*-0.0393*	*-0.0452*
BERTScore (base)	0.1690	0.1526
BERTScore (large)	0.1518	*0.1211*
Reference Free Metrics		
USR-MLM	*0.0795*	*0.0788*
USR-DR (x = f)	*-0.0495*	*-0.0454*
USR-DR (x = c)	**0.4814**	**0.6087**
USR	0.4693	0.4115

Table 3.7: Turn-level correlations between all automatic metrics and the *Overall Quality* ratings for the PersonaChat corpus. All values with $p > 0.05$ are italicized.

Results

The performance of the proposed USR metric, as well as several prior metrics, are evaluated by measuring correlation to human judgements. The output of the sub-metrics that approximate each dialog quality are used as input for the regression model of the USR metric. While the best performing sub-metrics are not consistent across the two datasets, the USR metric nonetheless consistently exhibits strong results.

Table 3.4 shows turn-level correlations of the best automatic metrics for each dialog quality on Topical-Chat. USR is shown to strongly outperform both word-overlap and embedding-based metrics across all of the dialog qualities. Interestingly, the best non-USR metric is consistently either METEOR or BERTScore – possibly because both methods are adept at comparing synonyms during referenced evaluation. For some dialog qualities, the overall USR metric outperforms the best sub-metric. For example, USR does better for *maintains context* than USR-DR. This is because the information from the other sub-metrics (e.g., *uses knowledge*) is valuable and effectively leveraged by USR.

Table 3.5 reports the turn-level correlations of the best automatic metrics for each dialog quality on the PersonaChat corpus. Across all dialog qualities, USR strongly outperforms the word-overlap and embedding-based metrics. Conversations in PersonaChat generally consist of individuals communicating facts from their own persona in a relevant and coherent manner. As such, when models trained on PersonaChat produce subpar outputs, it is generally because the outputs either (1) do not effectively use the persona or (2) are not relevant/coherent to the dialog context. This explains why the correlations are significantly higher for *maintains context* and *uses knowledge*. As a consequence of PersonaChat's strong dependency on both the dialog context and the persona, USR-DR (x = c) which uses both the dialog context and the persona to perform dialog retrieval, generally outperforms all other metrics.

Table 3.6 shows turn-level correlation with the *overall quality* ratings on Topical-Chat, for all of the automatic metrics. USR shows a strong improvement over all other methods. This strong performance can be attributed to two factors: (1) the ability of MLM and DR to accurately quantify qualities of a generated response without a reference response, and (2) the ability of USR to effectively combine MLM and DR into a better correlated overall metric.

USR shows a similar improvement over all other metrics on PersonaChat, as shown in Table 3.7. However, DR (x = c) outperforms USR despite the fact that four out of the five sub-metrics input into the USR regression are DR (x = c). This result is probably due to PersonaChat's strong dependancy on both dialog context and persona, both of which DR (x = c) explicitly leverages.

The system-level correlation between all automatic metrics and the *overall quality* ratings is compared. USR significantly ($p < 0.01$) outperforms all other metrics with a Spearman correlation of **1.0** on both datasets and Pearson correlations of **0.92** (Topical-Chat) and **0.82** (PersonaChat). The full set of system-level correlations can be found in Mehri and Eskenazi [2020b].

These results demonstrate USR's effectiveness. It strongly outperforms other metrics on both turn-level and system-level correlations. This signifies the effectiveness of self-supervised training objectives as a means of influencing what is learned by the model, and facilitating generalization to multiple dialog

qualities in the absence of annotated training data.

3.3.3 Discussion

The strong performance of USR across both Topical-Chat and PersonaChat validates the initial hypothesis that self-supervised *training objectives* can be leveraged as an inductive bias. Without any annotated training data, the USR metric is shown to effectively approximate five different fine-grained qualities of dialog and aggregate these into an overall quality score.

This work demonstrates that the choice of self-supervised training objective can influence what a model learns, and therefore be leveraged to model different qualities of dialog. Masked language modelling is shown to effectively model understandability/fluency. Dialog retrieval (using both the dialog history and the knowledge) is shown to approximate maintains context/interesting and dialog retrieval (with only the knowledge) can estimate whether a response *uses knowledge*.

A limitation of this work is that the self-supervised models that make up USR are unable to extend beyond the limitations of their specific training data. For example, the MLM sub-metric will assign a lower likelihood to responses that differ significantly from the language of the training corpus. While this fine for evaluating response generation models that are trained only on the datasets that were used to fine-tune USR, it is a problem when considering response generation models trained on other corpora. This problem can be mitigated by using the learnings from the previous study to extend the capabilities of the USR metric (or an equivalent), by selecting the most appropriate training data for self-supervised training.

3.4 Conclusion

This chapter demonstrates that self-supervised training is an inductive bias and is central to the notion of generalization. The first study shows that the self-supervised training data is an inductive bias by which a system developer can influence the data-driven model and thereby induce desired generalizations. Concretely, through CONVBERT and task-adaptive self-supervised training it is found that training on data which is similar to the downstream corpus induces better few-shot generalization. The second study validates the use of the self-supervised training objectives as an inductive bias that influences what is learned by the model and can therefore be leveraged to facilitate zero-shot generalization to new outputs. The USR metric is shown to correlate well with human judgements of dialog quality, through self-supervised training with well-motivated objectives. These two studies demonstrate that generalization can be induced through the choice of self-supervised training data and training objectives.

Chapter 4

Inductive Bias in the Model Architecture

4.1 Introduction

The model architecture defines the mechanisms by which a data-driven model processes inputs and generates outputs. The architecture plays an important role in defining the abstractions learned by a model, and can therefore be used to influence the set of assumptions by which it makes predictions for unobserved inputs (i.e., the mechanism by which it generalizes). Hand-crafted architectures for dialog models have achieved strong performance on a variety of problems [Bhargava et al., 2013; Hakkani-Tür et al., 2016; Serban et al., 2017]. Researchers incorporate knowledge about the domain and the problem into the design of an architecture, in order to influence the abstractions learned by the model. For example, the Hierarchical Encoder Decoder (HRED) [Serban et al., 2017] is designed to first encode every utterance in a dialog and then produce a dialog-level representation by considering a sequence of utterance-level representations. This hierarchical structure is an inductive bias, motivated by the inherent hierarchical structure in dialog, that forces a data-driven model to independently consider each utterance prior to producing a dialog-level representation.

This chapter explores mechanisms of incorporating inductive biases into the model architecture in order to induce generalization. Through modifications to the model architecture and the training algorithm, the resulting models are forced to learn specific high-level abstractions which are conductive to different types of generalization. The design of these inductive biases is motivated by knowledge of the problem (i.e., the tasks, data, evaluation metrics) and the desired generalization. Through modifications to the model architecture, the process of predicting an outputs from a given input is *prescribed* by the system developer (i.e., compare an input to examples from the training set before predicting an output). Through controlling the process and the learned abstractions, certain types of generalization can be induced. The work in this chapter aims to validate the hypothesis that architectural modifications, motivated by domain knowledge of the problem and the desired generalizations, can effectively serve as inductive biases that facilitate generalization.

Two studies are carried out to validate this hypothesis. Section 4.1 presents Structured Fusion Networks, which incorporate the traditional dialog pipeline into neural response generation models. Structured

Fusion Networks *prescribe* a specific procedure by which the model must perform the task of response generation. Through this specific procedure, which serves as an inductive bias, the resulting model performs better in low-resource settings and can generalize to new domains in few-shot settings. Section 4.2 introduces example-driven training for the problem of intent prediction, and demonstrates generalization to *new domains* and to *new outputs*. In example-driven training, the output class is predicted by measuring the similarity of a given input with a set of examples. By making the representation-to-intent mapping an explicit non-parametric process (rather than an implicitly learned set of weights in a classification layer), the resulting model is able to generalize to new outputs and can predict unseen intents, given the corresponding examples.

4.2 Structured Fusion Networks

End-to-end neural dialog systems have exhibited strong performance [Vinyals and Le, 2015; Dinan et al., 2019]. However such models suffer from a variety of shortcomings, including: a data-hungry nature [Zhao and Eskenazi, 2018], a tendency to produce generic responses [Li et al., 2016b], an inability to generalize [Mo et al., 2018; Zhao and Eskenazi, 2018], a lack of controllability [Hu et al., 2017], and divergent behavior when tuned with reinforcement learning [Lewis et al., 2017]. Traditional pipeline dialog systems, which are generally free of these problems, consist of three distinct components: the natural language understanding (NLU), which produces a structured representation of an input (e.g., a belief state); the natural language generation (NLG), which produces output in natural language conditioned on an internal state (e.g. dialog acts); and the dialog manager (DM) [Bohus and Rudnicky, 2009], which describes a policy that combines an input representation (e.g., a belief state) and information from some database to determine the desired continuation of the dialog (e.g., dialog acts). A traditional dialog system, consisting of an NLU, DM and NLG, is pictured in Figure 4.1.

The structured components of pipeline dialog systems facilitate effective generalizability, interpretability, and controllability. The structured output of each component allows for straightforward modification, interpretation and tuning of the system. On the other hand, end-to-end neural models of dialog lack explicit structure and are often treated as a black box. To this end, this work explores several methods of incorporating the structure of traditional dialog systems into neural dialog models. In this manner, domain knowledge of dialog can be used to prescribe a specific procedure upon a model of dialog. The structured pipeline is used as an inductive bias in the neural architecture, to improve performance in low-resource settings and to facilitate generalization to new inputs. This work studies the effect of inductive biases in the model architecture, motivated by decades of research on pipeline dialog systems, as a means of inducing generalization to new inputs and domains.

First, several neural *dialog modules* are constructed to serve the role of the NLU, the DM and the NLG. Next, a number of methods are proposed for incorporating these dialog modules into end-to-end dialog systems, including Naïve Fusion, Multitask Fusion and Structured Fusion Networks (SFN). Experiments demonstrate that SFN obtains strong results on the MultiWOZ dataset [Budzianowski et al., 2018] both

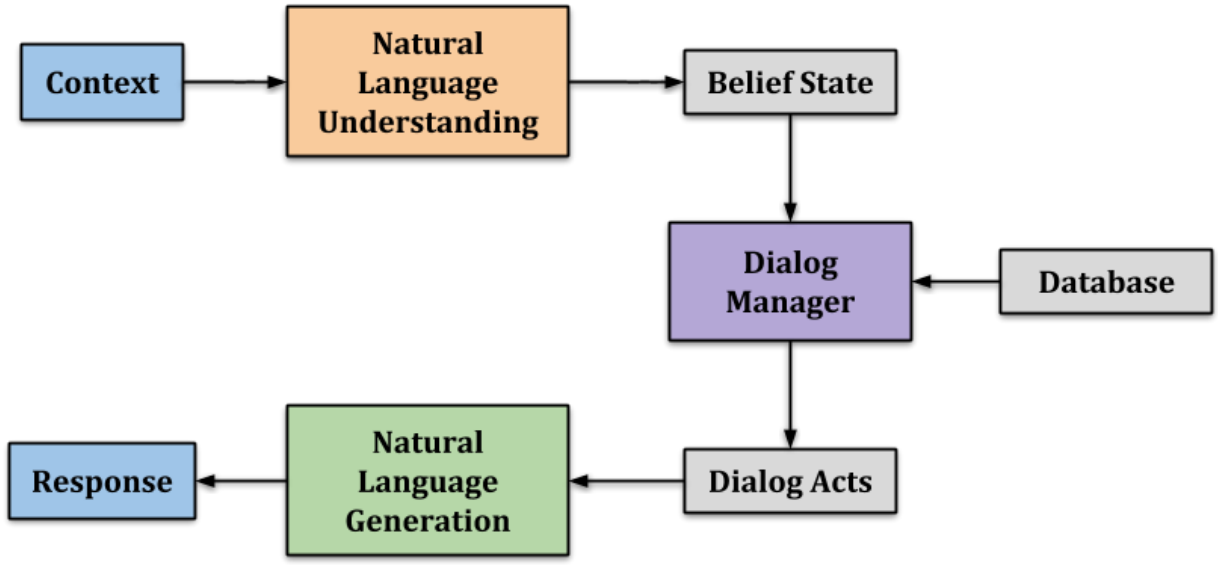


Figure 4.1: A traditional dialog system consisting of a natural language understanding (NLU), dialog manager (DM) and natural language generation (NLG).

with and without the use of reinforcement learning. Due to the explicit structure of the model, SFN is shown to exhibit several valuable properties including improved performance in few-shot settings, improved generalization to new domains and robustness to divergence during reinforcement learning [Lewis et al., 2017].

4.2.1 Methods

This section describes the methods employed for the task of dialog response generation. First, the baseline sequence-to-sequence model proposed by Budzianowski et al. [2018] is described. Next, several methods of incorporating a pipeline structure into end-to-end neural dialog models are introduced. Finally, a reinforcement learning paradigm with structured fusion networks is defined.

Sequence-to-Sequence

The baseline model for the task of response generation, depicted in Figure 4.2, consists of a standard encoder-decoder framework [Sutskever et al., 2014], augmented with a belief tracker (obtained from the annotations of the dialog state) and a database vector. The dialog system is tasked with producing the appropriate system response, given a dialog context, an oracle belief state representation and a vector corresponding to the database output.

The dialog context is encoded using an LSTM [Hochreiter and Schmidhuber, 1997] sequence-to-sequence network [Sutskever et al., 2014]. Experiments are conducted with and without an attention mechanism [Bah-

danau et al., 2015]. Given the final encoder hidden state, h_t^e, the belief state vector, v_{bs}, and the database vector, v_{db}, Equation 4.1 describes how the initial decoder hidden state is obtained.

$$h_0^d = tanh\left(W_e h_t^e + W_{bs} v_{bs} + W_{db} v_{db} + b\right) \tag{4.1}$$

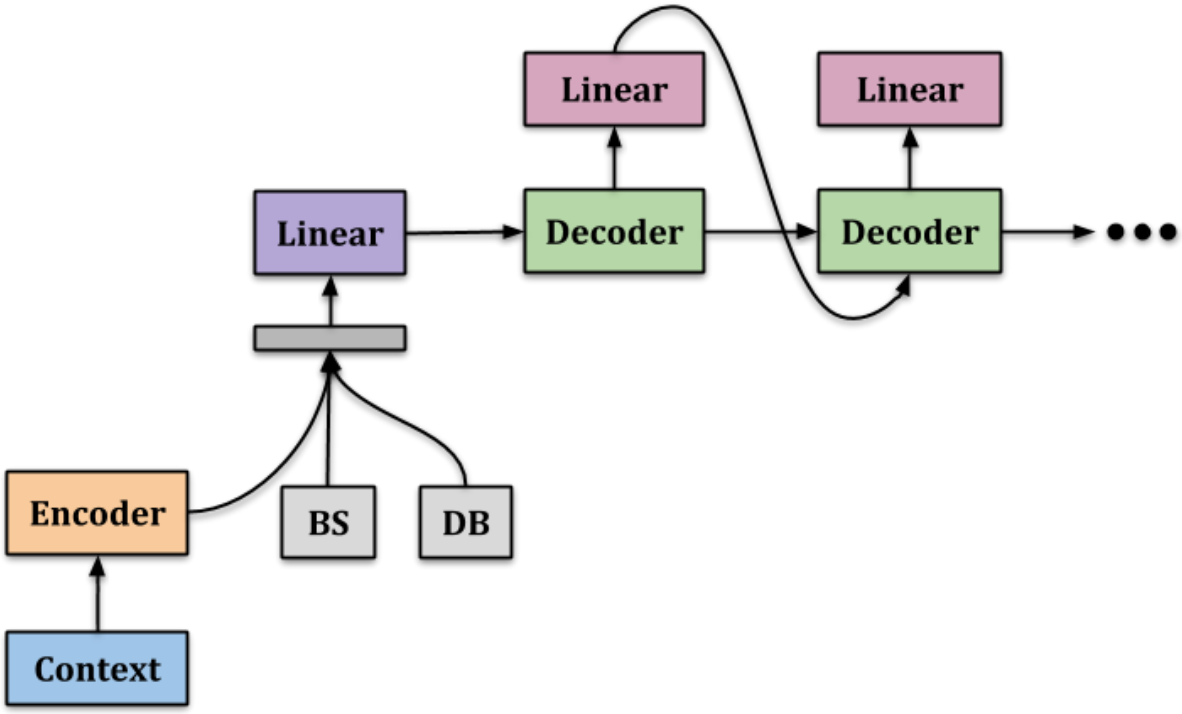


Figure 4.2: A diagram of the baseline sequence-to-sequence architecture. The attention mechanism is not visualized, however experiments are conducted both with and without attention.

Neural Dialog Modules

As depicted in Figure 4.1, a traditional dialog system consists of the NLU, the DM and the NLG. The NLU maps a natural language input to a belief state representation (BS). The DM uses the belief state and some database output, to produce dialog acts (DA) for the system response. The NLG uses the dialog acts to produce a natural language response.

A neural *dialog module* is constructed for each of these three components. A visualization of these architectures is shown in Figure 4.3. The NLU architecture uses an LSTM encoder to map the natural language input to a vector representation, h_t, which is then passed through a linear layer and a sigmoid function to obtain a multi-label prediction of the belief state. The DM module projects the belief state and database vector into a latent vector, through the use of a linear layer with a ReLU activation, which is then passed through another linear layer and a sigmoid function to predict the dialog act vector. The neural

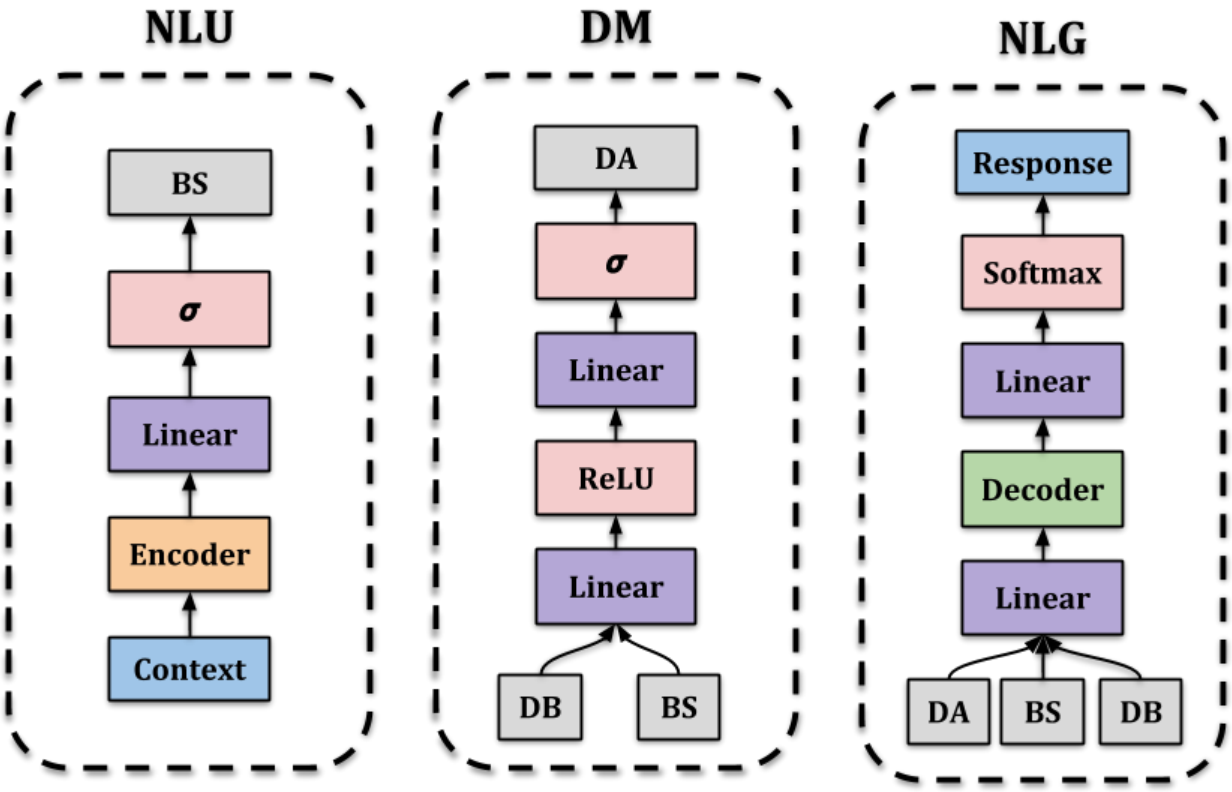


Figure 4.3: A visualization of the neural architectures for each of the three modules of traditional dialog systems.

architecture corresponding to the NLG is a conditioned language model with its initial hidden state given by a linear encoding of the dialog acts, belief state and database vectors.

The following equations define the structure of the modules, where the gt subscript on an intermediate variable denotes the use of the ground-truth value:

$$bs = \textbf{NLU}(\texttt{context}) \tag{4.2}$$

$$da = \textbf{DM}(bs_{gt}, db) \tag{4.3}$$

$$\texttt{response} = \textbf{NLG}(bs_{gt}, db, da_{gt}) \tag{4.4}$$

Naïve Fusion

Naïve Fusion (NF) is a straightforward mechanism for using the neural dialog modules for end-to-end dialog response generation. Both Zero-Shot Naïve Fusion and Naïve Fusion with Fine-tuning are explored.

In Zero-Shot Naïve Fusion, each dialog module is trained independently, meaning that it is given the ground truth input and supervision signal. However, during inference, the intermediate values (e.g., the dialog act vector) do not necessarily exist and the outputs of other neural modules must be used instead. For

example, the DM module is trained given the ground-truth belief state as input, however during inference it must rely on the belief state predicted by the NLU module. This results in a propagation of errors, as the DM and NLG may receive imperfect input.

Zero-Shot Naïve Fusion combines the pre-trained neural modules at inference time. The construction of the response conditioned on the context, is described as follows:

$$bs = \textbf{NLU}(\texttt{context}) \tag{4.5}$$

$$\texttt{response} = \textbf{NLG}(bs, db, \textbf{DM}(bs, db)) \tag{4.6}$$

Since the forward propagation described in Equations 4.5 and 4.6 is continuous and there is no sampling procedure until the response is generated, Naïve Fusion can be fine-tuned for the end-to-end task of dialog generation. In Naïve Fusion with Fine-tuning, The pre-trained neural modules are combined as described above, and fine-tuned on the task of dialog generation using the same data and learning objective as the baseline sequence-to-sequence model.

Multi-task Fusion

The structured pipeline of traditional dialog system can be incorporated into neural architectures through the use of multi-tasking. Multi-task Fusion (MF) is a method where the end-to-end generation task is learned simultaneously with the aforementioned dialog modules. The multi-tasking setup is seen in Figure 4.4.

By sharing the weights of the end-to-end architecture and each respective module, the learned representations should become stronger and more structured in nature. For example, the encoder is shared between the NLU module and the end-to-end task. As such, it will learn to both represent the information necessary for predicting the belief state vector and any additional information useful for generating the next utterance. Multi-tasking in this manner serves as an inductive bias that can facilitate generalization. Given an unobserved input, the encoder will produce representations which are more grounded in the information relevant to the belief state.

Structured Fusion Networks

Structured Fusion Networks (SFN), depicted in Figure 4.5, use the independently pre-trained neural dialog modules for the task of end-to-end dialog generation. Rather than fine-tuning or multi-tasking the independent modules, SFN aims to learn a higher-level model on top of the neural modules to perform the task of end-to-end response generation.

The output of the NLU is concatenated at each time-step of the encoder input. The output of the DM is similarly concatenated to the input of the linear layer between the encoder and the decoder of the higher-level model. The output of the NLG, in the form of logits at a decoding time-step, is combined with the hidden state of the decoder via cold-fusion [Sriram et al., 2017]. Given the NLG output as l_t^{NLG} and the higher-level decoder hidden state as s_t, the cold-fusion method is described as follows:

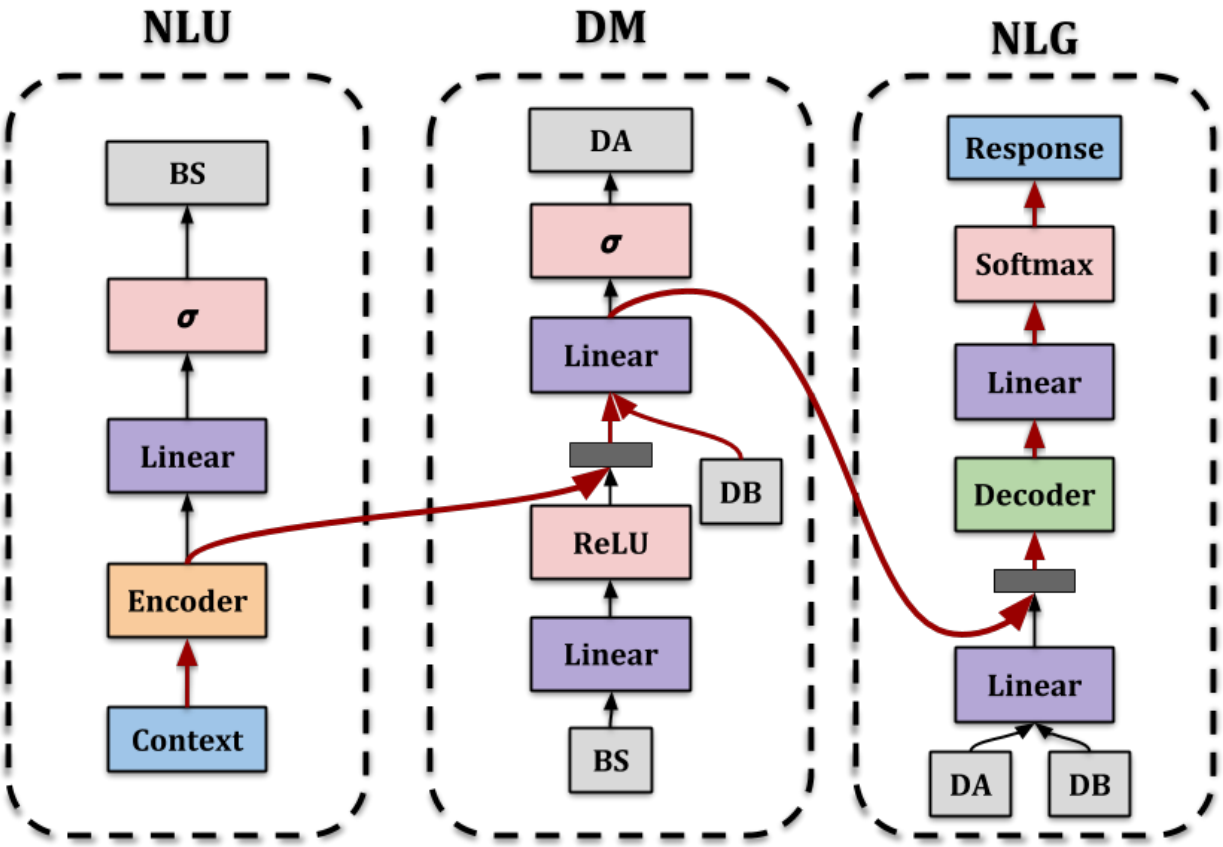


Figure 4.4: A depiction of Multitask Fusion, where the individual neural modules are learned simultaneously with the end-to-end task of dialog generation. The dashed boxes contain the individual components, while the red arrows depict forward propagation for the end-to-end task. The red arrows are the process used during response generation.

$$h_t^{NLG} = DNN(l_t^{NLG}) \tag{4.7}$$

$$g_t = \sigma(W[s_t; h_t^{NLG}] + b) \tag{4.8}$$

$$s_t^{CF} = [s_t; g_t \circ h_t^{NLG}] \tag{4.9}$$

$$y_t = softmax(DNN(s_t^{CF})) \tag{4.10}$$

By pre-training the modules and using their structured outputs, the higher-level model does not have to *re-learn* and *re-model* the dialog structure (i.e., representing the belief state and dialog acts). Instead, it can focus on the more abstract modelling that is necessary for the task, including recognizing and encoding complex natural language input, modelling a policy, and effectively converting a latent representation into a natural language output according to the policy.

The SFN architecture may seem complex due to the redundancy of the inputs. For example, the context

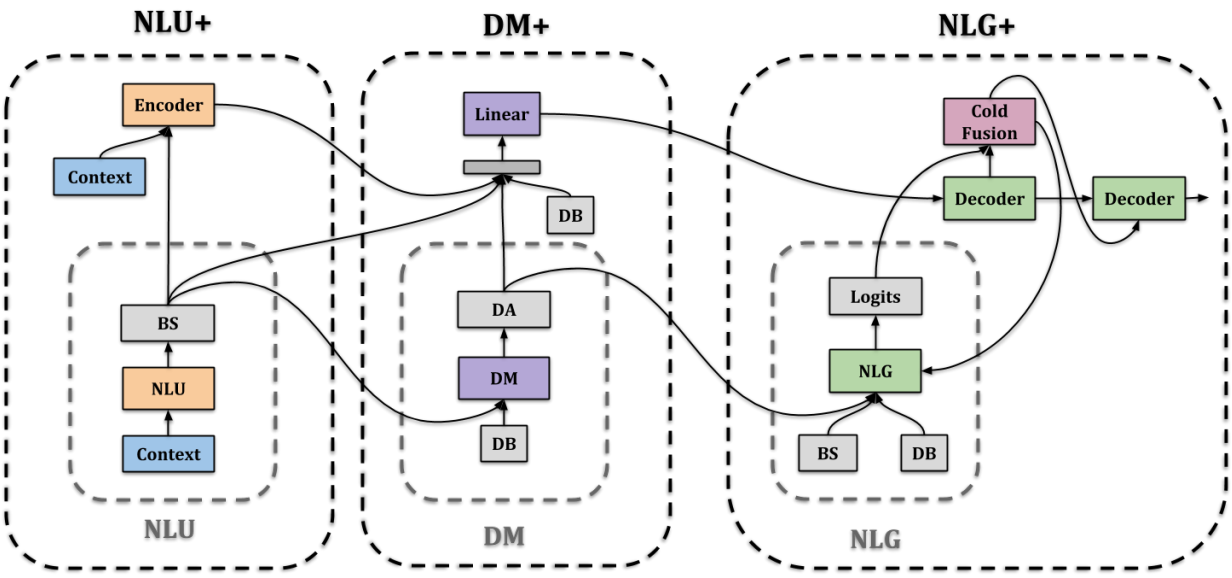


Figure 4.5: The Structured Fusion Network. The grey dashed boxes correspond to the pre-trained neural dialog modules. A higher-level is learned on top of the pre-trained modules, as a mechanism of enforcing structure in the end-to-end model.

is passed to the model in two places and the database vector in three places. This redundancy is necessary for two reasons. First, each of the neural modules must function independently and thus needs sufficient inputs. Second, the higher-level model should be able to function well independently. If any of the neural modules was to be removed, SFN should be able to perform reasonably. This means that the higher-level module should not rely on any of the neural modules to capture information about the input and therefore allow the neural modules to focus only on representing the structure. For example, if the context was not passed into the higher-level encoder and instead only to the NLU module, then the NLU may no longer be able to sufficiently model the belief state and may instead have to more explicitly model the context (e.g., as a bag-of-words representation).

Several variations of training SFN are considered: (1) The pre-trained neural modules are kept frozen, as a way of ensuring that the structure is not deteriorated. (2) The pre-trained neural modules are fine-tuned for the end-to-end task of response generation. This ensures that the model is able to abandon or modify certain elements of the structure if it helps with the end-to-end task. (3) The pre-trained modules are multi-tasked with the end-to-end task of response generation. This ensures that the structure is maintained and potentially strengthened while also allowing the modules to update and improve for the end-to-end task.

Reinforcement Learning with SFNs

A motivation for incorporating explicit structure into neural models is that it may reduce the effects of the implicit language model, and therefore mitigate degenerate output after reinforcement learning. This hypothesis is evaluated by fine-tuning SFN with reinforcement learning. The setup for this experiment is

Model	BLEU	Inform	Success	Combined Score
Supervised Learning				
Seq2Seq [Budzianowski et al., 2018]	18.80	71.29%	60.29%	84.59
Seq2Seq w/ Attn [Budzianowski et al., 2018]	18.90	71.33%	60.96%	85.05
Seq2Seq (Ours)	20.78	61.40%	54.50%	78.73
Seq2Seq w/ Attn (ours)	20.36	66.50%	59.50%	83.36
3-layer HDSA [Chen et al., 2019b]	**23.60**	**82.90%**	**68.90%**	**99.50**
Naïve Fusion (Zero-Shot)	7.55	70.30%	36.10%	60.75
Naïve Fusion (Fine-tuned Modules)	16.39	66.50%	59.50%	83.36
Multitasking	17.51	71.50%	57.30%	81.91
Structured Fusion (Frozen Modules)	17.53	65.80%	51.30%	76.08
Structured Fusion (Fine-tuned Modules)	18.51	77.30%	64.30%	89.31
Structured Fusion (Multitasked Modules)	16.70	80.40%	63.60%	88.71
Reinforcement Learning				
Seq2Seq + RL [Zhao et al., 2019]	1.40	80.50%	**79.07%**	81.19
LiteAttnCat + RL [Zhao et al., 2019]	12.80	**82.78%**	**79.20%**	**93.79**
Structured Fusion (Frozen Modules) + RL	**16.34**	**82.70%**	72.10%	**93.74**

Table 4.1: Experimental results for the various models. This table compares two classes of methods: those trained with supervised learning and those trained with reinforcement learning. All bold-face results are statistically significant ($p < 0.01$).

similar to that of Zhao et al. [2019]: (1) the model produces a response conditioned on a ground-truth dialog context, (2) the success rate is evaluated for the generated response, (3) using the success rate as the reward, the policy gradient is calculated at each word, and (4) the parameters of the model are updated. A learning rate of `1e-5` is used with the Adam optimizer [Kingma and Ba, 2015].

Reinforcement learning is used to fine-tune the best performing model trained in a supervised learning setting. During this fine-tuning, the neural dialog modules (i.e., the NLU, DM and NLG) are frozen. Only the high-level model is updated during reinforcement learning. Freezing maintains the structure, while still updating the higher level components. Since the structure is maintained, it is unnecessary to alternate between supervised and reinforcement learning.

4.2.2 Experiments

Dataset

The dialog systems are evaluated on the MultiWOZ corpus [Budzianowski et al., 2018], which consists of ten thousand human-human conversations covering several domains. The MultiWOZ corpus contains conversations between a tourist and a clerk at an information center which fall into one of seven domains - attraction, hospital, police, hotel, restaurant, taxi, train. Individual conversations span one to five of the domains. Dialogs were collected using the Wizard-of-Oz framework, where one participant plays the role of an automated system.

Each dialog consists of a goal and multiple user and system utterances. Each turn is annotated with two binary vectors: a belief state vector and a dialog act vector. A single turn may have multiple positive values in both the belief state and dialog act vectors. The belief state and dialog act vectors are of dimensions 94 and 593, respectively.

Several metrics are used to evaluate the models. BLEU [Papineni et al., 2002] is used to compute the word overlap between the generated output and the reference response. Two task-specific metrics, defined by Budzianowski et al. [2018], Inform rate and Success rate, are also used. Inform rate measures how often the system has provided the appropriate entities to the user. Success rate measures how often the system answers all the requested attributes. Similarly to Budzianowski et al. [2018], the best model is selected during validation using the combined score which is defined as $BLEU + 0.5 \times (Inform + Success)$. This combined score is also reported as an evaluation metric.

Experimental Setup

The hyperparameters match those used by Budzianowski et al. [2018]: embedding dimension of 50, hidden dimension of 150, and a single-layer LSTM. All models are trained for 20 epochs using the Adam optimizer Kingma and Ba [2014], with a learning rate of 0.005 and batch size of 64. The norm of the gradients are clipped to 5 Pascanu et al. [2012]. Greedy decoding is used during inference.

All previous work uses the ground-truth belief state vector during training and evaluation. Therefore the experiments with the SFNs have the NLU module replaced by an "oracle NLU" which always outputs the ground-truth belief state.

Results

Experimental results in Table 4.1 show that Structured Fusion Networks obtain strong results when compared to both methods trained with and without the use of reinforcement learning. Compared to previous methods trained only with supervised learning, SFN obtains a **+4.26** point improvement over seq2seq baselines in the combined score, with strong improvement in both Success and Inform rates. SFN is outperformed by the HDSA [Chen et al., 2019b] model which relies on BERT [Devlin et al., 2018] and conditioning on graph structured dialog acts. When using reinforcement learning, SFN matches the performance of LiteAttnCat [Zhao et al., 2019] on the combined score. Though the Inform rate is equivalent and the Success rate is lower (albeit still better than all supervised methods), the BLEU score of SFN is much better with an improvement of **+3.54** BLEU over LiteAttnCat.

In the reinforcement learning setting, the improved BLEU can be attributed to the explicit structure of the model. This structure enables the model to optimize for the reward (Success rate) without resulting in degenerate output [Lewis et al., 2017].

SFN obtains the highest combined score when the modules are fine-tuned. This is likely because, while the structured modules serve as a strong initialization for the task of dialog generation, forcing the model to maintain the exact structure (i.e., frozen modules) limits its ability to learn. In fact, the end-to-end model

may choose to ignore some elements of intermediate structure (e.g., a particular dialog act) which prove useless for the task of response generation.

Despite strong overall performance, SFN does exhibit a **-2.27** BLEU drop when compared to the strongest seq2seq baseline and a **-5.09** BLEU drop compared to HDSA. Though it is difficult to ascertain the root cause of this drop, one potential reason could be that the dataset contains many social niceties and generic statements (e.g., *"happy anniversary"*) which are difficult for a structured model to effectively generate (since it is not an element of the structure) while a free-form sequence-to-sequence network would not have this issue.

To a lesser degree, multi-tasking (i.e., multitasked modules) would also prevent the model from being able to ignore some elements of the structure. However, SFN with multitasked modules performs best on the Inform metric with a **+9.07%** improvement over the seq2seq baselines and a **+3.10%** over other SFN-based methods. This may be because the Inform metric measures how many of the requested attributes were answered, which benefits from a structured representation of the input.

Zero-Shot Naïve Fusion performs very poorly, suggesting that the individual components have difficulty producing good results when given imperfect input. Though the NLG module performs extremely well when given the oracle dialog acts (28.97 BLEU; 106.02 combined), its performance deteriorates significantly when given the predicted dialog acts. This observation is also applicable to Structured Fusion with frozen modules.

HDSA [Chen et al., 2019b] outperforms SFN possibly due to the use of a more sophisticated Transformer model [Vaswani et al., 2017] and BERT pre-training [Devlin et al., 2018]. A unique advantage of SFN is that the architecture of the *neural dialog modules* is flexible. The performance of HDSA could potentially be integrated with SFN by using the HDSA model as the NLG module of an SFN.

These strong performance gains reaffirm the hypothesis that adding explicit structure to neural dialog systems results in improved modelling ability particularly with respect to dialog policy as seen in the increase in Inform rate and in Success rate. The results with reinforcement learning suggest that the explicit structure allows *controlled fine-tuning* of the models, which prevents divergent behavior and degenerate output.

Human Evaluation

To supplement the results in Table 4.1, human evaluation was used to compare seq2seq, SFN, SFN fine-tuned with reinforcement learning, and the ground-truth human response. Workers on Amazon Mechanical Turk (AMT) were asked to read the context, and score the *appropriateness* of each response on a five-point scale (1-5). One hundred context-response pairs were labeled by three workers each. The results shown in Table 6.8 demonstrate that SFNs with RL outperform the other methods in terms of human judgment. These results indicate that in addition to improving on automated metrics, SFN results in user-favored responses.

Model	Avg Rating	≥ 4	≥ 5
Seq2Seq	3.00	40.21%	9.61%
SFN	3.02	**44.84%**	11.03%
SFN + RL	**3.12**	**44.84%**	**16.01%**
Human	3.76	59.79%	34.88%

Table 4.2: Results of human evaluation experiments. The ≥ 4 and ≥ 5 columns indicate the percentage of system outputs which obtained a greater than 4 and 5 rating, respectively.

Example Responses

Table 4.3 shows several examples of dialogs from the test set of MultiWOZ, along with the produced response from three different models: sequence-to-sequence networks, Structured Fusion Networks, and Structured Fusion Networks fine-tuned with reinforcement learning. These examples serve to provide insight into the respective strengths and weaknesses of the different models. A few noteworthy observations from the four examples are enumerated below:

(1) SFN fine-tuned with RL **consistently provides more attribute information.** It provides at least one attribute in every example response, for a total of 14 total attributes across the four examples. This, along with the high Success score of this model, is a consequence of the reinforcement learning fine-tuning which directly optimizes the Success score and rewards the model for providing the correct attributes.

(2) Seq2Seq **produces more generic responses**. In the second and fourth examples, the outputs produced by Seq2Seq are generic and are unrelated to the context. This suggests that the Seq2Seq model has an overwhelming implicit language model, which produces generic and dull responses. On the other hand, the explicit structure in SFN mitigates the effect of the implicit language model by relying on an *explicit* language model (in the form of the NLG).

(3) Seq2Seq **requests attributes which the user has already provided.** In the first and third example, the Seq2Seq output is requesting parameters that the user explicitly provided. This highlights the lack of structured representations in the Seq2Seq model. SFN which explicitly models the structure of the input effectively understands and captures the information provided in the context.

4.2.3 Analysis

Limited Data

Structured Fusion Networks should outperform sequence-to-sequence (seq2seq) networks in reduced data scenarios due to the explicit structure. While a baseline method would require large amounts of data to learn to infer structure, SFN does this explicitly.

Example 1
Dialog Context: USER: hello . i am looking for a [value_pricerange] place to eat in the [value_area] . can you help me ?
Sequence-to-Sequence Response: i have [value_count] options for you . do you have a preference on price range ?
Structured Fusion Network Response: [restaurant_name] is a [value_pricerange] -ly priced [value_food] restaurant in the [value_area] . would you like to make a reservation ?
Structured Fusion Network w/ Reinforcement Learning Response: [restaurant_name] is a [value_pricerange] [value_food] restaurant in the [value_area] . it is located at [restaurant_address] . the postcode is [restaurant_postcode] . the phone number is [restaurant_phone]
Example 2
Dialog Context: . . . USER: i want to book this hotel for the same group of people and [value_count] nights starting from [value_day] . i'll need a reference number please . SYSTEM: your reservation is confirmed and the reference number is ## USER: i am sorry ? were you able to book the express by holiday inn for [value_count] people ?
Sequence-to-Sequence Response: you are welcome . have a great day !
Structured Fusion Network Response: you are all set . your reference number is [hotel_reference] . can i help you with anything else ?
Structured Fusion Network w/ Reinforcement Learning Response: your booking was successful . the reference number is [hotel_reference] .
Example 3
Dialog Context: USER: can you see if there are any entertainment attractions in the [value_area] of town ?
Sequence-to-Sequence Response: there are [value_count] museums in the [value_area] . do you have a preference for area ?
Structured Fusion Network Response: there are [value_count] museums in the [value_area] . [attraction_name] is a museum and the [attraction_name] is a [value_count] .
Structured Fusion Network w/ Reinforcement Learning Response: [attraction_name] is a museum in the [value_area] and the [attraction_name] is located at [attraction_address] . the postcode is [attraction_postcode] .
Example 4
Dialog Context: USER: i need a taxi . i'll be departing from [restaurant_name] . SYSTEM: i can help you with that . when do you need to leave ? USER: i would like to leave after [value_time] please . SYSTEM: where will you be going ? USER: i'll be going to [restaurant_name] .
Sequence-to-Sequence Response: you are welcome . have a good time !
Structured Fusion Network Response: you are welcome . have a great day !
Structured Fusion Network w/ Reinforcement Learning Response: i have booked for you a [taxi_type] , the contact number is [taxi_phone] . is there anything else i can help you with ?

Table 4.3: Four examples of dialog contexts from the dataset, and the responses generated by three different models: Seq2Seq, SFN, and SFN with RL.

The performance of seq2seq and SFN are evaluated when training on 1%, 5%, 10% and 25% of the training data (total size of $\sim$ 55,000 utterances). The supervised-learning variant of SFN with fine-tuned modules is used. The pre-training of the modules and fine-tuning of the full model is done on the same data split. The full data is used during validation and testing.

The results in Figure 4.6 show the Inform and Success rates for different amounts of training data. SFN significantly outperforms the seq2seq model in low-data scenarios. Notably, improvement is considerably higher in the most extreme low-data scenario, when only 1% of the training data ($\sim$ 550 dialogs) is used. As the amount of training data increases, the gap between the two models stabilizes. The effectiveness at extreme low-data scenarios reaffirms the hypothesis that explicit structure makes SFN less data-hungry than sequence-to-sequence networks.

Domain Generalizability

The explicit structure of SFN should facilitate effective domain generalization. A domain transfer experiment was performed to compare the ability of seq2seq and SFN to generalize to unseen inputs. The models were both trained on a reduced dataset that largely consists of out-of-domain examples and evaluated on in-domain examples. Specifically, 2000 out-of-domain training examples and only 50 in-domain training examples were used. The restaurant domain of MultiWOZ was selected as in-domain.

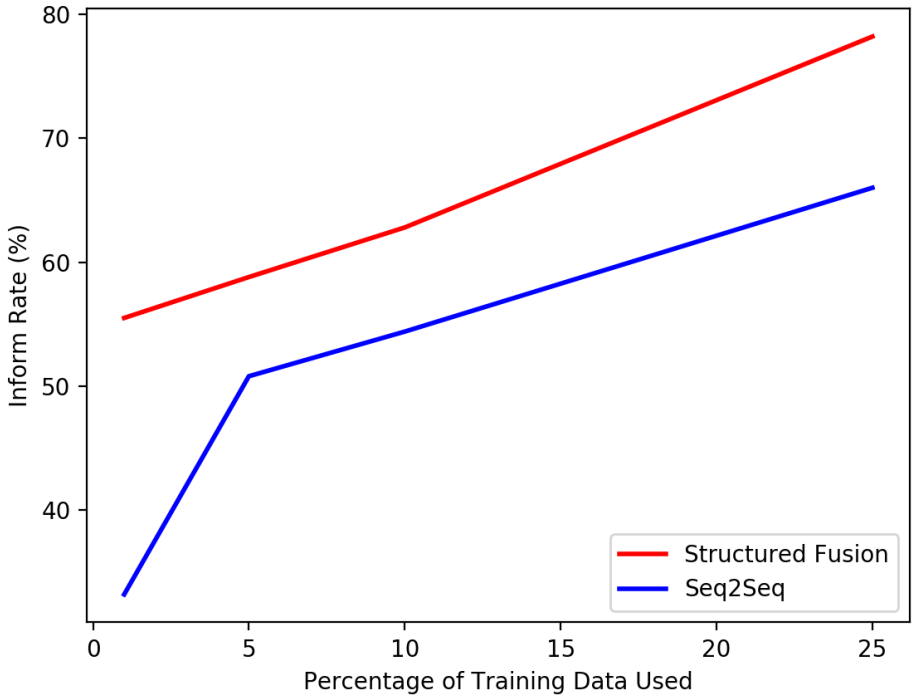
Model	BLEU	Inform	Success
Seq2Seq	10.22	35.65%	1.30%
SFN	7.44	**47.17**%	**2.17**%

Table 4.4: Results of the domain transfer experiment comparing sequence-to-sequence and Structured Fusion Networks. All bold-face results are statistically significant ($p < 0.01$).

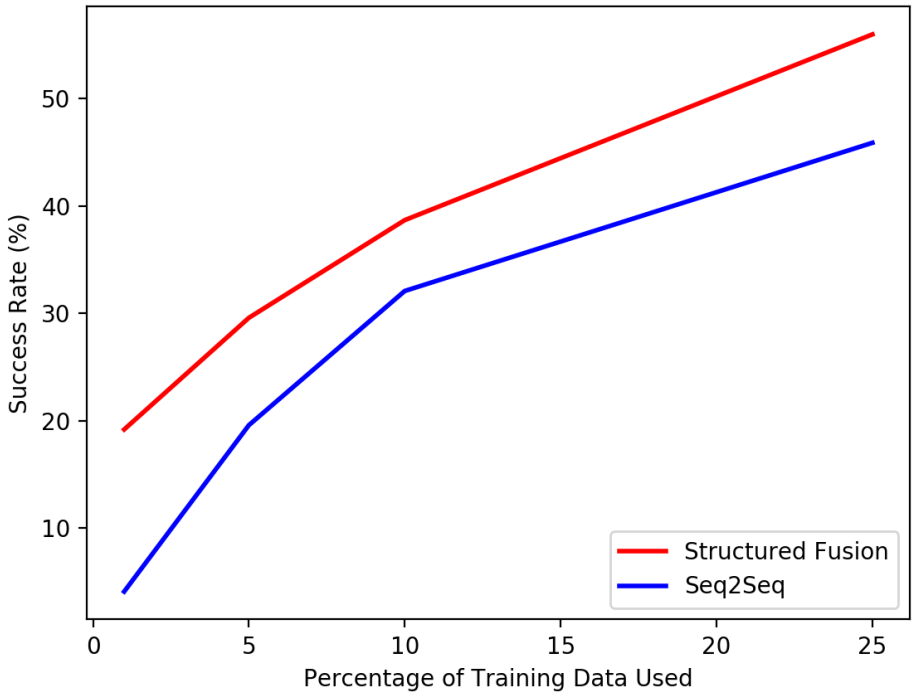
The results, seen on Table 4.4, show that SFN performs significantly better on both the Inform (**+11.52%**) and Success rate. Although SFN has a slightly higher Success rate, both models perform poorly. This is expected since the models would be unable to answer all the requested attributes when they have seen little domain data – their language model would not be tuned to the in-domain task. The **-2.78** BLEU reduction roughly matches the BLEU difference observed on the main task, therefore it is not an issue specific to domain transfer.

4.2.4 Discussion

This work studies several methods of incorporating explicit structure into end-to-end neural models of dialog. The structure of pipeline dialog systems is incorporated into the model architecture of neural dialog systems, as an inductive bias. This inductive bias *prescribes* a specific process by which Structured Fusion Networks should produce an output from a given input. Structured Fusion Networks, comprised of pre-trained *dialog modules* and a higher-level end-to-end network, are shown to obtain strong results on the MultiWOZ dataset both with and without the use of reinforcement learning. SFN is further shown to be ro-



(a)



(b)

Figure 4.6: Variation of Inform (a) and Success (b) rate at different amounts of training data.

bust to divergence during reinforcement learning, effective in few-shot settings and exhibits improvements on domain generalization.

4.3 Example-Driven Intent Prediction

Given the vast space of potential domains in task-oriented dialog, a key challenge of dialog systems research is to effectively and efficiently adapt to new domains [Rastogi et al., 2019]. Rather than adapting to new domains by relying on large amounts of domain-specific data, a scalable paradigm for adaptation necessitates the development of generalizable models that perform well in few-shot settings [Casanueva et al., 2020; Mehri et al., 2020a]. This work incorporates inductive biases into the model architecture in order to facilitate generalization to *new inputs* and *new outputs*, specifically for the task of intent prediction.

A universal goal of language encoders is that inputs with similar semantic meanings have similar latent representations [Devlin et al., 2018]. To maintain consistency with this goal, this work explores **example-driven training** wherein an utterance is classified by measuring similarity to a set of examples corresponding to each intent class. While standard approaches implicitly capture the latent space to intent class mapping in the learned weights (i.e., through a classification layer), example-driven training makes the prediction step an explicit non-parametric process that reasons over a set of examples. By maintaining consistency with the universal goal of language encoders and explicitly reasoning over the examples, example-driven training demonstrates improved generalizability to unseen intents and domains. Example-driven training is an inductive bias in the model architecture, that is motivated by knowledge of the desired generalization. By comparing to a set of examples in a non-parametric manner, it is possible to generalize to *new outputs* (i.e., new intents) without any additional training.

By leveraging example-driven training, in combination with CONVBERT (described in Section 3.1) and observers (described in detail in Mehri et al. [2020b]), this work attains state-of-the-art results on three intent prediction datasets: BANKING77 [Casanueva et al., 2020], CLINC150 [Larson et al., 2019], and HWU64 [Liu et al., 2019a] in both full data and few-shot settings. To measure the generalizability of the proposed models, experiments are carried out evaluating the ability of the proposed models to transfer to new intents and across datasets. By simply modifying the set of examples during evaluation and without any additional training, the example-driven approach attains strong results on both transfer to unseen intents and across datasets – thereby demonstrating generalization to *new inputs* and to *new outputs*.

4.3.1 Methods

Baselines

Several baseline models are leveraged for the task of intent prediction. The simplest baseline is an off-the-shelf BERT-base model [Devlin et al., 2018], fine-tuned for the task of intent prediction. Next, a CONVBERT model with task-adaptive self-supervised training (as described in Section 3.1) is fine-tuned. The final baseline model is a CONVBERT model that leverages *observers* [Mehri et al., 2020b]. Observers

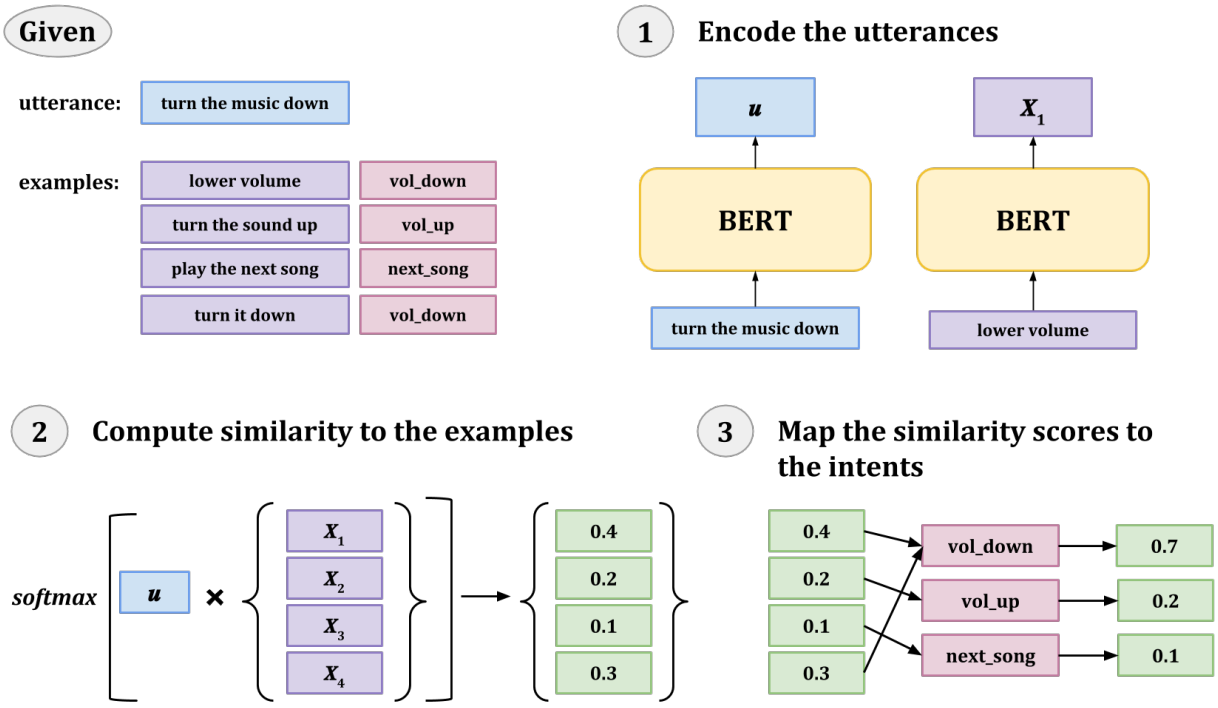


Figure 4.7: A visualization of the three step process of computing a probability distribution over the set of intents in the example-driven formulation.

improve the representational power of BERT-like models by mitigating the negative effects of the *[CLS]* token. Observers were motivated by an analysis of BERT's attention [Clark et al., 2019; Kovaleva et al., 2019] and are shown to better capture the semantic information of the input utterance [Mehri et al., 2020b]. The superior representations produced by observers are particularly conductive for example-driven training, which relies on meaningful representations to measure the similarity between utterances.

Example-Driven Training

A universal goal of language encoders is that inputs with similar semantic meanings should have similar latent representations. BERT [Devlin et al., 2018] has been shown to effectively identify similar sentences [Reimers and Gurevych, 2019], even without additional fine-tuning [Zhang et al., 2019b]. Example-driven training aims to reformulate the task of intent prediction to be more consistent with this universal goal of language encoders.

A BERT-like encoder is used to train an intent classification model to (1) measure the similarity of an utterance to a set of examples and (2) infer the intent of the utterance based on the similarity to the examples corresponding to each intent. Rather than implicitly capturing the *latent space* to *intent class* mapping in the learned weights (i.e., through a classification layer), this approach makes the mapping an explicit non-parametric process that reasons over a set of examples. Our formulation, similar to metric-based meta

learning [Koch et al., 2015], only performs gradient updates for the language encoder, which is trained for the task of *sentence similarity*. This example-formulation, serves as an inductive bias that *prescribes* a specific process for intent prediction. It is hypothesized that the resulting model will better generalize in few-shot scenarios, as well as to rare intents.

The set of variables used to define the model architecture are: (1) a language encoder $\mathcal{F}$ that encodes an utterance to produce a latent representation, (2) a natural language utterance utt, and (3) a set of n examples $\{(\mathrm{x}_1, y_1), \ldots, (\mathrm{x}_n, y_n)\}$ where $\mathrm{x}_{1,\ldots,n}$ are utterances and $y_{1,\ldots,n}$ are their corresponding intent labels. With $\mathcal{F}$ being a BERT-like model, the following equations describe example-driven intent classification:

$$\boldsymbol{u} = \mathcal{F}(utt) \tag{4.11}$$

$$\boldsymbol{X}_i = \mathcal{F}(\mathrm{x}_i) \tag{4.12}$$

$$\boldsymbol{\alpha} = \text{softmax}(tinky\boldsymbol{u}^T \cdot \boldsymbol{X}) \tag{4.13}$$

$$P(c) = \sum_{i:\, y_i = c} \boldsymbol{\alpha}_i \tag{4.14}$$

These equations describe a non-parametric process for intent prediction. Instead, through the example-driven formulation (visualized in Figure 4.7), the underlying language encoder (e.g., BERT) is being trained for the task of sentence similarity. A universal goal of language encoders is that inputs with similar semantic meaning should have similar latent representations. By formulating intent prediction as a sentence similarity task, this approach is adapting BERT-based encoders in a way that is consistent with this universal goal. In contrast to the baseline models, this formulation facilitates generalizability and has the potential to better transfer to new intents and domains.

At training time, the set of examples is populated in a two step process: (i) for each intent class that exists in the training batch, a *different* utterance of the same intent class is sampled from the *training set* and (ii) utterances are randomly sampled from the training set until the set of examples is double the size of the training batch size (128 example utterances). During inference, the example set includes all the utterances in the training data.

4.3.2 Experiments

Datasets

The aforementioned models are evaluated on three intent prediction datasets: BANKING77 [Casanueva et al., 2020], CLINC150 [Larson et al., 2019], and HWU64 [Liu et al., 2019a]. These datasets span several domains and consist of many different intents, making them more challenging and more reflective of commercial settings than commonly used intent prediction datasets like SNIPs [Coucke et al., 2018]. BANKING77 contains 13,083 utterances related to banking with 77 different fine-grained intents. CLINC150 contains 23,700 utterances spanning 10 domains (e.g., travel, kitchen/dining, utility, small talk, etc.) and 150 different

intent classes. HWU64 includes 25,716 utterances for 64 intents spanning 21 domains (e.g., alarm, music, IoT, news, etc.).

Casanueva et al. [2020] forego a validation set for these datasets and instead only use a training and testing set. In contrast, these experiments follow the setup of Mehri et al. [2020a] (i.e., the set up in Section 3.1), wherein a portion of the training set is designated as the validation set.

Experimental Setup

Evaluation is carried out in two experimental settings following prior work [Casanueva et al., 2020; Mehri et al., 2020a]: (1) using the full training set and (2) using 10 examples per intent or approximately 10% of the training data. In both settings, the models are evaluated on the validation set at the end of each epoch and early stopping is performed with a patience of 20 epochs for a maximum of 100 epochs. Since the few-shot experiments are more sensitive to initialization and hyperparameters, the experiments are repeated 5 times and the final result is the average over the experimental runs. For the few-shot settings, the models use *only* the few-shot training data for both masked language modelling and as examples at inference time in the example-driven models (i.e., they do not see any additional data).

Results

The experimental results, as well as the results presented by Casanueva et al. [2020] and Mehri et al. [2020a] are shown in Table 4.5. In combination with observers, example-driven training results in (1) SoTA results across the three datasets and (2) a significant improvement over the BERT-base model, especially in the few-shot setting (**+5.02%** on average).

Furthermore, the results show that the use of observers is particularly conducive to the example-driven training setup. Combining these two approaches gains strong improvements over the ConvBERT + MLM model (few-shot: **+4.98%**, full data: **+0.41%**). However, when considered independently, there is no consistent improvement for both example-driven (few-shot: **-0.46%** full data: **+0.24%**) and observers (few-shot: **+0%**, full data: **-0.42%**). The fact that these two methods are particularly conductive to each other signifies the importance of using them jointly. The representation step of intent prediction is tackled by observers, which aim to better capture the semantics of an input by disentangling the attention and therefore avoiding the dilution of the representations. The prediction step, is improved through example-driven training which uses the underlying BERT-based model to predict intents by explicitly reasoning over a set of examples. This characterization highlights the importance of jointly addressing both steps of the process simultaneously. Using observers alone does not lead to significant improvements because the linear classification layer cannot effectively leverage the improved representations. Using example-driven training alone does not lead to significant improvements because the *[CLS]* representations do not capture enough of the underlying utterance semantics. The enhanced semantic representation of observers is necessary for example-driven training: by improving the latent representations of utterances, it is easier to measure similarity in the set of examples.

Model	BANKING77		CLINC150		HWU64	
	Few	**Full**	**Few**	**Full**	**Few**	**Full**
Prior Work						
USE* [Casanueva et al., 2020]	84.23	92.81	90.85	95.06	83.75	91.25
CONVERT* [Casanueva et al., 2020]	83.32	93.01	92.62	97.16	82.65	91.24
USE+CONVERT* [Casanueva et al., 2020]	85.19	93.36	93.26	97.16	85.83	92.62
BERT-BASE [Mehri et al., 2020a]	79.87	93.02	89.52	95.93	81.69	89.97
CONVBERT [Mehri et al., 2020a]	83.63	92.95	92.10	97.07	83.77	90.43
CONVBERT + MLM [Mehri et al., 2020a]	83.99	93.44	92.75	97.11	84.52	92.38
Proposed Models						
CONVBERT + MLM + *Example*	84.09	**94.06**	92.35	97.11	83.44	92.47
CONVBERT + MLM + *Observers*	83.73	92.83	92.47	96.76	85.06	92.10
CONVBERT + MLM + *Example* + *Observers*	**85.95**	93.83	**93.97**	**97.31**	**86.28**	**93.03**

Table 4.5: Accuracy scores ($\times 100\%$) on all three intent detection data sets with varying number of training examples (**Few:** 10 training utterances per intent; **Full:** full training data). The full data results of Casanueva et al. [2020] are trained on more data as they forego a validation set. These experiments follow the setup of Mehri et al. [2020a], wherein a portion of the training set is used as the validation set. Results in bold-face are statistically significant by t-test ($p < 0.01$).

Model	BANKING77	CLINC150	HWU64
BERT-BASE (OFF-THE-SHELF)	19.50	26.50	26.56
CONVBERT (OFF-THE-SHELF)	19.50	26.50	26.56
CONVBERT + MLM + *Example*	67.36	79.69	62.24
CONVBERT + MLM + *Example* + *Observers*	**84.87**	**94.35**	**85.32**
BEST FULLY TRAINED MODEL	85.95	93.97	86.28

Table 4.6: Accuracy scores ($\times 100\%$) for transferring to unseen intents averaged over 30 runs wherein 4-10 intents are removed from the few-shot setting during training and added back in during evaluation. The last row corresponds to the best results that were trained with all of the intents, shown in Table 4.5. Note that the non example-driven models are incapable of predicting unseen slots, and their perform is equivalent to random chance.

Transfer to Unseen Intents

By formulating intent prediction as a sentence similarity task, the example-driven formulation allows for the potential to predict intents that are unseen at training time. Experiments are carried out in the few-shot setting for each dataset, by (1) randomly removing 4 - 10 intent classes when training in an example-driven manner, (2) adding the removed intents back to the set of examples during evaluation and (3) reporting

Model	BANKING77	CLINC150	HWU64
TRAINED ON BANKING77	*93.83*	91.26	83.64
TRAINED ON CLINC150	**85.84**	*97.31*	**86.25**
TRAINED ON HWU64	77.95	**92.47**	*93.03*

Table 4.7: Accuracy scores ($\times$100%) for transferring across datasets (in the full data setting) using the ConvBERT + MLM + Example + Observers model. The diagonal consists of results where the model was trained and evaluated on the same dataset.

results only on the unseen intents. This process is repeated 30 times for each dataset and the results are reported in Table 4.6. It should be noted that task-adaptive self-supervised training is not performed on the utterances corresponding to the unseen intents.

These results demonstrate that the example-driven formulation generalizes to *new intents*, without having to re-train the model. The performance on the unseen intents approximately matches the performance of the best model which has seen all intents (denoted BEST FULLY TRAINED MODEL in Table 4.6). These results highlight a valuable property of the proposed formulation: namely, that new intent classes can be added in an online manner without having to re-train the model. While the off-the-shelf BERT-base and CONVBERT models, which are not at all fine-tuned on the datasets, are able to identify similar sentences to some extent – training in an example-driven manner drastically improves performance.

The addition of observers, in combination with example-driven training, significantly improves performance on this experimental setting (**+18.42%**). This suggests that the observers generalize better to unseen intents, potentially because the observers are better able to emphasize words that are key to differentiating between intents (e.g., *turn the volume **up*** vs *turn the volume **down***).

Transfer Across Datasets

While transferring to unseen intents is a valuable property, the unseen intents in this experimental setting are still from the same domain. To further evaluate the generalizability of our models, experiments are carried out evaluating the ability of models to transfer to *other datasets*. Using the full data setting with 10 training utterances per intent: (1) a model is trained on a dataset and (2) the trained models are evaluated on a new dataset, using the training set of the new dataset as examples during inference. This evaluates the ability of the models to transfer to unseen intents and domains without additional training.

The results in Table 4.7 demonstrate the ability of the the model with observers and example-driven training to transfer to new datasets, which consist of both unseen intents and unseen domains. These results show that the example-driven model performs reasonably well even when transferring to domains and intents that were not seen at training time. These results, in combination with the results shown in Table 4.6 speak to the generalizability that is induced through example-driven training. Specifically, by formulating intent prediction as a sentence similarity task through example-driven training, the models maintain consistency

Model	BANKING77	CLINC150	HWU64
CONVBERT + MLM + *Example*	34.22	31.92	19.73
CONVBERT + MLM + *Example* + *Observers*	**35.34**	**33.84**	**21.19**

Table 4.8: Micro-averaged F-1 scores for the task of reproducing the words of the input (using only the most frequent 1000 words) given the different latent representations.

with a universal goal of language encoders (i.e., that utterances with similar semantic meanings have similar latent representations) that effectively transfers to new settings.

Examples

Table 4.9 shows examples of intent predictions on the HWU corpus using both observers and example-driven. These examples show that semantically similar example utterances are identified, particularly when using observers. Furthermore, these examples show that explicitly reasoning over examples makes intent classification models more interpretable.

4.3.3 Discussion

Example-driven training *prescribes* a specific process by which the data-driven model goes from utterance to intent. The prescribed process is motivated, in particular, by knowledge of the desired generalization. In order to facilitate transfer to unseen intents, example-driven training reformulates intent prediction to be an entirely non-parametric process. The design of this non-parametric process (i.e., comparing to examples) serves as an inductive bias that facilitates generalization to new outputs.

Furthermore, example-driven training maintains consistency with a universal goal of language encoders: i.e., that similar sentences have similar representations. By maintaining consistency with this universal goal, as well as relying on large-scale pre-trained language models, the resulting model is able to generalize to new inputs and domains very efficiently. This is particularly demonstrated by the result of the dataset transfer experiments, wherein the models are shown to be effective in entirely new domains.

4.4 Conclusion

This chapter demonstrates that inductive biases in the model architecture can force a data-driven model to learn specific high-level abstractions which are conductive to generalization. The first body of work introduced in this chapter is Structured Fusion Networks, which prescribe a specific procedure by which a model must perform the task of response generation. By incorporating the structure of pipeline dialog systems in the model architecture, SFN relies on inductive bias to faciltate better performance in low-resource settings and in few-shot domain adaptation. The second study introduces example-driven training, which makes the *representation to output* mapping an explicit non-parametric process and thereby facilitates generalization

Utterance: It is too loud. Decrease the volume
Intent: `audio-volume-down`
Model: CONVBERT + MLM + *Example*
Predicted Intent: `audio-volume-up`
Nearest Examples:
Make sound louder (`audio-volume-up`)
Your volume is too high, please repeat that lower (`audio-volume-down`)
Too loud (`audio-volume-down`)
Can you speak a little louder (`audio-volume-up`)
Model: CONVBERT + MLM + *Example* + *Observers*
Predicted Intent: `audio-volume-down`
Nearest Examples:
It's really loud can you please turn the music down (`audio-volume-down`)
Up the volume the sound is too low (`audio-volume-up`)
Too loud (`audio-volume-down`)
Decrease the volume to ten (`audio-volume-down`)
Utterance: Please tell me about the historic facts about India
Intent: `qa-factoid`
Model: CONVBERT + MLM + *Example*
Predicted Intent: `general-quirky`
Nearest Examples:
How has your life been changed by me (`general-quirky`)
Is country better today or ten years ago? (`general-quirky`)
What happened to Charlie Chaplin? (`general-quirky`)
How does production and population affect us? (`general-quirky`)
Model: CONVBERT + MLM + *Example* + *Observers*
Predicted Intent: `qa-factoid`
Nearest Examples:
Tell me about Alexander the Great (`qa-factoid`)
Give me a geographic fact about Vilnius (`qa-factoid`)
Tell me about Donald Trump (`qa-factoid`)
I want to know more about the upcoming commonwealth games (`qa-factoid`)

Table 4.9: Examples of predictions on the HWU corpus with both observers and example-driven training.

both to new inputs and to new outputs. Like SFNs, the work in example-driven training prescribes a specific procedure by which the model must infer the output, which is shown to be conductive to generalization, particularly to new intents. Through these two studies, this chapter validates the use inductive biases in the model architecture as a means of enforcing specific abstractions and behaviors, thereby facilitating the desired generalizations.

Chapter 5

Inductive Bias in the Problem Formulation

5.1 Introduction

The formulation of a problem strongly influences the abstractions learned by the resulting data-driven model. As described in the previous chapter, inductive biases in the model architecture are motivated by knowledge of the problem and the desired generalizations. Similarly, inductive biases in the problem formulation are motivated by *knowledge of the model's capabilities* and the *desired generalizations*. Understanding and effectively leveraging the properties and pre-existing capabilities of the model is particularly imperative when working with large-scale pre-trained models. Such models have various capabilities that can be leveraged to facilitate generalization. Particularly for zero-shot generalization, it is important to achieve a strong alignment between the pre-existing capabilities of a pre-trained model and the requirements of a downstream problem.

This chapter studies mechanisms of incorporating inductive biases into the problem formulation, in order to effectively leverage the capabilities of pre-trained language models and consequently induce generalization. By modifying the data representation, training algorithm and inference algorithm – the resulting models are *prescribed* a specific process by which they must predict the output. The focus of this chapter is to incorporate inductive biases that prescribe a process which is well-aligned with the pre-existing capabilities of the model. For example, if a pre-trained model was trained with a ranking objective, it would likely be more effective to reformulate a classification/generation problem as a ranking problem. Likewise, large-scale pre-trained models (e.g., DialoGPT [Zhang et al., 2019c]) may have learned certain skills in the process of learning to produce human-level responses. These skills may include implicit notions of dialog quality and slot filling. As such, by reformulating downstream problems, it may be possible to leverage these implicitly learned skills in order to induce zero-shot and few-shot generalization.

In Section 5.2 the problem of dialog evaluation is reformulated to leverage the strong response generation capabilities of DialoGPT [Zhang et al., 2019c]. This reformulation is motivated by the hypothesis that DialoGPT, in the process of learning to produce human-level responses, has learned to capture an implicit notion of dialog quality. The FED metric uses DialoGPT to calculate the likelihood of various *follow-*

up utterances in order to evaluate eighteen different qualities of dialog (e.g., relevant, fluent, topic depth, engaging, etc.) in a zero-shot manner. By incorporating this inductive bias in the problem formulation, the FED metric is shown to exhibit zero-shot generalization to new outputs (i.e., qualities). In Section 5.3, the problem of slot-filling is reformulated to, again, leverage the capabilities of DialoGPT. GenSF reformulates the problem of slot-filling as a response completion task, wherein DialoGPT is tasked with completing partial responses (e.g., "Ok the `slot_name` is"). In combination with constrained decoding strategies, GenSF is shown to strongly outperform prior work in few-shot and zero-shot settings, thereby demonstrating generalization to new inputs and new outputs (i.e., slots).

5.2 Reformulating Dialog Evaluation

The problem of dialog evaluation is challenging for several reasons: (1) The one-to-many nature of dialog [Zhao et al., 2017b] makes word-overlap metrics ineffective for scoring valid responses that deviate from the ground-truth [Liu et al., 2016; Gupta et al., 2019]. (2) Dialog quality is inherently multi-faceted [Walker et al., 1997; See et al., 2019] and an interpretable metric should measure several qualities (e.g., *interesting*, *relevant*, *fluent*). (3) Dialog systems have begun to be evaluated in an interactive setting [Ram et al., 2018; Adiwardana et al., 2020] wherein a real user has a back-and-forth conversation with a system. Interactive evaluation is not constrained to a static corpus and better captures the performance of a system in a realistic setting. Measuring multiple distinct qualities of dialog, without a reference response and in an interactive setting, in the absence of annotated training data is a problem of generalization. This work introduces the **FED** metric (fine-grained evaluation of dialog) which assesses eighteen qualities of dialog in a zero-shot manner without relying on a reference response [Mehri and Eskenazi, 2020a].

In order to facilitate zero-shot generalization to multiple different outputs (i.e., dialog qualities), the problem of dialog evaluation is *reformulated* to better align with the capabilities of pre-trained language models. Through large-scale self-supervised pre-training, DialoGPT [Zhang et al., 2019c] is able to generate practically human-level responses. Kocijan et al. [2019] assert that pre-trained models implicitly capture world knowledge and can therefore perform common-sense reasoning. Similarly, this work hypothesizes that DialoGPT has implicitly captured some notion of dialog quality and can therefore be used for dialog evaluation. By understanding the capabilities of the pre-trained DialoGPT (i.e., superior response generation and implicit notion of dialog quality), the problem of dialog evaluation can be reformulated to leverage these capabilities and achieve the desired zero-shot generalization. The problem reformulation serves as an inductive bias that is motivated by knowledge of the capabilities of the pre-trained model.

Assessing the quality of a system utterance in an interactive setting by looking at the *following user response* has been shown to produce better human evaluations [Eskenazi et al., 2019]. The FED metric is based on the same intuition. Given a system response, its quality is measured by computing the likelihood that DialoGPT will respond to it with a particular follow-up utterance (e.g., *"That is really interesting!"*). DialoGPT is more likely to respond in this way to what it believes is an *interesting* system response. A set of follow-up utterances is constructed for each of the eighteen qualities and the likelihoods of these follow-

up utterances are used to measure dialog quality. The FED metric is shown to obtain moderate to strong correlation with human judgement for turn-level and dialog-level evaluation without any training data or ground-truth response.

5.2.1 Methods

The FED (fine-grained evaluation of dialog) metric is an automatic evaluation metric for dialog which (1) does not need to compare to a reference response, (2) measures eighteen fine-grained qualities of dialog, and (3) does not use training data. Capturing a diverse set of fine-grained qualities without supervision is an especially challenging problem. To achieve zero-shot generalization to various dialog qualities, the FED metric reformulates the problem of dialog evaluation to achieve better alignment with the capabilities of pre-trained language models. The reformulation of dialog evaluation is motivated by two areas of prior work: (1) pre-trained language models and their capabilities and (2) the use of follow-up utterances as a mechanism for evaluation.

DialoGPT

Zhang et al. [2019c] extend GPT-2 [Radford et al., 2018] to train DialoGPT on 147M conversation-like interactions from Reddit. DialoGPT is shown to outperform humans at producing relevant, interesting and human-like responses.

Kocijan et al. [2019] show that pre-trained language models, specifically BERT [Devlin et al., 2018], implicitly capture world knowledge and can therefore perform common sense reasoning. By calculating which answer results in a more probable sentence according to BERT, they strongly outperform other methods on the Winograd Schema Challenge [Levesque et al., 2012].

Just as BERT has been shown to capture world knowledge, this work hypothesizes that DialoGPT has implicitly captured some notion of dialog quality. The qualities of a particular dialog context (e.g., *interesting*, *relevant*, *informative*) likely inform DialoGPT's response and, as such, must be captured by the model. If there was training data for the eighteen dialog qualities, this hypothesis could be verified by fine-tuning DialoGPT for the task of dialog evaluation. Without training data, however, the challenge is to devise an unsupervised mechanism for extracting the quality information implicitly captured by DialoGPT.

Follow-Up Utterance for Evaluation

Eskenazi et al. [2019] assess the quality of a system utterance in an interactive setting, by looking at the *following user response*. When users speak to a system, their response to a given system utterance may implicitly or explicitly provide feedback for the system. For example, if a user follows up a system utterance with *"That's not very interesting"*, they are providing information about the quality of the system utterance.

The conversations in the FED dataset were collected in an interactive setting. Thus the use of the follow-up utterance is a valid option. However, even if users consistently provided feedback, it would be difficult to interpret the feedback without training data.

FED Metric

The proposed FED metric is motivated by (1) the intuition that DialoGPT has implicitly learned to reveal dialog quality and (2) that the follow-up utterance can provide valuable information about a system response. By leveraging an understanding of the capabilities of the pre-trained model, the problem of dialog evaluation can be reformulated to achieve the desired zero-shot generalizations. To measure the quality of a system response s, FED computes the likelihood of the model generating various follow-up utterances (e.g., *"Wow! Very interesting."*) in response to s. DialoGPT will be more likely to respond with a positive follow-up utterance if given a better (e.g., more *interesting/relevant/fluent*) preceding system utterance.

For each of the eighteen fine-grained dialog qualities, a set of positive follow-up utterances, p, and a set of negative follow-up utterances, n, is constructed. Specifically, given a dialog context c, a system response r and a function $\mathcal{D}$ that computes the log-likelihood of DialoGPT generating a particular response, the predicted score for a dialog quality is calculated as:

$$\sum_{i=1}^{|p|} \mathcal{D}(c+r, p_i) - \sum_{i=1}^{|n|} \mathcal{D}(c+r, n_i) \tag{5.1}$$

This equation can be modified to predict scores for dialog-level qualities, by simply removing the system response r from the equation.

A response is said to be *interesting* if it is more likely that DialoGPT (acting as the user) responds with a positive follow-up utterance (e.g., *"Wow! Very interesting"*) than with a negative one (e.g., *"That's really boring"*). For each of the eighteen qualities, several positive and negative utterances were hand-written[1] and minimally tuned on a small subset of the dataset (10 conversations).

Generally, negative follow-up utterances are more meaningful than positive ones. For example, if a system response is *irrelevant*, a follow-up utterance of *"That's not relevant"* is reasonable. However, acknowledging the relevance of a system response is less likely. Therefore the log-likelihood produced by DialoGPT will be noisier and less informative. The number of positive utterances for each dialog quality ranges between 0 and 4, and the number of negative utterances ranges between 1 and 4. The overall impression scores are calculated by taking an average of the scores for either the turn-level or dialog-level qualities.

5.2.2 Experiments

Experiments are carried out to measure the correlation of the FED metric with human judgments. At the time these experiments were carried out, there were no evaluation metrics that could (1) operate without a reference response and (2) effectively assess interactive dialog (i.e., not grounded in a specific corpus). As such, the experiments described in this section do not compare to other metrics. Since then, Yeh et al. [2021] has performed a more exhaustive comparison between FED and more recent metrics.

[1]The hand-written follow-up utterances can be found at `https://github.com/shikib/fed`.

Dataset

The FED corpus, described in detail in Mehri and Eskenazi [2020a], consists of a total of 124 conversations (40 Meena, 44 Mitsuku, 40 Human) each annotated by five different workers. Each conversation had one dialog-level annotation and three turn-level annotations for chosen system responses that were randomly sampled from the conversation. There were 9 questions for turn-level annotation and 11 for dialog-level annotation. In total, the FED dataset includes 3348 turn-level and 1364 dialog-level data points, for a total of 4712. This dataset intended to be used solely for the evaluation of metrics, as the number of annotated conversations is not large enough to accommodate both training and testing.

Since dialog quality is inherently multi-faceted it is important to measure several different qualities of dialog. The FED corpus consists of eighteen fine-grained dialog qualities: eight at the turn level and ten at the dialog level. The full set of qualities are described in Mehri and Eskenazi [2020a].

Experimental Setup

The FED metric was evaluated using four variations of the pre-trained DialoGPT model. The pre-trained DialoGPT models can be either medium size: 345M or large: 762M. They are either fine-tuned from GPT-2 [Radford et al., 2018] or trained from scratch. The follow-up utterances were handwritten and minimally tuned on 10 conversations using the 762M fine-tuned model. The small (117M) DialoGPT model was not used since Zhang et al. [2019c] demonstrated its poor performance.

Most of the turn-level qualities were scored using only the last system response as context. For *relevant*, *correct* and dialog-level metrics, the entire conversation was used as context.

Correlation with Human Judgement

The Spearman correlation was measured between the predicted quality scores and the mean of the annotated scores. Correlations for all the dialog qualities, and all four variations of the underlying DialoGPT model are shown in Table 5.1. The best overall turn-level correlation is **0.209** and the best overall dialog-level correlation is **0.443**. Despite being entirely zero-shot and not relying on a reference response, FED achieves correlations which are competitive with prior work on dialog evaluation. Multi-reference evaluation for dialog achieves correlations in the 0.10 - 0.27 range [Gupta et al., 2019] and ADEM demonstrates correlations in the 0.28 - 0.42 range [Lowe et al., 2017].

The FED metric works better for some dialog qualities than others. This is because DialoGPT was trained on Reddit and is therefore more likely that it has captured certain dialog qualities that Reddit exhibits. For example, it is more likely that DialoGPT learns to measure qualities like *interesting* and *engaging*, than *understandable* and *consistent*. In the Reddit training data, the former two qualities show more variation than the latter. For example, there are interesting and un-interesting utterances, however most utterances on Reddit are generally understandable. The former two qualities are also more likely to influence the system response. Conversely, the latter two qualities are unlikely to be acknowledged in the response. For example, since Reddit is a multi-participant forum and not a one-on-one conversation, inconsistencies in conversation

Quality	345M fs	345M ft	762M fs	762M ft
Turn-Level				
Interesting	0.388	**0.431**	0.406	0.408
Engaging	0.268	0.285	0.278	**0.318**
Specific	0.260	**0.326**	0.270	0.267
Relevant	*0.028*	*-0.027*	*0.001*	**0.152**
Correct	*0.000*	*0.037*	*0.020*	**0.133**
Semantically Appropriate	*0.040*	**0.177**	0.141	0.155
Understandable	*0.047*	*0.048*	*0.075*	**0.111**
Fluent	0.157	0.184	0.133	**0.224**
Overall	0.122	*0.092*	*0.094*	**0.209**
Dialog-Level				
Coherent	0.195	*0.151*	*0.149*	**0.251**
Error Recovery	*0.165*	*0.128*	*0.126*	*0.165*
Consistent	*0.041*	*0.011*	*0.006*	*0.116*
Diverse	**0.449**	0.431	0.414	0.420
Topic Depth	**0.522**	0.479	0.470	0.476
Likeable	*0.047*	*0.172*	0.224	**0.262**
Understanding	0.237	0.174	0.192	**0.306**
Flexible	0.260	**0.408**	0.298	0.293
Informative	0.264	0.328	**0.337**	0.288
Inquisitive	*0.137*	*0.143*	**0.298**	0.163
Overall	0.401	0.359	0.355	**0.443**

Table 5.1: Spearman correlations with human judgement. All values that are not statistically significant ($p > 0.05$) are italicized. The highest correlation for each quality is shown in bold.

history are unlikely to be reflected in the response. As such, it is unsurprising that this approach struggles to measure the consistency of a dialog.

An optimal generation model (e.g., a human) should exhibit compositionality and be capable of producing utterances that have never been observed. For example, even if *'That is not consistent'* has never appeared in the training data, a compositional model would be capable of generating it. This difference in performance across the different dialog qualities suggests that DialoGPT exhibits some degree of compositionality, as evidenced by its ability to compose some follow-up utterances which are not frequently observed in the Reddit data (e.g., *'You really don't know much?'*), however it still struggles with follow-up utterances consisting of less frequently observed concepts (e.g., *consistent*, *understandable*).

5.2.3 Discussion

The FED metric is shown to effectively measure **eighteen fine-grained qualities of dialog** without any supervision and without comparing to a reference response. This type of zero-shot generalization to unseen outputs (i.e., dialog qualities) is facilitated by incorporating inductive biases in the problem formulation. Motivated by knowledge of the capabilities of pre-trained language models, FED reformulates the problem of dialog evaluation in order to leverage the notion of dialog quality that was implicitly captured by DialoGPT. This reformulation serves as an inductive bias that facilitates zero-shot generalization to new outputs. While inductive biases in the model architecture incorporate are motivated by knowledge of the problem, inductive biases in the problem formulation are conversely motivated by knowledge of the pre-trained model's capabilities. The problem reformulation serves as an inductive bias that achieves better alignment between the *capabalities of DialoGPT* and the *requirements of the problem of dialog evaluation*.

5.3 Reformulating Slot Filling

Recent work has validated the idea that stronger alignment between pre-trained models and the downstream problem formulation results in improved performance. Rather than fine-tuning off-the-shelf models, it is more effective to first understand the downstream problem and adapt the model's architecture, pre-training and inference algorithm accordingly. Adapting pre-trained models in this manner is equivalent to **incorporating inductive biases about the downstream problem** into the model architecture, as described in the previous chapter. For example, pre-training on open-domain dialog data results improves performance on downstream dialog tasks [Henderson et al., 2019; Mehri et al., 2020a]. Designing task-specific pre-training objectives has yielded strong results in extractive question answering [Glass et al., 2019], paraphrase and translation [Lewis et al., 2020] and slot filling [Henderson and Vulić, 2020]. This body of work attains stronger alignment by significantly modifying the pre-trained model through task-specific pre-training. However, this approach necessitates a new pre-trained model for every downstream problem, and therefore relinquishes the inherent scalability of the transfer learning paradigm. In this work, stronger alignment is instead achieved by *simultaneously adapting* both the pre-trained model and the downstream problem

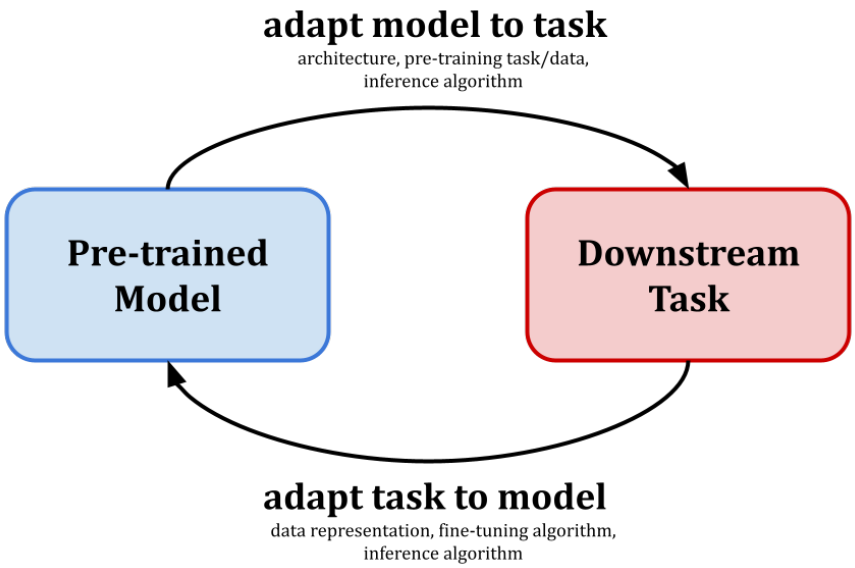


Figure 5.1: To achieve a stronger alignment, both the downstream problem and the pre-trained models must be adapted. The downstream problem can be adapted with knowledge of the properties and capabilities of the pre-trained models. Likewise, the pre-trained model can be adapted with knowledge of the downstream problem/data.

formulation, such that both contain inductive biases about the other.

The downstream problem formulation can be adapted to achieve stronger alignment with the capabilities of the pre-trained model. To effectively leverage pre-trained models, it is important to first understand the properties and capabilities of the model derived from the model architecture, the pre-training data and task. Then the downstream problem can be adapted to be better aligned with the model. Adapting the task formulation to the model is equivalent to **incorporating inductive biases about the pre-trained model** into the downstream problem. By simultaneously adapting both the downstream problem and the pre-trained model, it is possible to achieve stronger alignment without sacrificing the inherent scalability of the transfer learning paradigm (i.e., avoiding task-specific pre-trained models).

This work addresses the task of slot filling, a natural language understanding task with the goal of identifying values for pre-defined attributes (slots) in a natural language utterance. DialoGPT [Zhang et al., 2019c], a generative language model pre-trained on open-domain dialog data, is leveraged for the task of slot filling. To achieve strong alignment between the slot filling task and DialoGPT, this work proposes to (1) reformulate slot filling as a natural language response generation task, and (2) augment the DialoGPT architecture with a copy-mechanism, constrained decoding and a post-processing heuristic. The resulting model, GENSF (**Gen**erative **S**lot **F**illing), is shown to achieve state-of-the-art results on two slot filling datasets. GENSF achieves the strongest performance gains in few-shot and zero-shot settings, highlighting the importance of stronger alignment as a mechanism of inducing generalization both to new inputs and new outputs.

5.3.1 Methods

In order to effectively leverage a pre-trained generative dialog model, DialoGPT [Zhang et al., 2019c], for the task of slot-filling, this work introduces the GENSF model which achieves stronger alignment between the downstream problem and the pre-trained model, by simultaneously adapting the task to the model and the model to the task. First, the slot filling task is reformulated as a natural language response generation task to be better aligned with the DialoGPT model. Next, several modifications are made to the DialoGPT architecture and inference algorithm that act as inductive biases for the slot filling task.

Slot Filling as Response Generation

Given an utterance $u = \{w_1, w_2, \dots w_n\}$, a set of possible slot keys $s = \{s_1, s_2, \dots s_k\}$, and a list of slots requested by the system $r = \{r_1, r_2, \dots r_m\}$ (where $r_i \in s$ and $m \geq 0$), the task of slot filling is to assign a value to a subset of the slot keys. Concretely, for a given slot key s_i, the output will either be `NULL` or a contiguous span of words from the utterance: $s_i = \{w_i, \dots w_{i+j}\}$.

In response generation, given a dialog context consisting of a sequence of utterances: $c = \{x_1, x_2, \dots x_n\}$ wherein each utterance x_i is a sequence of words, the task is to generate a valid response $y = \{w_1, w_2, \dots w_m\}$.

Many tasks can be represented as an *input to output* mapping [Raffel et al., 2019; Hosseini-Asl et al., 2020; Peng et al., 2020a], making sequence-to-sequence a universal formulation. Trivially, slot filling can be represented as a sequence-to-sequence task by setting the context to be the concatenation of the utterance and the requested slots: $c = \{u, r\}$ and the target response to be the slot mappings $y = \{(s_1, w_{i:j}), (s_2, \texttt{NULL}), \dots (s_k, (w_{j:n})\}$. However, this does not leverage the natural language capabilities of pre-trained dialog models. While this trivial formulation may suffice with sufficient training, it will under-perform in few-shot and zero-shot settings. To this end, this work presents a reformulation of slot filling that better aligns with the natural language capabilities of DialoGPT.

We hypothesize that to some degree, large-scale dialog pre-training can result in a model implicitly learning to fill slots. For example, given the slot key *'time'*, such a model should understand what *time* is and should be able to generate a valid time (e.g., *'4:15 pm'*). An effective problem formulation can leverage these implicitly learned slot filling capabilities. An off-the-shelf pre-trained model is likely to only be capable of filling generic slots (e.g., time, date, price, etc.). But by reformulating slot filling in a manner that is better aligned with the pre-training task, it should be easier for the model to adapt to novel slot keys.

Concretely, given a slot filling input (u, r) and a particular slot key s_i, a natural language dialog context is constructed using a template-based approach: $c =$ *'What is the* $\{f(r)\}$*?* `[eos]` $\{u\}$ `[eos]` *Ok, the* $\{f(s_i)\}$ *is'*. Here, f denotes a manually constructed function that maps slot keys to a natural language phrase (e.g., $first\_name$: *first name*, $departure\_location$: *leaving from*). Given the constructed dialog context, the model is tasked with completing the partial response (i.e., *Ok, the* $\{f(s_i)\}$ *is*) by auto-regressively generating the slot value. During training the model would be tasked with generating either the slot value or the phrase *not provided*. With this natural language reformulation, the slot filling task is being adapted to better leverage the capabilities of the pre-trained DialoGPT model. As this achieves better alignment

Utterance	Requested Slots	Slot Key	Natural Language Context
We will require an outside table to seat 9 people on August 23rd	None	`date`	We will require an outside table to seat 9 people on August 23rd `[EOS]` Ok, the date is
Laurice Hoisl	`first_name`, `last_name`	`first_name`	What is the first name, last name? `[EOS]` Laurice Hoisl `[EOS]` Ok, the first name is
My party will be 9 people. My name is Nancie Waltemeyer and the time is 7pm	None	`people`	My party will be 9 people. My name is Nancie Waltemeyer and the time is 7pm `[EOS]` Ok, the number of people is

Table 5.2: Examples of slot filling inputs reformulated as natural language dialog contexts

between the pre-trained model and the downstream problem, it should be more effective for zero-shot and few-slot filling. To better illustrate the conversion of the slot-filling input (utterance u and request slots r), several examples are shown in Table 5.2.

DialoGPT for Slot Filling

In order to adapt the pre-trained DialoGPT model to the slot filling task, the architecture is augmented and the inference algorithm is modified. These adaptations are motivated by the observation that if the slot value is provided, it will always be a contiguous span of tokens from the utterance. As such, the generative model can only produce: (1) *'not provided'* if the slot does not appear in the utterance, (2) the end of sentence token, and (3) tokens from the input utterance.

A copy-mechanism is incorporated into the DialoGPT architecture to allow the model to explicitly generate tokens from the input utterance. Given a context $c = \{x_1, x_2, \ldots x_n\}$, through its self-attention layers, the model will produce a hidden state representation for each token, $h = \{h_1, h_2, \ldots, h_n\}$. A probability distribution over the vocabulary is then obtained by passing h_n through a classification layer:

$$P_{vocab} = \text{softmax}(Wh_n + b) \tag{5.2}$$

To explicitly generate tokens from the input, h_n is used to attend to $h_{1:n}$ to produce a probability distribution over $x_{1:n}$. The process for computing the probability for a specific word, $P_{copy}(w)$ is as follows:

$$\alpha = \text{softmax}(h_n^T h_{1:n}) \tag{5.3}$$

$$P_{copy}(w) = \sum_{i:x_i=w} \alpha_i \tag{5.4}$$

These two probability distributions are combined through a weighted sum. The weight assigned to each

of the distributions is predicted using h_n:

$$p_{copy} = \sigma(W_{copy} h_n + b_{copy}) \tag{5.5}$$

The final probability distribution is therefore:

$$P_{final} = (1 - p_{copy}) P_{vocab} + p_{copy} P_{copy} \tag{5.6}$$

The copy-mechanism requires training, as it introduces new weights (w_{copy}, b_{copy}) and the off-the-shelf DialoGPT model does not necessarily produce meaningful attention weights, α, that can be used to create an output probability distribution. As such, to attain strong zero-shot performance, the inference algorithm is also modified to account for the aforementioned observation. This is done using both constrained decoding and a post-processing heuristic.

Constrained decoding is a modification of greedy decoding wherein the argmax sampling is modified to only generate (1) words that appear in the input utterance, (2) the end of sentence token and (3) the phrase *'not provided'*.

The slot values may consist of terms that the model has not frequently observed during pre-training (e.g., names, times). As such, because the DialoGPT model leverages a subword vocabulary, some subword tokens may be dropped during generation and therefore the slot values may be generated with typos (e.g., 'Mocer' vs 'Mocher'). A simple post-processing heuristic is applied to mitigate this problem. If the slot value produced by the model is not present in the utterance, the Levenshtein distance to every contiguous span of tokens in the utterance is computed. If the best edit distance is within a certain threshold ($0.3 \times \text{len}(y)$), the corresponding span is returned as the slot value.

Through these modifications, the DialoGPT model is adapted to reflect the properties of the slot filling task. The copy-mechanism, constrained decoding and post-processing mechanism serve as an inductive bias to enable the pre-trained model to be better adapted for the downstream slot filling task.

5.3.2 Experiments

Experiments are performed to empirically validate the hypothesis that simultaneously adapting the downstream problem and the pre-trained model results in stronger alignment and improved performance. Experiments are presented on two datasets which assess GENSF in full-data, few-shot and zero-shot settings. An ablation study is performed to characterize the source of the performance gains and demonstrate the importance of simultaneous adaptation.

Datasets

Experiments are carried out on RESTAURANTS-8K [Coope et al., 2020] and the DSTC8 datasets [Rastogi et al., 2020a]. RESTAURANTS-8K consists of 8,198 utterances from a commercial restaurant booking system and includes 5 slots (date, time, people, first name, last name). The DSTC8 datasets span four different

Fraction	Span-ConveRT	Span-BERT	ConVEx	GenSF
1 (8198)	95.8	93.1	96.0	**96.1**
$1/2$ (4099)	94.1	91.4	94.1	**94.3**
$1/4$ (2049)	91.2	88.0	92.6	**93.2**
$1/8$ (1024)	88.5	85.3	90.6	**91.8**
$1/16$ (512)	81.1	76.6	86.4	**89.7**
$1/32$ (256)	63.8	53.6	81.8	**82.1**
$1/64$ (128)	57.6	42.2	**76.0**	**76.1**
$1/128$ (64)	40.5	30.6	71.7	**72.2**

Table 5.3: F_1 scores across all slots for the evaluation on the RESTAURANTS-8K test data with varying proportions of the training set. Numbers in brackets denote the training set sizes. The best scores (statistically significant by t-test to $p < 0.05$) are shown in boldface.

domains (buses, events, homes, rental cars) for a total of 5,569 utterances with slot annotations extracted by Coope et al. [2020].

In both datasets, the value for a particular slot is always a contiguous span of the utterance. Some utterances consist of a set of slots requested by the system prior to the user utterance. This allows an otherwise ambiguous utterance like *'four'* to be interpreted as either *'four people'* or *'four o'clock'*.

Experimental Setup

The experiments use the pre-processing and evaluation scripts provided by the DialoGLUE benchmark [Mehri et al., 2020a]. The setup of Coope et al. [2020] and Henderson and Vulić [2020] is followed, wherein a validation set is not used and the experiments are therefore performed with fixed hyperparameters. Throughout all the experiments, the medium version of DialoGPT [Zhang et al., 2019c] is used. The AdamW optimizer [Loshchilov and Hutter, 2017] is used with a learning rate of `5e-5`. On RESTAURANTS-8K, the models are trained for 10 epochs in the full-data setting, 20 epochs in the few-shot settings and 40 epochs in the extreme few-shot settings ($1/32$ - $1/128$; or less than 256 training examples). On the DSTC8 datasets, the models are trained for 20 epochs in the full-data setting and 40 epochs in the few-shot setting.

The models are evaluated on the full test set, regardless of the amount of training data, using macro-averaged F_1 score [Coope et al., 2020].

Slot Filling Results

The experiments compare GENSF to several models from prior work. Span-ConveRT [Coope et al., 2020] and Span-BERT train a CNN and a CRF on top of contextual subword embeddings produced by ConveRT [Henderson et al., 2019] and BERT [Devlin et al., 2018], respectively. ConVEx [Henderson and Vulić, 2020] devises a *pairwise cloze* pre-training objective specifically for slot-filling. This task-specific pre-training objective is an example of significantly adapting the pre-trained model to the downstream problem. In

	Setting	Span-ConveRT	Span-BERT	ConVEx	GenSF
Buses_1	Full-Data (1133)	93.5	93.3	96.0	**98.1**
	Few-Shot (283)	84.0	77.8	86.7	**90.5**
Events_1	Full-Data (1498)	92.7	84.3	91.7	**94.7**
	Few-Shot (374)	82.2	78.6	87.2	**91.2**
Homes_1	Full-Data (2064)	94.8	96.3	**98.3**	96.9
	Few-Shot (516)	**95.4**	**95.1**	94.5	93.7
RentalCars_1	Full-Data (874)	**94.0**	92.8	92.0	93.5
	Few-Shot (218)	83.0	81.4	**87.4**	86.7

Table 5.4: F_1 scores across all slots for evaluation on the DSTC8 single-domain datasets in the full-data and few-shot settings. Numbers in brackets denote training set sizes. The best scores (statistically significant by t-test, to $p < 0.05$) are shown in boldface.

contrast to ConVEx, GENSF achieves strong alignment between the pre-trained model and the downstream problem by simultaneously adapting both the problem and the model. As such, GENSF does not need a task-specific pre-trained model and is inherently more scalable. The ConVEx pre-training takes 8 hours to train on 12 GPUs, while GENSF takes less than four hours to train on a single GTX 1080TI.

As shown in Table 5.3, GENSF achieves state-of-the-art results across all experimental settings on the RESTAURANTS-8K dataset. In the full-data setting, GENSF slightly outperforms ConVEx. Though the performance gain is small, this result signifies that the proposed model can leverage an abundance of data. The value of strong alignment between the downstream problem and the pre-trained model is better exemplified in the few-shot settings. Especially in the extreme few-shot settings (i.e., $1/32$ - $1/128$ of the training set), GENSF strongly outperforms Span-ConveRT and Span-BERT, with greater than 30 F_1 score improvements. The few-shot performance of both ConVEx and GENSF in these few-shot settings underlies the value of effectively aligning the pre-trained model and the downstream problem. However, GENSF achieves this alignment by simultaneously incorporating inductive biases about the model into the problem rather than designing a complex pre-training objective. By incorporating inductive biases into both the problem formulation and the model, the proposed approach does not require task-specific pre-trained models and therefore preserves the inherent scalability of the transfer learning paradigm. Furthermore, GENSF attains moderate improvements over ConVEx, especially in the few-shot settings, with a 3 F_1 score improvement in the $1/16$th setting.

The results on the DSTC8 single-domain datasets is shown in Table 5.4. Here, the models are evaluated on both full-data and few-shot (25% of the training data) settings. On average, GENSF achieves strong performance improvements over prior work. In the full-data settings the best performance is observed on the buses and events domains, where GENSF achieves a 2.1 and 3.0 F_1 score improvement over ConVEx, respectively. In the few-shot settings, GENSF achieves a 4.0 F_1 score improvement over ConVEx on these

domains and a 6.5 and 9.0 point improvement over Span-ConveRT. These strong improvements, over both Span-ConveRT and ConVEx, highlight the value of strong alignment between the pre-trained model and the downstream problem, particularly in the few-shot experiments.

GENSF moderately underperforms on the homes and rental cars domains. On the homes domain, GENSF outperforms Span-ConveRT and Span-BERT but scores 1.4 points below ConVEx. Similarly, on the rental cars domain, GENSF outperforms ConVEx and Span-BERT, but is 0.5 points below Span-ConveRT. Though GENSF is still competitive in these domains, these results nonetheless highlight a weakness of the model. The use of a generative pre-trained dialog model, specifically DialoGPT [**?**], was motivated by the hypothesis that such models can implicitly learn to identify certain slots through response generation pre-training. This hypothesis is empirically validated through improved performance on RESTAURANTS-8K and the buses/events domains of DSTC8. GENSF relies on the pre-trained model having an implicit understanding of the slots. This implicit understanding results in strong performance on slots like *'time'* or *'first name'*, since such terms are likely to have been observed during pre-training. However, this is not the case for all slots and GENSF can underperform on slots that are ambiguous, ill-defined or are unlikely to have been observed during open-domain dialog pre-training. The homes domain consists of the slot, *'area'*, which has several definitions and is therefore challenging for the pre-trained model to understand and detect. The rental cars domain contains the slots *'pickup date'* and *'dropoff date'*. While the DialoGPT model has learned to detect a *'date'*, the distinction between these two slots is more nuanced and therefore may cause some amount of confusion. As such, while GENSF is competitive in these domains and is only outperformed by one of the three models, these domains demonstrate that there are limitations at present to leveraging a generative pre-trained model. However, it is possible that by further adapting the downstream problem to the pre-trained model, for example by renaming these slots (e.g., *'area'* may be renamed to *'city'*), the performance drops may be mitigated.

Overall, GENSF achieves impressive performance gains in both full-data and few-shot settings, underlying the value of achieving strong alignment between the pre-trained model and the downstream problem. Furthermore, GENSF achieves this alignment by simultaneously adapting both the problem and the model and without sacrificing the inherent scalability of the transfer learning paradigm or necessitating task-specific pre-training. In the RESTAURANTS-8K and the single-domain DSTC8 datasets, GenSF achieves state-of-the-art results and outperforms prior work. In few-shot settings, the proposed model achieves a 30 F_1 score improvement over Span-BERT and Span-ConveRT. On average, GenSF moderately outperforms ConVEx, with > 2.0 F_1 score improvements in the few-shot settings on RESTAURANT-8K, and both the full data and few-shot settings on two of the DSTC8 datasets. These experiments empirically validate (1) the importance of aligning the pre-trained model and the downstream problem by simultaneously incorporating inductive biases into both the problem and the model and (2) that through response generation pre-training, dialog models have implicitly learned to detect certain slots, which can be leveraged by effectively adapting the downstream problem.

Slot	Metric	Coach+TR	ConVEx	GenSF
First Name	P	1.7	2.3	**13.7**
	R	4.1	20.1	**36.1**
	F_1	2.5	4.1	**19.8**
Last Name	P	0	1.9	**10.6**
	R	0	16.2	**19.7**
	F_1	0	3.4	**13.8**
Date	P	10.2	2.2	**10.7**
	R	**34.8**	10.1	15.3
	F_1	**15.7**	3.6	12.6
Time	P	**47.4**	5.6	27.5
	R	27.9	23.6	**46.9**
	F_1	**35.1**	9.1	34.7
People	P	0	3.8	**14.5**
	R	0	13.9	**18.9**
	F_1	0	6.0	**16.4**
Average	F_1	10.7	5.2	**19.5**

Table 5.5: Zero-shot slot filling results on RESTAURANTS-8K. All models are evaluated on the test set without any training on the dataset.

Zero-shot Slot Filling

For zero-shot slot filling, there must be strong alignment between the pre-trained model and the downstream problem. Since the model is not fine-tuned on the downstream corpus, it is necessary to effectively align the formulation of the downstream problem to the capabilities of the model. As such, zero-shot experiments validate the inductive biases incorporated into the problem formulation, i.e., slot filling being reformulated as natural language response generation.

For these experiments, GENSF is compared to the published results of ConVEx [Henderson and Vulić, 2020]. Furthermore, a Coach+TR model [Liu et al., 2020] is run on the RESTAURANT-8K dataset. Note that while ConVEx and GENSF have only been trained on open-domain dialog, Coach+TR trains on adjacent task-oriented domains (i.e., SNIPS), meaning that the zero-shot performance is higher on slots that are domain agnostic.

The experiments use the RESTAURANTS-8K dataset. The copy-mechanism is removed from the model, as it adds additional weights to the model and therefore requires training. However, the constrained decoding and the post-processing heuristic of GENSF, allow the model to enforce that the slot values will always be a contiguous span from the input utterance. Table 5.5 demonstrates that GENSF significantly outperforms

Model	Full-Data	Few-Shot ($^1/_{16}$)	Zero-Shot
GenSF	**96.1**	**89.7**	**19.5**
Removing Model Adaptation			
– Copy-mechanism	95.6	87.8	**19.5**
– Constrained Decoding	95.4	89.5	0.5
– Post-processing	**96.1**	**89.7**	18.1
– All model adaptation	95.4	87.8	0.5
Removing Problem Adaptation			
– Natural Language Slot Names	95.3	86.6	12.2
– Natural Language Templates	94.8	88.5	0.0
– All Natural Language	95.5	88.9	0.0
Removing All Adaptation			
– All Adaptation	95.8	89.2	0.0

Table 5.6: Ablation experiments. We remove (1) adaptations to the model, (2) adaptations to the downstream problem and (3) all adaptations proposed in this work. The experiments are carried out on the full-data, few-shot ($^1/_{16}$th of the training set) and zero-shot settings of RESTAURANTS-8K.

prior work on zero-shot slot filling with a **14 $\mathbf{F_1}$ score improvement** over ConVEx and a **9 $\mathbf{F_1}$ score improvement** over Coach+TR. These results further validate the hypothesis that pre-trained dialog models have implicitly learned to detect slots and that this ability can be leveraged through the proposed problem reformulation.

Most noteworthy is the performance on the *'first name'* and *'last name'* slots. This suggests that, to some degree, DialoGPT [Zhang et al., 2019c] can disambiguate between a first name and a last name when provided simultaneously (e.g., *'my name is Lakesha Mocher'*). It should be noted that the macro-averaged F_1 score used to evaluate the models considers a slot value to be incorrect unless it exactly predicts the ground-truth slot value. In many cases, the GENSF model produces appropriate slot values that differ from the ground-truth, e.g., *'wednesday'* instead of *'next wednesday'*. It is possible that by incorporating additional inductive biases about the specific formulation of the slot values (e.g., slots should have maximal information) into the inference algorithm, the zero-shot performance can be further increased.

GENSF is shown to strongly outperform prior work on zero-shot slot filling. This impressive performance validates the proposed approach of simultaneously adapting both the downstream problem and the pre-trained model. Furthermore, zero-shot performance also confirms the hypothesis that pre-trained response generation models have implicitly learned to understand and detect slots, thereby highlighting the potential of leveraging generative pre-trained models for language understanding tasks. Future work should explore mechanisms for reformulating other downstream problems (e.g., intent prediction, dialog state tracking) in order to leverage generative pre-trained models. Furthermore, it is possible that these zero-shot results

could be further improved through two-stage pre-training (e.g., further pre-train with the *'pairwise cloze'* task).

Ablation

GENSF has been shown to outperform prior work in full-data, few-shot and zero-shot settings. To determine the source of the improvements, an ablation study is carried out. The ablation experiments remove the adaptations used in GENSF and evaluate on RESTAURANTS-8K across full-data, few-shot ($1/16$ of the training set) and zero-shot settings. Removing all the ablation, is equivalent to training a DialoGPT model from scratch on the problem, similar to the approach proposed by Madotto [2020].

As shown in Table 5.6, the various adaptations are vital to the strong performance of GENSF. Of the model adaptations, only the copy-mechanism is necessary in the full-data setting, since the model effectively learns to copy tokens from the input utterance and therefore does not need constrained decoding and post-processing. However, constrained decoding is necessary for the zero-shot settings, as the zero-shot model does not leverage a copy-mechanism. Problem adaptation, especially the use of natural language templates, is shown to be important across all of the experimental settings. This highlights the importance of formulating the downstream problem in a manner that can effectively leverage the capabilities of the pre-trained models.

The results of the ablation study further validate this work's primary hypothesis. Pre-trained models work better for downstream problems, when the problem and the model are effectively aligned. As shown in the results of the ablation study, removing this adaptation results in a performance decrease.

5.3.3 Discussion

GENSF is shown to perform well on the problem of slot filling in full data, few-shot and zero-shot settings. The strong performance of GENSF is facilitated by achieving strong alignment between the capabilities of the pre-trained model and the requirements of the downstream problem. This work incorporates inductive biases into the problem formulation, and facilitates generalization to new inputs and new outputs (i.e., unseen slots) in both few-shot and zero-shot settings. The reformulation of slot filling as a generation problem, is motivated by knowledge of the capabilities of DialoGPT – specifically the hypothesis that DialoGPT has implicitly learned the meaning of certain slots over the course of large-scale pre-training (e.g., DialoGPT knows what *time* is). The strong zero-shot performance of GENSF validates the notion that reformulating a problem can result in stronger alignment and thereby facilitate generalization. GENSF demonstrates that understanding the capabilities of a pre-trained model and consequently incorporating inductive biases into the problem formulation can induce generalization to new inputs and new outputs.

5.4 Conclusion

This chapter validates the efficacy of incorporating inductive biases into the problem formulation as a means of inducing generalization and effectively leveraging pre-existing capabilities of pre-trained models. The first study reformulates dialog evaluation as a response generation task, thereby leveraging the pre-existing capabilities of DialoGPT. The FED metric evaluates dialogs by measuring the likelihood of follow-up responses, demonstrating zero-shot generalization to new inputs and new outputs. The second study similarly reformulates slot filling as a response completion task. Similar to FED, the GENSF model achieves few-shot and zero-shot generalization through an inductive bias in the problem formulation which allows the model to leverage the pre-existing knowledge and abilities of DialoGPT. Inductive biases in the problem formulation can result in stronger alignment between the capabilities of a pre-trained model and the requirements of a downstream problem, thereby facilitating few-shot and zero-shot generalization.

Chapter 6

Task Specification as an Inductive Bias

6.1 Introduction

This thesis has discussed three forms of inductive biases thus far: (i) through self-supervised training (Chapter 3), (ii) inductive biases in the model architecture (Chapter 4) and (iii) inductive biases in the problem formulation (Chapter 5). These inductive biases are shown to induce generalization to new inputs (i.e., domain shift), to new outputs (i.e., new intents, dialog qualities and slots) and to new problems (i.e., one model for multiple downstream problems). The previous chapters demonstrate the efficacy of well-motivated inductive biases as a means of prescribing specific learned abstractions in data-driven models and thereby inducing generalization. A system developer armed with knowledge (i.e., of the domain, problem, pre-trained model, desired generalization, etc.) can induce generalization by modifying a component of a data-driven model. In an effort to facilitate the fourth class of generalization, i.e., *generalization to new tasks*, this chapter leverages and extends the inductive biases studies thus far.

Generalization to new tasks is the problem of flexibly adapting a model of dialog to an unseen task. Consider a system that has been trained to handle several different tasks (e.g., restaurant reservations, ride booking, weather, etc.). How can this dialog system be *extended to handle a new task* (e.g., hotel booking), without collecting additional data? Zero-shot adaptation to unseen tasks is a long-standing challenge in dialog that is central to the notion of generalization in dialog and is particularly desirable for practical applications. This class of generalization is the most challenging, as it necessitates that a model be capable of generalizing to new inputs (caused by domain shift), to new outputs (because new tasks may require new output classes), and furthermore to unseen and unknown *dialog policies*. A task-specific dialog policy *defines* a particular task, by specifying how the system should respond to different user intents. For example, the policy may indicate that after the user provides their name – the system should ask for their date of birth. Standard data-driven models of dialog implicitly learn the task-specific dialog policy from training examples. This inherently impedes zero-shot generalization, as such models do not and can not have any notion of the dialog policy for an *unseen* task.

In order to facilitate zero-shot transfer to unseen tasks, it is important to better understand the desired

generalization. Consider the following analogy: *Mary is a new employee at a customer support center. Her job requires her to interact with customers and handle a wide variety of tasks. What is the best way of training Mary to do her job?*

1. *Mary is instructed to read the logs of all of the previous interactions held by the other support agents.* This is analogous to **full-shot training**. Mary's human-level language understanding and reasoning abilities will allow her effectively learn the various tasks performed at the support center. However, if the support center needs to handle an additional task — Mary will be unable to do so without additional training examples. Furthermore, because she is used to learning from a vast number of examples, she will require many examples for the new task.

2. *Mary is instructed to read a few (e.g., less than ten) logs for each of the necessary tasks.* This is analogous to **few-shot training**. It will take Mary considerably less time and data to learn to complete the necessary tasks. If the support center needs to handle an additional task — Mary will still need new training examples, albeit less than the previous scenario.

3. *To handle the large number of new tasks that are being added on a daily basis, Mary is provided with a thorough specification (e.g., a flow-chart) of each task. Mary reads the logs of previous interactions, with the goal of understanding how to interpret and leverage the task specification.* This is analogous to using the **task-specification as an inductive bias**. By reading the interaction logs while attending to the task specification, Mary will learn to leverage the task specification to interact with a customer. If the support center needs to handle a new task — Mary will be able to quickly adapt by leveraging the new task specification without the need for additional examples.

Through this thought experiment, we understand what is necessary for inducing zero-shot generalization to unseen tasks. In order for Mary, armed with human-level language understanding and reasoning abilities, to adapt to a new task without any data — she must (1) learn to interact conditioned on a task specification and (2) be given the specification for the unseen task. As such, this thesis proposes to address zero-shot generalization to new tasks by using the *task specification as an inductive bias*. Since a data-driven model does not and can not have prior knowledge of an unseen task (i.e., the policy and ontology), the task specification (or schema) is fundamentally *necessary* for zero-shot generalization. Through the three different studies in this chapter, we aim to demonstrate that using the task specification as an inductive bias is a *sufficient* mechanism for inducing zero-shot generalization.

The task specification aims to define a particular task through a generalized representation of the task-specific dialog policy and dialog ontology (i.e., the set of intent and slot classes). This chapter first presents the task specification which is designed to be a *minimal expression* of the necessary properties for a particular task. Next, three studies are carried out to demonstrate the efficacy of the task specification as an inductive bias: (1) task specification as an input, (2) task specification for synthetic data, and (3) task specification in the model architecture. Collectively, the work in this chapter aims to explicitly use the task specification as

an inductive bias in order to disentangle task-specific and task-agnostic abstractions in a data-driven model of dialog, thereby facilitating zero-shot generalization to unseen tasks.

6.2 Task Specification

The task specification must define a particular task. In full-shot or few-shot settings, the training corpus can be thought of as a task specification wherein the task-specific properties (i.e., policy, ontology) are conveyed through examples. Since the goal of this chapter is to facilitate zero-shot generalization, we aim to define a task specification that is a *minimal expression*[1] of the task-specific properties. Throughout this chapter, **zero-shot** refers to a setting wherein the only *human-annotated* data is the task specification. Since a task specification is a minimal expression of the necessary task-specific properties, we argue that it is impossible to use less data, without making assumptions about the prior knowledge of a pre-trained model. Such assumptions would limit the generality of a method for zero-shot generalization.

The next sub-sections consider three dialog problems (intent prediction, slot filling and next action prediction). For each of these problems, we (1) define the necessary task-specific properties that a data-driven model must learn and (2) define a task specification that can effectively convey these properties. The task specification must be (1) general and domain-agnostic, to facilitate effective description of a wide variety of tasks, (2) sufficiently expressive, to avoid making assumptions about an underlying data-driven model, and (3) human interpretable, to allow efficient annotation and modification by system developers.

6.2.1 Task-Specific Properties

To effectively construct a data-driven model, particularly in zero-shot settings, it is imperative to define *what the model must learn*. This section aims to conceptualize the task-specific properties and high-level abstractions that must be learned by a model $\mathcal{M}$ for three different dialog problems. Understanding these task-specific properties allows us to design an effective paradigm to facilitate zero-shot generalization. Concretely, knowledge of these properties influence (1) the design of the task specification (i.e., schema) and (2) the inductive biases in the model architecture and problem formulation. We begin with a neural network, $\mathcal{M}$, with general language understanding abilities and limited knowledge of task-oriented dialog (e.g., BERT [Devlin et al., 2018]). The high-level abstractions that $\mathcal{M}$ must learn are implied by the target dialog setting, i.e., the problem (e.g., next action prediction), the domain (e.g., restaurants) and the task (e.g., restaurant reservation). For three dialog problems (intent prediction, slot filling, next action prediction), we define the task-specific properties that $\mathcal{M}$ must learn.

Intent prediction is the problem of classifying an utterance $u \in \mathcal{U}$ to an intent $i \in \mathcal{I}$. An intent prediction model $\mathcal{M}_I$ must learn to produce similar representations $\mathcal{M}_I(u)$ for all utterances that have the same intent. Learning this abstraction is equivalent to transforming the unstructured representation space of

[1] A minimal expression can be defined as the *smallest* amount of data necessary to express a particular property. For example, one utterance to define an intent class.

$\mathcal{M}$ to the structured output space (i.e., the intent classes). Example-driven training (Section 4.3) explicitly learns this by predicting intents through the non-parametric process of comparing to example utterances.

In the problem of **slot filling**, for a given utterance $u = \{w_1, w_2, \ldots, w_n\}$ and a slot key $s \in \mathcal{S}$, we must predict the corresponding slot value for s. The value will either be a contiguous span from u, $w_{i:i+k}$, or none. A slot filling model $\mathcal{M}_S$ can be said to be capable of slot filling for a particular task, if it is able to (1) detect that an utterance provides a particular slot and (2) detect that a span of tokens corresponds to a slot value. As such, $\mathcal{M}_S$ must learn two sets of task-specific properties. First, the representation of u (or the contextual representation of $w \in u$) must be similar to other utterances that communicate the same slots. That is to say that $\mathcal{M}_S$ must learn the abstractions necessary for intent prediction (such that the intents are `{slot}-provided`). Second, each slot value representation $\mathcal{M}_S(w_{i:i+k})$ should be similar to other values for the slot s. These two task-specific properties impose structure on both the utterance-level and the span-level representations of $\mathcal{M}_S$.

The task-specific properties of **next action prediction** are more complex. Next action prediction is the problem of predicting the next system action $a \in \mathcal{A}$ conditioned on the dialog history $u_1, u_2, \ldots, u_n$ according to some task-specific dialog policy. Given the intents and slots in the dialog history, $\mathcal{I}_D = \{i_1, i_2, \ldots, i_m\}$ and $\mathcal{S}_D = \{s_1, s_2, \ldots, s_k\}$, the dialog policy can be expressed as a function of these intents and slots, $a = \mathrm{policy}(\mathcal{I}_D, \mathcal{S}_D)$. As such a next action prediction model $\mathcal{M}_A$ must learn (1) to predict intents, (2) to detect slots and (3) to generate system actions according to the mapping defined by the task-specific policy function.

6.2.2 Schema

While the task-specific properties conceptualize what a model $\mathcal{M}$ must learn, the schema (i.e., task specification) is a minimal expression of these constraints. Imagine that our objective is to train a human (i.e., $\mathcal{M}$ with human-level language understanding and reasoning abilities) to perform task-oriented dialog. The task-specific properties define *what* the human must learn. The schema is the *minimum* amount of information needed for the human to learn the necessary task-specific properties, without prior knowledge of the task.

For **intent prediction**, we define the schema to be a single utterance u for each intent $i \in \mathcal{I}$. The single utterance for each intent class is the minimum amount of information necessary for a model $\mathcal{M}$ to predict an unseen intent. **Slot filling** similarly relies on one utterance u for each slot type $s \in \mathcal{S}$. However, this one utterance only conveys the first property of slot filling. To ensure that $\mathcal{M}_S$ can learn meaningful span-level representations, the schema for slot filling also includes multiple[2] examples of values for each slot.

A **next action prediction** model must learn three task-specific properties: (1) intent prediction, (2) slot filling and (3) the task-specific dialog policy. To address the first two properties, the schema includes both (1) one utterance for each intent and (2) a set of slot values for each slot type. To effectively convey the task-specific dialog policy, we introduce a graph-based representation of the task-specific dialog policy.

[2]While the number of slot value examples could potentially reduced to 1, up to 20 are used in this thesis.

The task-specific policy graphs, visualized in Figure 6.1, consist of nodes for each user intent and system action. The edges in the graph define the dialog policy and thereby the desired system behavior. For example, in Figure 6.1, if a user indicates that they forgot their account number — the system must ask for their date of birth. These policy graphs have several noteworthy properties. First, the system actions are consistently deterministic. Nodes corresponding to a database response or to a user utterance will always have a single outgoing edge to a system response node. Furthermore, such nodes will also have a single incoming edge from a system response node. For a given user/database node, u, the previous system response node is denoted as $\textbf{prev}(u)$ and the following system response as $\textbf{next}(u)$. Each node has some text associated with it, denoted as $\textbf{text}(u)$. This text is a template for either a system utterance, database response or user utterance. System nodes will also have an associated system action, $\textbf{act}(u)$. There is a one-to-one mapping between the system actions and the system response templates.

Mosig et al. [2020] propose baseline policy graphs wherein the nodes of the graph correspond to system actions and database states. There are nodes for user intents only in situations where the system behavior differs depending on the user's actions (e.g., *'Yes'* $\rightarrow$ *ask-time*, *'No'* $\rightarrow$ *ask-date*). The consequence of this representation is that when the model aligns the dialog history to the schema, it largely relies on the system utterances. However, this representation fails to account for realistic user behavior and therefore the schema-guided model of Mosig et al. [2020] yields only marginal improvement over the baseline. Specifically, users will often provide information out of turn (e.g., *System: 'Where would you like to go?'* $\rightarrow$ *User: 'Leaving from the airport and going downtown'*). In this example, it is difficult for the model to realize that the question *System: 'Where are you leaving from?'* has also been answered and therefore should not be the next system action. Users can also ignore the system utterance (e.g., *System: 'Where would you like to go?'* $\rightarrow$ *User: 'Actually, what's the weather?'*). It is thus ineffective to represent dialog policy only in terms of the system utterances. To this end, the policy graphs leveraged in this thesis incorporate user utterances into the schema graph.

6.3 Task Specification as an Input

This work aims to induce generalization to unseen tasks by leveraging the task specification as an input, in the form of a structured graph representation of the task-specific dialog policy. The explicit task specification serves as an inductive bias that facilitates generalization to unseen dialog tasks. Rather than forcing models of dialog to implicitly memorize the task-specific dialog policy, the proposed Schema Attention Model (SAM) instead trains a dialog model that learns to leverage an explicit representation of the task specification (i.e., the schema) in order to complete the task. The Schema Attention Model, SAM, improves upon baseline models presented by Mosig et al. [2020] by introducing user-aware schema graphs as well as improving the model architecture and the training algorithm. SAM obtains a **+22 $\mathbf{F_1}$** score improvement over baseline approaches in the zero-shot setting, validating the use of the task specification as an inductive bias and demonstrating the feasibility of zero-shot generalization to new dialog tasks.

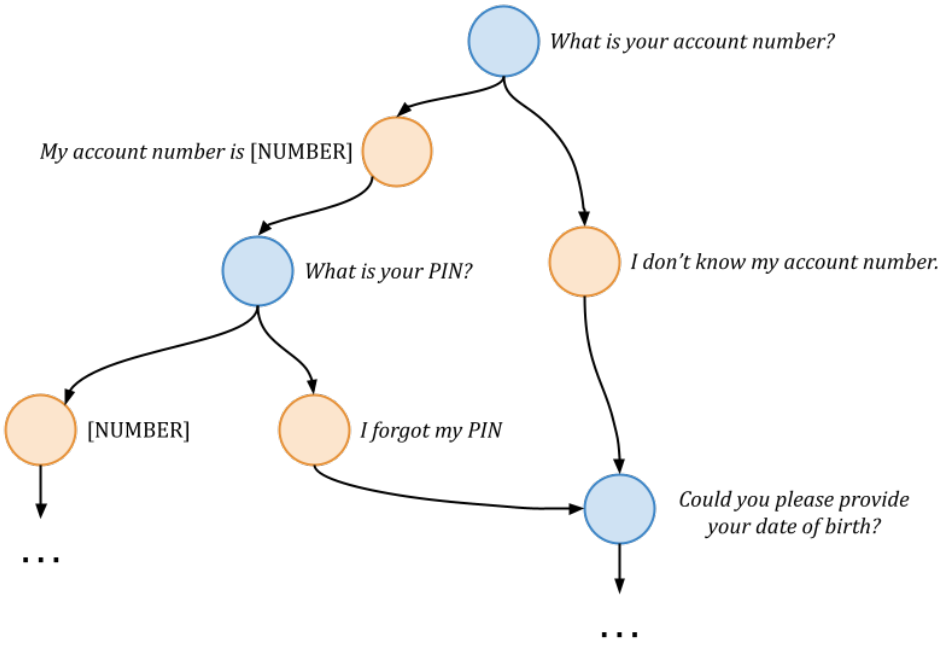


Figure 6.1: A section of the task-specific policy graph for the *bank-balance* task. The system must authenticate the user with their account number and PIN. However, if the user has forgotten either of these, it must ask backup security questions. The blue nodes correspond to system actions and the yellow nodes denote user utterances.

6.3.1 Task Definition

In the STAR dataset, there are 23 dialog tasks (13 domains) with single-task dialogs. This work performs two types of transfer learning experiments: task transfer and domain transfer. In task transfer, a model is trained on $n-1$ tasks (i.e., 22) and evaluated on the last one. This is repeated for each of the 23 tasks. For domain transfer, a model is trained on $n-1$ domains (i.e., 12) and evaluated on the last one. In task transfer, there may be some overlap between the training and testing, for example, the domain-specific terminology. In contrast, in domain transfer there is very limited overlap. When the model is tasked with generalizing to the restaurant domain, it has seen nothing related to restaurants during training.

In both of these settings, the model is aware of which task it is being evaluated on, meaning that it can leverage a *manually constructed* task specification (e.g., schema) for the new task. This experimental design resembles a real-world setting where a system developer would be aware of the new task. For example, if a developer wanted to extend a dialog system to handle a COVID-19 related questions, they would be able to manually create a new task specification. As such, the goal of this work is to develop a model that can generalize to an unseen task conditioned on a task specification. This still constitutes zero-shot transfer, as the dialog model has not observed any data from the target task.

6.3.2 Methods

In order to enable zero-shot transfer to new dialog tasks and domains, the Schema Attention Model (SAM) is introduced. It leverages the task-specific policy graphs as an inductive bias when predicting the next system action. This section begins by describing the baseline model for the task of next action prediction. Next, the schema-guided paradigm is introduced (Figure 6.2). The schema-guided paradigm leverages the task specification as an inductive bias, thereby facilitating zero-shot generalization to unseen tasks.

Baseline

This section describes the baseline model proposed by Mosig et al. [2020]. Given an arbitrary language encoder, denoted as $\mathcal{F}$, the baseline model obtains a vector representation of the dialog history, c. This representation is then passed through a softmax layer to obtain a probability distribution over the actions.

$$\boldsymbol{h} = \mathcal{F}(c) \tag{6.1}$$

$$P_{\text{clf}} = \text{softmax}(\boldsymbol{W}\boldsymbol{h}^T + \boldsymbol{b}) \tag{6.2}$$

Throughout this work, BERT-base [Devlin et al., 2018] is used as the language encoder.

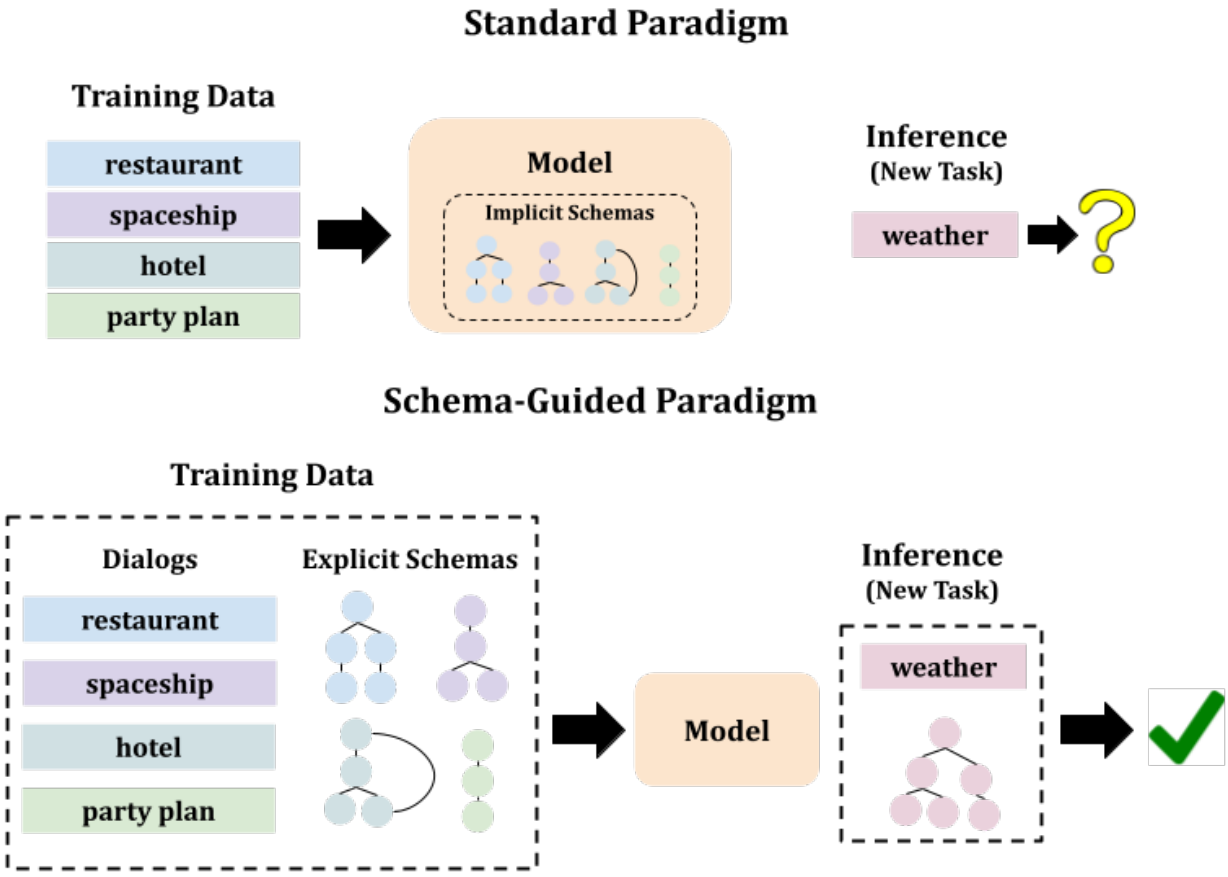


Figure 6.2: In the standard paradigm, data driven models implicitly learn the task-specific dialog policies (i.e., schemas). This precludes generalization to an unseen task at inference time. In contrast, in the schema-guided paradigm, dialog policy is explicitly provided to the model through a schema graph. At inference time, the model is given the schema for the new task and can therefore generalize in a zero-shot setting.

Schema-Guided Paradigm

The baseline model simultaneously needs to (1) interpret the dialog context and identify the relevant intents and slots, and (2) learn the task-specific dialog policies (i.e., if the user wants the weather, ask the city) for the different tasks in the training data. This model is incapable of generalizing to a new task in a zero-shot setting, as it would lack knowledge of the task-specific policy for the new task. To mitigate this problem and to enable zero-shot task transfer, the schema-guided paradigm decouples the task-specific dialog policy from the language understanding.

An example is shown in Figure 6.2: the schema-guided paradigm explicitly provides task-specific schema graphs as input to the model in order to decouple the dialog policy from language understanding . These schema graphs serve as complete representations of the dialog policy for a given task. Therefore, while the baseline needs to implicitly learn the dialog policies, a schema-guided model instead learns to leverage the explicit task specification. As such, a schema-guided model can generalize to a new task as

long as it is provided with the corresponding schema.

In this paradigm, the role of the model is to interpret a dialog context and align it to the explicit schema. The role of the task-specific policy graphs is to determine the next action according to the dialog policy. In this manner, the language encoder is being trained for the task of sentence similarity. With the help of pre-trained models, language understanding in a schema-guided paradigm can be considered to be task-agnostic. By decoupling the task-agnostic language understanding and the task-specific dialog policy, the schema-guided paradigm better facilitates zero-shot transfer learning.

Schema Attention Model

In the schema-guided paradigm, the role of the model is to understand the dialog history and align it to the schema representation. The **Schema Attention Model**, SAM, attends between the dialog history, $c = c_1, \ldots, c_N$ and the schema. SAM extends the schema-guided model presented by Mosig et al. [2020] by (1) leveraging a stronger attention mechanism, (2) improving the training algorithm, and (3) removing the linear classification layer which is detrimental to zero-shot performance.

The objective of SAM is to predict the node in the schema graph that best corresponds to the dialog context. SAM will produce a probability distribution over the nodes corresponding to user utterances and database responses. Given an attention distribution over the nodes, SAM can obtain a probability distribution over the set of actions by propagating the attention probabilities over the graph. Concretely, if node u has an attention weight of p, we add p to the probability of **action**(**next**(u)).

Every node u that corresponds to either a database response or a user utterance is considered. Each node u is then represented as the concatenation of the previous node and the current node, i.e., **text**(**prev**(u)) + **text**(u). For all nodes $u \in U$, this textual representation is obtained and is denoted as $s \in S$.

We are given a language encoder, $\mathcal{F}$, the dialog context, $c = c_1, \ldots, c_N$, the nodes U, their corresponding textual representations S, and the set of possible actions A. Note that unlike in Equation 6.2, $\mathcal{F}$ is used to produce a vector representation of each word in the input. SAM produces a probability distribution over the actions as follows:

$$\boldsymbol{h}_{1,\ldots,N} = \mathcal{F}(c\colon c_1, \ldots, c_N) \tag{6.3}$$

$$\boldsymbol{S}_{i;1,\ldots,M} = \mathcal{F}(S_i\colon s_1, \ldots, s_M) \tag{6.4}$$

$$\boldsymbol{w}^i_{j,k} = \boldsymbol{h}^T_j \boldsymbol{S}_{i;k} \tag{6.5}$$

$$\alpha = \text{softmax}(\boldsymbol{w}^{1,\ldots,|S|}) \tag{6.6}$$

$$p_i = \sum_{j \leq N} \sum_{k \leq M} \alpha^i_{j,k} \tag{6.7}$$

Here, $\boldsymbol{w}^i$ is an $N \times M$ dimensional matrix corresponding to the dot product between the N words of the dialog history and the M words of the i-th textual representation in S. To get the attention weights over all

of the words of the schema, a softmax is performed over all $\boldsymbol{w}^i, 1 \leq i \leq |S|$. Summing over the attention weights in α^i produces p_i, a scalar value which denotes the attention between the dialog history and the i-th node (i.e., the corresponding textual representation S_i). Given p_i, a probability distribution over the actions A is produced as follows:

$$g(i, a) = \begin{cases} p_i, & \text{if } \textbf{action}(\textbf{next}(u_i)) = a \\ 0, & \text{otherwise} \end{cases} \tag{6.8}$$

$$P(a) = \sum_{i \leq |S|} g(i, a) \tag{6.9}$$

To align the dialog history to the schema graph, SAM performs word-level attention using a BERT-base model. In contrast, the schema-guided model of Mosig et al. [2020] attends with the sentence level vector representation produced by BERT. With the word-level attention, SAM can better align ambiguous dialog contexts, such as situations where the user provides multiple pieces of information in a single utterance. Since this word-level attention operates on the sub-word tokens used in BERT, it can also potentially handle spelling errors in the user utterances.

Furthermore, in their schema-guided model, Mosig et al. [2020] combine the probability distribution produced by attending to the schema graph with their baseline model (i.e., Section 3.1). While this may result in better performance on the tasks the model is trained with, the baseline model will not generalize to unseen tasks. In contrast, SAM computes the probability for an action using only the attention over the schema graph.

Mosig et al. [2020] train their schema-guided model to predict the appropriate node, u_i, from a set of nodes U' (s.t., $U' \subset U$). At training time, for efficiency reasons, the set of nodes U' is obtained by using the corresponding node for every dialog context in the training batch. Since the training batches are randomly sampled, this results in U' including nodes from a variety of different schema graphs. At inference time, the dialog task is known and therefore only the corresponding schema graph needs to be attended to (i.e., U' will contain nodes from a single schema graph). It is valuable to train the model to distinguish between different nodes of the same schema graph. Specifically, the attention mechanism will learn stronger fine-grained relationships when trained with negative samples from the *same* domain. As such, the training algorithm is augmented to sample batches from the same dialog task, meaning that U' will only include nodes from a single schema.

SAM improves on the baseline schema-guided model introduced by Mosig et al. [2020] by (1) leveraging a stronger attention mechanism that better handles realistic user behavior, (2) computing a probability distribution *only* by attending to the schema graph and (3) modifying the training algorithm to have in-domain negative samples which result in the model learning to identify fine-grained relationships. In combination with the improved schema representation, SAM is better suited to handle realistic user behavior in zero-shot settings.

Model	F_1 score	Accuracy
Baseline ⋄	**73.79**	**74.85**
BERT+S ⋄	71.59	72.27
SAM − [1]	54.35	60.51
SAM − [2,3,4]	70.22	71.01
SAM − [2]	70.27	71.93
SAM − [3]	70.18	71.64
SAM − [4]	69.68	69.79
SAM	70.38	71.45

Table 6.1: Performance in the standard experimental setting. Models marked with ⋄ are attributed to Mosig et al. [2020]. We denote their schema-guided model, 'BERT + *Schema*', as BERT+S. SAM consists of four improvements upon BERT+S: (1) user-aware schema, (2) word-level attention, (3) using negative samples from the same task at training, (4) removing the linear classification layer. Results in boldface are statistically significant by t-test ($p < 0.01$)

Model	Task Transfer		Domain Transfer	
	F_1 score	Accuracy	F_1 score	Accuracy
Baseline ♢	31.23	30.65	31.82	33.92
BERT+S ♢	28.12	28.28	29.70	32.43
SAM − [1]	33.81	37.84	41.77	45.64
SAM − [2,3,4]	43.28	46.11	43.78	45.19
SAM − [2]	50.72	53.69	52.20	54.68
SAM − [3]	45.54	49.29	50.56	52.13
SAM − [4]	47.26	47.99	47.67	48.92
SAM	**53.31**	**55.51**	**55.74**	**57.75**

Table 6.2: Performance in zero-shot transfer. We present results on both task transfer and domain transfer. Models marked with ♢ are attributed to Mosig et al. [2020]. SAM consists of four improvements upon BERT+S: (1) user-aware schema, (2) word-level attention, (3) using negative samples from the same task at training, (4) removing the linear classification layer. Results in bold-face are statistically significant by t-test ($p < 0.01$).

6.3.3 Experiments

To validate the effectiveness of SAM, a number of *next action prediction* experiments are carried out on the STAR dataset [Mosig et al., 2020]. First, SAM is evaluated in the standard experimental setting, i.e., training and testing on the same tasks. Next, SAM is evaluated in zero-shot transfer experiments. The evaluation uses accuracy and weighted F_1 score.

The experiments presented by Mosig et al. [2020] are rerun using the same code. In the following results, the model introduced by Mosig et al. [2020] is denoted as BERT+S. Their original results were obtained on an older version of STAR, with annotation errors[3] that have since been fixed.

Standard Experiments

In the standard experimental setting, models are trained and tested on the same tasks. Following Mosig et al. [2020], 80% of the dialogs are used for training and 20% for testing. All models are trained for 50 epochs.

The results shown in Table 6.1 show SAM to be comparable to the baseline model on the standard setting. Since the augmentations to SAM are primarily intended to induce zero-shot generalization to new tasks, it is unsurprising that there is no performance improvement compared to the standard setting. When evaluating on *seen* tasks, the linear classification layer is significantly more effective than attending to the schema. This suggests that a large neural model (i.e., BERT) is able to implicitly learn meaningful dialog policies from dialog data. It is possible that this performance difference may decrease with more expressive schemas (e.g., having multiple examples for each user utterance, automatically learning schemas from the dataset). The value of the newly introduced schema graphs is nonetheless shown when comparing SAM to SAM$-[1]$ (i.e., the old schema graphs). These experiments provide an upper bound for the performance in zero-shot transfer.

Zero-Shot Transfer

Table 6.2 shows the results of the zero-shot experiments. SAM obtains strong improvements over the baseline models for both zero-shot task transfer and domain transfer. These experimental results validate the effectiveness of the schema-guided paradigm, as well as the specific design of SAM.

Compared to the baseline model (described in Section 3.1), SAM obtains a $+\mathbf{22}\ \mathbf{F_1}$ score improvement in task transfer and a $+\mathbf{24}\ \mathbf{F_1}$ score improvement in domain transfer. Since the baseline model is unable to predict classes it has not observed at training time, its performance is limited to actions that are consistent across domains (e.g., *'hello'*, *'goodbye'*, *'anything-else'*). This improvement highlights the effectiveness of the schema-guided paradigm for zero-shot transfer learning.

BERT+S also leverages schemas for transfer learning. Yet, it under-performs relative to the baseline model. SAM attains even larger improvements over this baseline schema-guided model. The weak performance of BERT+S is largely a consequence of it being incapable of handling realistic user behavior. The

[3] Specifically, certain dialogs were misattributed as being *happy* single-task dialogs.

design of BERT+S (i.e., the schema only having system nodes) results in the model essentially predicting the subsequent system actions. This is equivalent to sequentially predicting the next system action, regardless of user behavior. With improved schema representations and model architecture, SAM achieves much stronger performance in zero-shot transfer.

The ablation experiments shed more light on the performance of SAM relative to BERT+S. A significant performance drop is observed when removing the newly constructed schema representations (i.e., SAM$-[1]$). In contrast, adding the schema graphs to BERT+S (i.e., SAM$-[2, 3, 4]$) results in a strong performance improvement of $+\mathbf{15}$ $\mathbf{F_1}$ score. This confirms the hypothesis that the schema graphs of Mosig et al. [2020], which are largely comprised of system action nodes are insufficient for modelling realistic user behavior.

Word-level attention is shown to give moderate, albeit statistically significant, improvement. In contrast to SAM$-[2]$, SAM obtains a $+\mathbf{3}$ $\mathbf{F_1}$ score improvement. While word-level attention allows the model to better align the dialog to the schema, it is an architectural improvement that is not central to the schema-guided paradigm.

Modifying the training algorithm to sample batches from the same task results in better negative samples during training. This allows the model to learn to distinguish between nodes from the same schema graph when aligning the dialog to the schema graph. When this modification is removed (i.e., SAM$-[3]$), the performance of SAM drops by $\mathbf{8}$ $\mathbf{F_1}$ score for zero-shot task transfer.

The fourth and final component of SAM is the removal of the linear classification layer. Since this classification layer is unable to predict classes it has not seen at training time, it is ineffective in zero-shot settings. Unsurprisingly, removing it increases performance and SAM obtains a $+\mathbf{6}$ $\mathbf{F_1}$ score improvement over SAM$-[4]$.

The zero-shot experiments shown in Table 6.2 empirically validate several hypotheses. First, the strong improvement over the baseline demonstrates the efficacy of the schema-guided paradigm for inducing zero-shot generalization to new tasks in end-to-end dialog. Decoupling dialog policy and the language understanding by explicitly representing the task-specific dialog policies as schema graphs results in an improved ability to transfer to unseen tasks. Next, the proposed approaches improve over the schema-guided model of Mosig et al. [2020] through (1) an improved schema representation and (2) a collection of modifications to the model. The improved schema representation better models realistic user behaviors in dialog, and therefore results in better alignment of the dialog and the schema. The proposed architectural modifications result in the model being able to learn better fine-grained relationships during alignment (e.g., through better negative sampling and word-level attention) and better handle zero-shot transfer (e.g., by removing the linear layer).

In contrast to prior work on zero-shot generalizability [Zhao and Eskenazi, 2018; Qian and Yu, 2019], this approach is shown to effectively transfer between the vastly dissimilar domains of the STAR corpus [Mosig et al., 2020] (e.g., trivia or spaceship maintenance). Rather than modelling a cross-domain mapping and leveraging similar concepts across different domains, the schema-guided paradigm *decouples* the domain-specific (i.e., the dialog policy) and domain-agnostic (i.e., language understanding) aspects of dialog systems.

6.3.4 Discussion

This work shows strong results in zero-shot task transfer and domain transfer using the schema-guided paradigm. This work was motivated by the hypothesis that the difficulty of zero-shot transfer in dialog stems from the dialog policy. When neural models implicitly memorize dialog policies observed at training time, they struggle to transfer to new tasks. To mitigate this, the schema-guided paradigm explicitly provides the task specification as an *input* to the model, in the form of a schema graph. This work carries out an initial study of the schema-guided paradigm by introducing the Schema Attention Model (SAM) and improved schema graphs for the STAR corpus. The proposed approach attains significant improvement over prior work in the zero-shot setting, with a $+\mathbf{22}$ $\mathbf{F_1}$ **score improvement**. Furthermore, the ablation experiments demonstrate the effectiveness of both SAM and the improved schema representations. This work validates the efficacy of using the *task specification as an inductive bias*, and takes an important step towards inducing generalization to unseen tasks.

6.4 Task Specification for Synthetic Data Creation

The work on the Schema Attention Model takes an important step towards validating the schema-guided paradigm as mechanism for facilitating generalization to new tasks. However, this work has two key limitations. First, it only addresses the problem of next action prediction. Next, and more importantly, providing the task specification as an input may be insufficient as a means of effectively prescribing specific learned abstractions. Though using the task specification as an input yields significant improvements in zero-shot generalization for next action prediction — it is possible that a data-driven model may ignore the task specification, particularly when extending the approach to other problems. This section aims to (1) better learn the task-specific properties by using the task specification as an inductive bias and (2) leverage the strengths of large language models (GPT-3; Brown et al. [2020]), by using the task specification to create synthetic data.

The advent of large-scale pre-training [Devlin et al., 2018; Liu et al., 2019b; Zhang et al., 2019c] has brought about significant progress in few-shot and zero-shot generalization across many different problems in Natural Language Processing [Brown et al., 2020; Wei et al., 2021], zero-shot generalization in **task-oriented dialog** remains elusive. A likely reason for this discrepancy is that dialog models require significant data because they need to learn task-specific properties, such as the domain ontology and the dialog policy. While large language models (e.g., GPT-3) exhibit strong language understanding and generation abilities [Brown et al., 2020], they have no *a priori* knowledge of the task-specific properties implied by a specific (unseen) problem setting (e.g., relevant intents, dialog policy, etc.). As such, in order to adapt a pre-trained LM for task-oriented dialog, it is necessary to impose specific abstractions on the unstructured representation space of a pre-trained model. Fine-tuning moderately-sized language models (LMs) (e.g., BERT) with well-motivated inductive biases facilitates sample-efficient learning of the necessary high-level abstractions [Peng et al., 2020a; Henderson and Vulić, 2020; Mehri and Eskenazi, 2021]. However, fine-tuning can be

impractical (e.g., in academic settings) with large LMs (e.g., GPT-3) due to the cost, computational power and immutable architectures. To this end, this work aims to leverage the strong language understanding and generation abilities of large LMs to facilitate **zero-shot generalization** in task-oriented dialog by using the task-specific as an inductive bias.

Given the in-context meta-learning abilities of large LMs [Brown et al., 2020], prior work has explored prompt-engineering or prompt-tuning [Reynolds and McDonell, 2021; Lester et al., 2021; Madotto et al., 2021]. Well-designed prompts can convey the necessary task-specific properties. However, it is challenging to express complex properties (e.g., a dialog policy) in natural language. Prompting also precludes inductive biases in the model (architecture, training algorithm, etc.) and over-relies on the meta-learning abilities of large LMs. As such, there is a tradeoff between prompting large LMs (i.e., generalizable NLU and NLG) and fine-tuning smaller LMs (i.e., problem-specific inductive biases, efficiency). A potential interpretation for the strength of large LMs is that they learn the distributional structure of language [Harris, 1954] by observing web-scale data [Sinha et al., 2021]. Motivated by this interpretation, this work proposes *Language Models as Data* (**LAD**).

LAD is a novel paradigm in which large LMs are used in a zero-shot domain-agnostic manner to induce *linguistic diversity* in synthetic data. Given a task specification (i.e., schema), LAD (1) creates a seed synthetic dataset using domain-agnostic algorithms, (2) leverages large LMs to *reformulate* utterances, and (3) validates the resulting data to ensure adherence to the schema. The resulting synthetic data, which is sufficiently diverse and expresses the necessary task-specific properties, can be used to train neural dialog models. In contrast to prompting, LAD facilitates zero-shot generalization by (1) leveraging the sophisticated abilities of large LMs (knowledge of the distributional structure of language) to induce *linguistic diversity* in the synthetic data while (2) maintaining inductive biases in the problem-specific model architectures.

The challenge of creating synthetic data that is indistinguishable from human-annotated data, both in its expression of the task-specific properties and in its diversity, is highly impractical [Lin et al., 2021; Feng et al., 2021]. Instead, the goal of this work is to create synthetic data that is *sufficient* to train a sample-efficient and robust model. Therefore, the claim of this work is that LAD can create synthetic data, conditioned on a task specification (i.e., a schema), that can be used to train robust and sample-efficient neural models and induce performance gains in zero-shot settings.

To validate this claim, LAD is applied to three problems in dialog: intent prediction, slot filling and next action prediction. Next action prediction is particularly difficult in zero-shot settings since the task-specific properties include the *dialog policy*. LAD demonstrates significant gains across five datasets (**+10** to **+30** improvements on F-1 and accuracy) in zero-shot settings when evaluating on human-annotated corpora. To further validate the efficacy of LAD, an interactive evaluation with humans (over 1600 dialogs) is performed. The results of this interactive evaluation suggest that LAD can yield performance comparable to training on human dialogs. The aforementioned claim is validated empirically across multiple datasets. LAD is shown to generate diverse and accurate synthetic data, which is subsequently used to train neural dialog models and facilitate zero-shot generalization.

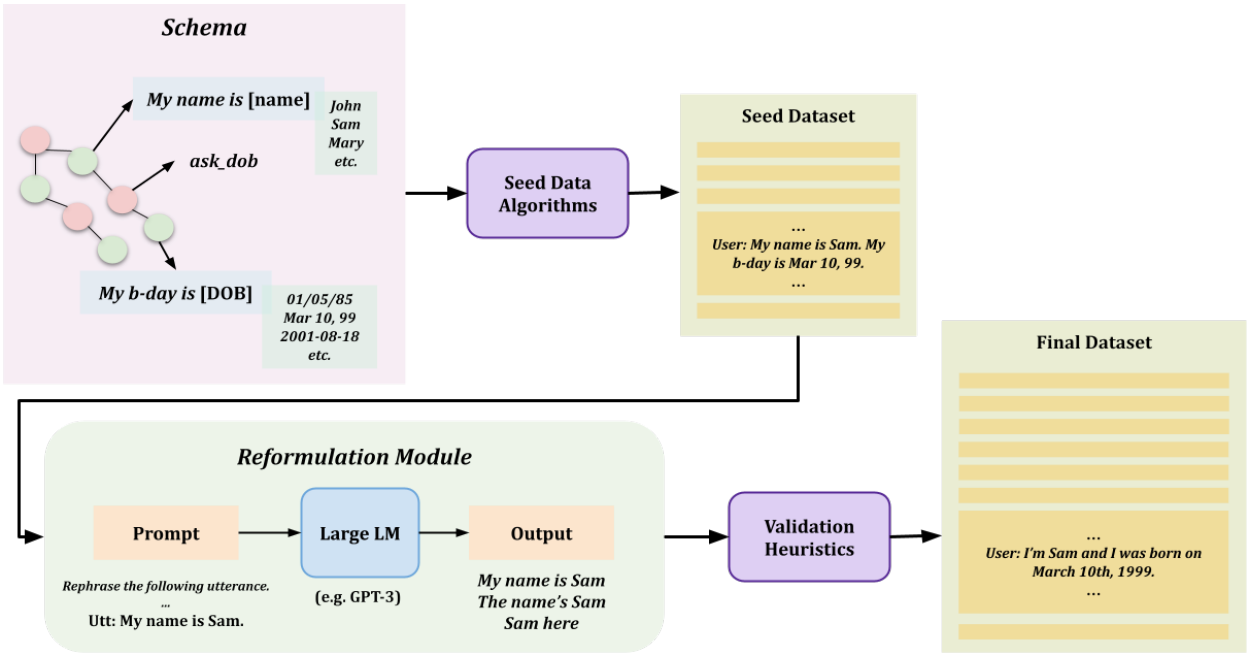


Figure 6.3: Visualization of LAD. (1) Domain-agnostic algorithms use the schema to create a seed dataset which conveys the necessary task-specific properties (e.g., dialog policy, domain ontology). (2) Large LMs reformulate individual utterances to add linguistic diversity. (3) Validation heuristics are used to ensure adherence to the schema.

6.4.1 Methods

LAD is a framework for inducing zero-shot generalization in task-oriented dialog by creating *diverse* and *accurate* synthetic data. LAD, visualized in Figure 6.3, is a three step process: (§6.4.1) domain-agnostic algorithms generate a *seed dataset* from a schema, (§6.4.1) GPT-3 *reformulates* utterances in order to induce linguistic diversity, (§6.4.1) heuristics are used to *validate* the reformulated data to ensure adherence to the schema. LAD facilitates zero-shot generalization by explicitly leveraging the strengths of large LMs (knowledge of the distributional structure of language) without sacrificing the inductive biases in the downstream neural dialog models.

Seed Data Creation

LAD begins by creating seed synthetic data from a given schema. This is a domain-agnostic process that aims to generate synthetic data which accurately convey the necessary task-specific properties. A downstream data-driven model trained on this synthetic data is expected to learn the necessary task-specific properties from the synthetic data.

For **intent prediction**, the schema consists of one utterance for each intent class (sampled from the original corpus) and is used verbatim as the seed dataset. For **slot filling**, the schema consists of one manually-written template utterance and multiple slot values for each slot type. To construct the seed data: (1) begin

with the utterance templates from the schema (e.g., *'My first name is {first_name}'*), (2) exhaustively combine template utterances to ensure coverage of slot type combinations, and (3) fill slot values by sampling from the schema.

Since the schema does not contain information about the co-occurrence of different slot types, we assume a uniform distribution. This is a simplifying assumption that will not hold true for human utterances (e.g., humans will rarely fill five slots in a single utterance), however for the purposes of teaching a model the necessary task-specific properties, this assumption is sufficient.

The relative complexity of the task-specific dialog policy for **next action prediction**, necessitates a more sophisticated algorithm for generating the seed data. In order to avoid over-fitting and to ensure that the task-specific properties are effectively learned by the model, it is imperative that the synthetic data produced by LAD be diverse and realistic. While linguistic diversity is induced through the reformulation with GPT-3, the synthetic dialogs created for next action prediction must also exhibit diversity of *user behavior*. The dialog policy expressed by the schema deterministically defines the *system* behavior. However, users should be able to deviate from the policy, e.g. by providing information out of turn. To account for this, Algorithm 1 generates a dialog by traversing the dialog policy graph and randomly combining multiple template utterances (e.g., *'**System:** What is your name? **User:** My name is John. My phone number is...'*).

In Algorithm 1, the choice of user actions (intents, slots) are randomly sampled and randomly combined. The system action is deterministically expressed by the schema. The policy function identifies the appropriate system action by considering the policy graph, $\mathcal{G}_{policy}$ and the partial dialog $\mathcal{D}$. The random combination of user actions ensures that the synthetic data exhibits sufficient diversity of user behavior.

Reformulation

To ensure that downstream neural dialog models can effectively learn the task-specific properties, it is imperative that the synthetic data is sufficiently diverse. The seed synthetic data is formulaic and artificial: (1) there is a single template utterance for each user action and (2) when multiple user actions are combined they are simply concatenated. As such, the goal of the reformulation step is two-fold: (1) to induce linguistic diversity and (2) to rephrase concatenations of disjoint template utterances (*'My name is Sarah. I want to plan a party. The day should be Sunday'*) into a natural utterance (*'I'm Sarah and I'd like to plan a party for Sunday.'*).

To reformulate utterances in a domain-agnostic manner, LAD leverages the in-context meta-learning abilities of GPT-3 [Brown et al., 2020]. Through manual experimentation in the OpenAI Playground[4], an appropriate prompt is constructed. The prompt begins with an instruction (*'Given a set of sentences, generate 5 natural utterances that convey the same meaning.'*) and includes six examples (full prompt in Table 6.3). Rather than producing a single reformulation of the input, the chosen prompt instructs GPT-3 to generate **five** utterances. Through the examples provided in the prompt, GPT-3 learns that it should produce five *diverse* reformulations. As such, linguistic diversity is induced through both the decoding algorithm

[4]https://beta.openai.com/playground

Algorithm 1 Next action prediction seed data generation. This algorithm defines the process of generating a *single* dialog from the schema. The input to this function is the schema, consisting of a set of utterances, slot values and the graph-based representation of the dialog policy.

function GENDIALOG($\mathcal{U}, \mathcal{S}, \mathcal{G}_{policy}$)
  $cur \leftarrow \text{start\_node}(\mathcal{G}_{policy})$
  $\mathcal{D} \leftarrow \{\}$
  while $cur \neq \text{end\_node}(\mathcal{G}_{policy})$ **do**
    $r \leftarrow \text{random}(0, 1)$
    $u_{cur} \leftarrow \text{utt}(\mathcal{U}, cur)$
    if $r < 0.15$ **then**
      $n_2 \leftarrow \text{sample}(\text{unvisited}(\mathcal{G}_{policy}))$
      $n_3 \leftarrow \text{sample}(\text{unvisited}(\mathcal{G}_{policy}))$
      $u_2 \leftarrow \text{utt}(\mathcal{U}, n_2)$
      $u_3 \leftarrow \text{utt}(\mathcal{U}, n_3)$
      $u_{cur} \leftarrow \text{concat}(u_{cur}, u_2, u_3)$
    else if $r < 0.5$ **then**
      $n_2 \leftarrow \text{sample}(\text{unvisited}(\mathcal{G}_{policy}))$
      $u_2 \leftarrow \text{utt}(\mathcal{U}, n_2)$
      $u_{cur} \leftarrow \text{concat}(u_{cur}, u_2)$
    else
      $u_{cur} \leftarrow u_{cur}$
    end if

    $u_{cur} \leftarrow \text{sample\_slots}(\mathcal{S}, u_{cur})$
    $\mathcal{D} \leftarrow \mathcal{D} + u_{cur}$

    $n_{sys} \leftarrow \text{policy}(\mathcal{D}, \mathcal{G}_{policy})$
    $u_{sys} \leftarrow \text{utt}(\mathcal{U}, n_{sys})$
    $\mathcal{D} \leftarrow \mathcal{D} + u_{sys}$

    $cur \leftarrow \text{sample}(\text{next}(\mathcal{D}, \mathcal{G}_{policy}))$
  end while
  return $\mathcal{D}$
end function

and the six examples in the prompt.

Scalability

The cost of the GPT-3 API is approximately $0.05 USD per reformulation. In order to generate a substantial amount of synthetic data without incurring significant costs, the reformulation step of LAD must be performed in a scalable manner. The seed utterances are grouped by their intents and slot keys (e.g., 'name;date;time', 'name;date', 'date;time'). A subset of utterances in each group is reformulated. These reformulated utterances are used as templates and the slot values are randomly replaced. In this manner, the cost scales with respect to the number of distinct intent/slot combinations rather than the desired size of the synthetic dataset.

Validation

The seed data will always adhere to the schema and therefore *accurately* convey the necessary task-specific properties. However, the reformulated utterances may not be accurate. GPT-3 may modify the intended meaning of an input utterance, for example by ignoring certain slot values. To ensure that the task-specific properties are accurately expressed in the final dataset, the reformulation step of LAD filters out erroneous reformulations. For slot filling and next action prediction, this is done by ensuring that all of the slot values present in the original utterance (from the seed dataset) are also present in the reformulated utterances (produced by GPT-3).

6.4.2 Experiments

This work aims to demonstrate that LAD can use a task-specification as an inductive bias to create a sufficiently diverse and accurate synthetic dataset, which can be used to train neural dialog models and facilitate performance gains in zero-shot settings. To validate this claim, experiments are carried out on intent prediction, slot filling and next action prediction across five datasets. For each problem, an appropriate model is identified from the previous chapters of this thesis. The chosen models have been shown to exhibit strong zero-shot and few-shot generalization through various forms of inductive biases. Though LAD is not guaranteed to produce *perfectly* accurate and diverse data, the inductive biases in the chosen models make them more robust to potential errors and limitations in the synthetic data.

Synthetic Datasets

LAD is evaluated on five different datasets. For intent prediction, Banking77 [Casanueva et al., 2020], CLINC150 [Larson et al., 2019], and HWU64 [Liu et al., 2019a] are used. For slot filling, Restaurant8k [Coope et al., 2020] is used. For next action prediction, STAR [Mosig et al., 2020] is used. Given a human-annotated corpus, a schema is created to express the necessary constraints. LAD is then leveraged to create

GPT-3 Prompt

Given a set of sentences, generate 5 natural utterances that convey the same meaning.

==

Original: My name is John. I want to reserve a flight.

1. The name is John and I need to reserve a flight.
2. I'm John and I'd like to book a flight.
3. I'm booking a flight and my name is John.
4. I want to reserve a flight, under the name John.
5. I am John and I'd like to make a flight reservation.

==

Original: The name of the hotel is Shadyside Inn. I will be arriving May 7th. I will be leaving May 27th.

1. I want a reservation for the Shadyside Inn from May 7th to the 27th.
2. Please reserve the Shadyside Inn from the 7th to the 27th of May.
3. I want you to reserve the Shadyside Inn from May 7th to the 27th.
4. I'll be in town from May 7th to the 27th and I'd like to stay at the Shadyside Inn during that time.
5. Give me a room at the Shadyside Inn from May 7th to the 27th.

==

Original: I would like to follow-up on my appointment with Dr. Veronica. My name is Andy. My appointment was on Friday.

1. I'm Andy and I had an appointment with Dr. Veronica on Friday. I'd like to follow up with her.
2. My name is Andy, I'd like to follow up on my appointment with Dr. Veronica on Friday.
3. I had an appointment with Dr. Veronica last Friday and I'd like to follow up. My name is Andy.
4. Andy here. I'm trying to follow-up on my appointment with Dr. Veronica last Friday.
5. Can you help me follow-up on an appointment with Dr. Veronica last Friday? My name is Andy.

==

Original: Thanks

1. Thank you!
2. Thanks for your help.
3. Thank you so much.
4. Thank you for helping
5. thx

==

Original: The meeting should be on Monday. The reason for the meeting is to allocate department budgets.

1. The meeting is Monday and the reason for the meeting is to allocate department budgets.
2. The meeting will be held Monday to discuss department budgets.
3. The meeting is scheduled for Monday to discuss department budgets.
4. We will have the meeting on Monday to allocate department budgets.
5. Let's meet on Monday to allocate department budgets.

==

Original: Hello.

1. Hey!
2. hi
3. Hi there
4. Hello, can you help me?
5. Yo

Table 6.3: Prompt used to induce linguistic diversity with GPT-3.

Original Dataset	Seed	LAD	Cost (USD)
Intent Prediction			
HWU64 (8955)	64	800	$19
CLINC150 (15000)	150	1664	$43
Banking77 (8633)	77	848	$25
Slot Filling			
Restaurant8k (8633)	85	32000	$89
Next Action Prediction			
STAR (1200)	24000	22327	$226

Table 6.4: Statistics for the synthetic datasets created by LAD. This table lists the size of the original dataset, the seed dataset and the final synthetic dataset produced by LAD. The last column indicates the approximate cost of using GPT-3 for each of the datasets.

Model (Training Data)	BANKING77	CLINC150	HWU64
CBEO (ONE-SHOT)	31.36	53.96	43.12
CBEO (ONE-SHOT + LAD)	**51.17**	**68.11**	**65.50**
CBEO (FULL-SHOT)	93.83	97.31	93.03

Table 6.5: Experimental results on intent prediction. We report the accuracy of training CBEO on (1) one utterance/intent (i.e., the seed data) and (2) the synthetic data produced by LAD. For reference, we also show the results obtained with full-shot training on the human-annotated datasets.

a synthetic dataset conditioned on the schema. Table 6.4 describes the size and creation cost of each of the synthetic datasets.

Intent Prediction

CONVBERT+*Example-Driven+Observers* (CBEO) (Section 4.3) is used for intent prediction. CBEO learns to predict utterance intents by explicitly comparing to a set of examples. Predicting intents through an explicit non-parametric comparison to examples is an inductive bias that facilitates sample-efficient learning of the task-specific properties.

The experimental results shown in Table 6.5 demonstrate that the synthetic data produced by LAD significantly increase performance on one-shot[5] intent prediction. LAD facilitates **15%+** accuracy improvement across all three intent prediction datasets. For intent prediction, LAD does not use any heuristics during the creation of the seed data or during the validation step. As such, these improvements can be attributed to the reformulation step, which leverages the prompt-driven generation abilities of GPT-3 [Brown et al., 2020].

[5]This setting is characterized as one-shot since the utterances in the schema are sampled from the respective dataset.

Model	F-1
Zero-Shot Results	
ConVEx [Henderson and Vulić, 2020]	5.2
Coach+TR [Liu et al., 2020]	10.7
GenSF (Section 5.3)	19.5
GenSF + LAD	**50.9**
Non Zero-Shot Results	
GenSF (64 utterances)	72.2
GenSF (8633 utterances)	96.1

Table 6.6: Experimental results on the Restaurant8k corpus. We compare GenSF + LAD with zero-shot results reported by prior work. For reference, we also show the performance of models (reported by prior work) when trained in few-shot and full-shot settings.

Slot Filling

For slot filling, experiments are carried out with GenSF (Section 5.3) which is the state-of-the-art model on the Restaurant8k corpus [Coope et al., 2020], in both zero-shot and full-shot settings. GenSF reformulates slot filling as response generation in order to better leverage the capabilities of DialoGPT [Zhang et al., 2019c]. As shown in Table 6.6, GenSF + LAD achieves a **+31.4** F-1 improvement over GenSF on the test set of Restaurant8k, without observing any examples from the corpus. GenSF + LAD learns to detect slots in the restaurant domain given only the schema, which consists of (1) a single manually written utterance for each slot type and (2) a collection of up to 20 slot values for each slot type. This significant performance improvement in zero-shot generalization validates the claim of this work for the problem of slot filling. LAD is able to create synthetic data which effectively teaches GenSF the necessary task-specific properties. However, GenSF achieves a 72.2 F-1 score by only observing 64 human-written examples. Despite the relative success of LAD in zero-shot settings, there remains significant room for improvement.

Next Action Prediction

Next action prediction is particularly challenging due to the complexity of the task-specific dialog policy. In addition to the task-specific properties of intent prediction and slot filling, next action prediction models must also learn to follow the *dialog policy*. SAM (Section 6.3) learns to predict the system action by attending to a graph-based representation of the dialog policy.

Table 6.7 shows the results for three models. BERT+S [Mosig et al., 2020] trains a BERT model to attend to a rudimentary graph-based representation of the dialog policy. SAM [Mehri and Eskenazi, 2021] improves the model architecture and introduces more expressive policy graphs. These two models are trained on the STAR corpus, which includes 24 different tasks and 24 different policy graphs. The zero-shot

Model	F-1
Zero-Shot Results	
BERT+S [MOSIG ET AL., 2020]	28.12
SAM [MEHRI AND ESKENAZI, 2021]	53.31
SAM + LAD	**64.36**
Full-Shot Results	
SAM [MEHRI AND ESKENAZI, 2021]	70.38

Table 6.7: Experimental results on the STAR corpus. SAM + LAD is compared with zero-shot results reported by prior work. For reference, the performance of SAM when trained on the full corpus is also shown.

Model (Training Data)	COMPLETE %	ASKS ALL %	AVOIDS REDUNDANCY %
SAM (ZERO-SHOT)	98.02	76.15	78.90
SAM (FULL-SHOT)	98.31	75.69	80.65
SAM + LAD	98.52	**78.39**	79.13

Table 6.8: Results of the interactive human evaluation. We compare three models: (1) SAM (ZERO-SHOT), (2) SAM (FULL-SHOT) and (3) SAM + LAD. The three columns correspond to the three post-dialog questions: (1) task completion, (2) asking all necessary information and (3) avoiding redundancy. Results in boldface are statistically significant by one-tailed t-test ($p < 0.05$).

results are obtained by training on $n - 1$ tasks (i.e., 23) and evaluating on the remaining task, repeated 24 times. In contrast, SAM + LAD observes **no human-written dialogs** whatsoever. Instead, SAM+LAD is trained only on the synthetic dialogs produced by LAD. In the zero-shot setting, SAM + LAD achieves an **+11.05** F-1 improvement over SAM. Furthermore, this result is only **6.02** points below the full-shot results of SAM. This significant gain further validates the claim of this work. SAM + LAD learns the necessary task-specific properties by leveraging the task specification as an inductive bias in LAD during the creation of synthetic data.

Interactive Human Evaluation

SAM + LAD achieves strong zero-shot results on the STAR corpus, especially relative to the performance of SAM (FULL-SHOT). This leads us to question the performance gap between these two models. Is the full-shot model better at next action prediction, or is it just better at modelling artifacts in the STAR corpus? STAR is known to have some degree of inconsistency with the policy graphs [Mosig et al., 2020]. Furthermore, static evaluation is not necessarily reflective of the performance of a model in **real settings**. Because of variable user behavior, there may be a distribution shift between the STAR corpus and interactive

settings. To this end, we perform an interactive human evaluation using Amazon Mechanical Turk (AMT).

Three models are evaluated: (1) SAM (ZERO-SHOT), (2) SAM (FULL-SHOT) and (3) SAM + LAD. Ten scenarios are defined, each of which consists of an objective (e.g, *'You want to plan a party'*) and slot values (e.g., `Name`: Kevin, `Date`: Sunday, `Num Guests`: 85). An AMT worker is instructed to interact with a dialog system according to the provided scenario. Upon completion of the dialog, three questions are answered:

1. Did the system successfully complete the dialog?
2. Did the system ask for all of the necessary information?
3. Did the system ask for information that you had already provided it?

The instructions tell the worker to interact *naturally* (e.g., by providing information out of turn). Detailed instructions, including examples and counter-examples, are provided for the three post-dialog questions. Pre-screening is performed to ensure that AMT workers read and understood these instructions. During pre-screening, the worker must answer the post-dialog questions given two completed dialogs and the corresponding scenarios. Workers with a score of at least 5/6 qualify to participate in the interactive evaluation (45% of workers pass the pre-screening). Pre-screening is paid $0.75USD, regardless of the result. Each HIT of the interactive evaluation includes five scenarios and pays $3.25USD (approx. 10 minutes). A post-hoc quality check is performed to remove erroneous annotations. Simple heuristics are constructed to predict the post-dialog answers and any discrepancies with the annotations are manually verified. If an error is identified through manual validation, the annotation is removed.

Tables 6.9 and 6.10 show the instructions for the pre-screening qualification task. Figures 6.4 and 6.5 show the pre-screening examples that the AMT workers answered the three-post dialog questions for. Workers qualified to complete the annotation HIT only if their pre-screening score was at least 5/6 correct and 2/2 on the completion question (as this was the most erroneous in preliminary experiments).

1628 dialogs were collected, with at least 500 for each system. The results, shown in Table 6.8, demonstrate that the performance of all three models is fairly similar in interactive settings. For the second post-dialog question, SAM+LAD asks for all of the necessary slots **+2.7%** more often. Assuming that the number of observations is equal to the total number of turns, this result is statistically significant ($p < 0.05$) by one-tailed t-test.

Both SAM (FULL-SHOT) and SAM (ZERO-SHOT) are trained on human dialogs, though the latter does not observe data from the target task. In contrast, SAM + LAD is trained only on synthetic data produced by LAD. Despite not observing any human dialogs, in interactive settings SAM + LAD attains zero-shot performance comparable to training on human dialogs from the STAR corpus. Though there remains significant room for improvement, the results of this interactive human evaluation demonstrate the efficacy of LAD. By using the task specification as an inductive bias and leveraging strengths of large language models to induce linguistic diversity, LAD produces synthetic data that effectively conveys the

This HIT is a pre-screening test for the Chatbot Evaluation HIT. Please read the instructions carefully and answer the questions below.
You will be qualified for the "Chatbot Evaluation" HIT, only if you correctly answer these pre-screening questions. You will be compensated $0.75 for completing this pre-screening test. The Chatbot Evaluation task will pay $3.25/HIT ($\tilde{1}$0 minutes/HIT, 600 total HITs), and will involve interacting with chatbots.

For each dialog, you must answer three questions. Please read the instructions below carefully, before proceeding to the questions:

Question 1: Did the system succesfully complete the dialog? This is asking whether the system successfully asked you all of the necessary information, understood your answers and performed a query ('[QUERY]') to find what you asked for. You should consider the dialog complete, even if the system indicated that they were unable to (for example) make you a reservation. A dialog might be incomplete, if (for example) the system fails to make a [QUERY] or repeatedly asks the same question.

Complete Dialog Example: This dialog is considered complete because the system succesfully understood your instructions, but the restaurant is unavailable:
User: make me a reservation at Lucca's for 5 people at 7pm
System: `[QUERY]`
System: `[RESULT] [RestaurantName] is unavailable.`
System: [RestaurantName] is unable to make a reservation at that time, would you like to find another restaurant?

Incomplete Dialog Example: This dialog is considered incomplete because the system fails to understand your instructions, even after multiple attempts:
User: make me a reservation at Lucca's for 5 people at 7pm
System: What is the name of the restaurant?
User: Lucca's
System: What is the name of the restaurant?
User: Lucca's
System: What is the name of the restaurant?
User: The name of the restaurant is Lucca's
System: What is the name of the restaurant?

Question 2: Did the system ask you for all of the necessary information? If the system asked you for all of the information in the scenario, the answer to this question should be Yes. If the system neglected to ask you information even though it was provided in the scenario, the answer to this question should be No.

Positive Example: In this dialog, the system received all of the information in the scenario before proceeding.
Scenario: Objective: Book a ride, Name: Sam, Departure Location: Union Grill, Destination Location: Central Park
User: i want a ride from Union Grill to Central Park
System: What is your name?
User: Sam
System: `[QUERY]`
System: `[RESULT] Departure=[DepartureLocation], Arrival=[ArrivalLocation], DriverName=[DriverName]`
System: Your ride is on its way.

Table 6.9: Instructions for the pre-screening task. Continued in Table 6.10

Negative Example: In this dialog, the system failed to ask for the user's name before make a query.

Scenario: Objective: Book a ride, Name: Sam, Departure Location: Union Grill, Destination Location: Central Park

User: i want a ride from Union Grill to Central Park

System: `[QUERY]`

System: `[RESULT] Departure=[DepartureLocation], Arrival=[ArrivalLocation], DriverName=[DriverName]`

System: Your ride is on its way.

Question 3: Did the system ask you for information that you had already provided it? If the system asked you for information that you had already provided, the answer to this question should be Yes. If the system did not ask for redundant information, the answer to this question should be No.

Positive Example: In this dialog, the system did not ask for information that you had already provided.

User: i want a ride from Union Grill to Central Park

System: What is your name?

User: Sam

System: `[QUERY]`

System: `[RESULT] Departure=[DepartureLocation], Arrival=[ArrivalLocation], DriverName=[DriverName]`

System: Your ride is on its way.

Negative Example: In this dialog, the system asked for the departure location, even though the user had already provided it.

User: i want a ride from Union Grill to Central Park

System: What is your name?

User: Sam

System: Where would you like to go?

User: Central Park

System: `[QUERY]`

System: `[RESULT] Departure=[DepartureLocation], Arrival=[ArrivalLocation], DriverName=[DriverName]`

System: Your ride is on its way.

This HIT is a pre-screening test for the "Chatbot Evaluation" HIT. Please read the instructions carefully and answer the questions below. Succesfully passing this pre-screening will qualify you to do the Chatbot Evaluation HIT ($3.25/HIT & approx 10min/HIT & approx 300 HITs).

Consider the scenario (left) and the dialog (right) in the image below. Answer the three questions according to the dialog.

Table 6.10: Continuation of the pre-screening instructions.

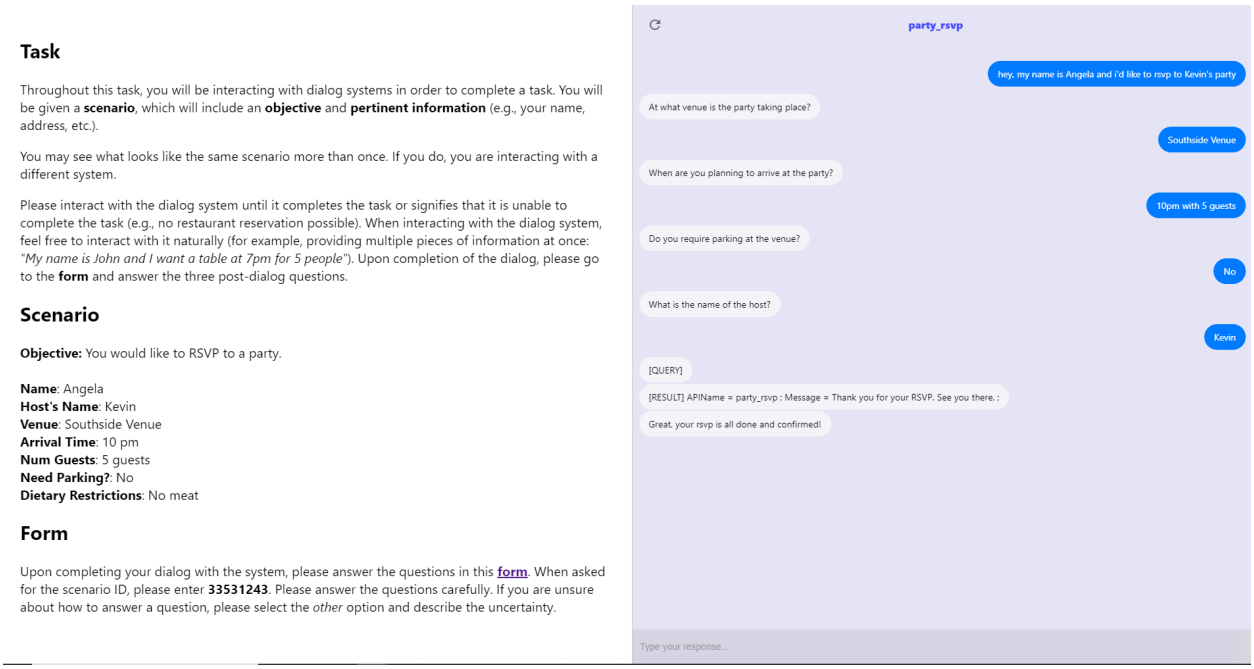


Figure 6.4: First example dialog that was used to evaluate AMT workers during the pre-screening. The correct answers for this example are: (1) the dialog is complete, (2) the system did not ask for all of the information (dietaray restrictions), (3) the system asked for redundant information (name of the host).

necessary task-specific properties and facilitates zero-shot generalization, even in challenging interactive settings.

Ablation

In an effort to gain insight into the performance of LAD, an ablation is carried out on next action prediction (Table 6.11). To ensure a fair comparison, we generate 1000 dialogs with each approach. With reduced data, the performance of SAM + LAD drops to **60.57**, which is still a strong improvement over prior work, without observing any human dialogs.

Removing the GPT-3 reformulation, i.e., using the seed data, results in an F-1 of 54.93. Removing the diversity of user behavior induced by Algorithm 1 (and keeping the GPT-3 reformulation) results in a score of 48.91. This suggests that the diversity of user behavior is more important for facilitating effective learning of the necessary task-specific properties (specifically the dialog policy) than the GPT-3 reformulation.

The last row of Table 6.11 shows the results of leveraging an alternate prompt for GPT-3. Instead of generating **five** reformulations, this prompt only generates a single reformulation at a time. The reduced performance of 57.76 demonstrates the positive impact of our specific prompt.

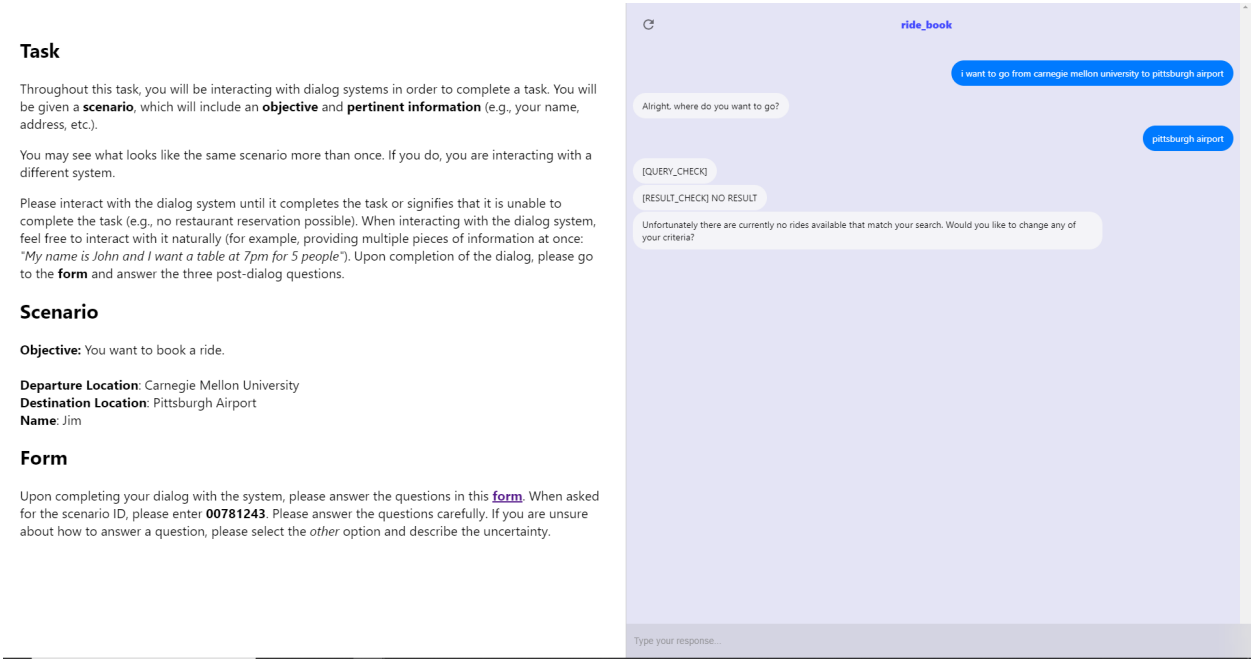


Figure 6.5: Second example dialog that was used to evaluate AMT workers during the pre-screening. The correct answers for this example are: (1) the dialog is complete, (2) the system did not ask for all of the information (name), (3) the system asked for redundant information (destination). The chosen slots in this image are sampled from the STAR corpus [Mosig et al., 2020], which is not necessarily the case for all of the scenarios.

Model	F-1
SAM + LAD	**60.57**
– *reformulation*	54.93
– *seed diversity*	48.91
– *seed diversity – reformulation*	48.00
– *five reformulation prompt*	57.76

Table 6.11: Ablation experiments on the STAR corpus. To ensure a fair comparison, 1000 synthetic dialogs are generated with each approach. SAM was trained for 40 epochs on each synthetic dataset.

6.4.3 Discussion

This work aims to further validate the use of the task specification as an inductive bias to facilitate zero-shot generalization to unseen tasks. The synthetic data created by LAD is (1) accurate due to the task specification being used to create the seed data and validate the reformulated utterances and (2) diverse because LAD leverages the strengths of large language models (i.e., knowledge of the distributional structure of language) to reformulate utterances in a domain-agnostic manner. By using the task specification as an inductive bias during the creation of synthetic data, LAD is able to effectively convey the necessary task-specific properties through the synthetic data. The significant performance gains in zero-shot settings across intent prediction, slot filling and next action prediction validate the efficacy of leveraging the task specification as an inductive bias during the creation of synthetic data.

6.5 Task Specification in the Model Architecture

The goal of Chapter 6 is to demonstrate that using the task specification as an inductive bias can facilitate zero-shot generalization to new tasks. Using the task specification as an input (Section 6.3: SAM) and for synthetic data creation (Section 6.4: LAD) has yielded significant gains in zero-shot intent prediction, slot filling and next action prediction. This section proposes to leverage the task specification as an inductive bias in the model architecture through Schema-Guided Fusion (SGF), which uses the task specification to explicitly define the relationship between different dialog components as a parameter in the model architecture. The motivation for SGF is threefold.

First, SGF aims to enforce adherence to the task-specific dialog policy by explicitly defining the relationship between intents, slots and actions in the model architecture. Whereas SAM predicts the next system action by simply attending to the policy graph, SGF first predicts the intents and slots covered by each utterance in the dialog history, and next determines which system actions have their *prerequisites* satisfied. By modelling the task of next action prediction in this manner, SGF will ensure that the resulting dialogs are complete and adherent to the task specification, regardless of variable user behavior (e.g., providing information out of turn).

Second, SGF leverages the strong performance of zero-shot intent prediction and slot filling facilitated by LAD, to further improve next action prediction. SGF considers each user utterance in the dialog history and predicts the user intents and provided slots. These predictions are then used, in combination with the policy matrix, to determine the appropriate system action according to the task specification. This process results in improved robustness and interpretability.

Third, by using the task specification to define the relationship between components, SGF facilitates *learning* of the task-specific dialog policy. Since the task specification is formulated as a parameter in the model architecture, this parameter can be randomly initialized and the task-specific dialog policy can be learned from data.

To validate the efficacy of using the task specification as an inductive bias in the model architecture, several experiments are carried out on the STAR corpus [Mosig et al., 2020]. SGF is shown to result in moderate performance gains in zero-shot generalization across a variety of experimental settings. Furthermore, SGF exhibits promising initial results when tasked with learning the task-specific dialog policy from noisy human dialogs. Finally, a human evaluation is performed, wherein individuals (1) define a novel task specification and (2) assess various systems that leverage the created task specification as an inductive bias.

6.5.1 Methods

SGF leverages the task specification as an inductive bias in the model architecture. Concretely, the task specification is used to define the *policy matrix*, a parameter in the model architecture that defines the relationship between the predicted intents/slots in the dialog history and the system actions. First, this section describes how the task specification is used to create the policy matrix. Next, the Schema Attention Model (SAM) is adapted for zero-shot intent prediction. Third, SGF is introduced, including (1) dialog components for intent prediction and slot filling, (2) the policy matrix, (3) the process of predicting the system action using the utterance-level intent/slot predictions and the policy matrix, and (4) loss functions. Finally, the process of learning the policy matrix from data is described.

Policy Matrix

The policy matrix defines the relationship between the user intents/slots in the dialog history and the system actions. Let M^p denote the policy matrix and v_i denote the binary vector representing the intents in the dialog history (s.t., v_i is of dimensions $|\mathcal{I}| \times 1$, where $\mathcal{I}$ is the set of intents) — $\mathrm{softmax}(M^p \cdot v_i)$ should produce a probability distribution over the set of system actions, such that the appropriate system action has the highest probability. To construct this policy matrix, we first consider each system node $\mathcal{V}_i^s$ in the task-specific policy graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. For each system node $\mathcal{V}_i^s$, let $\mathrm{paths}(\mathcal{V}_i^s)$ be the set of distinct paths through the policy graph starting from the initial node (i.e., `user-hello`) and ending at the particular system node. Each path $\mathcal{P}_j$ in $\mathcal{G}$ passes through a unique set of user nodes and therefore describes a unique set of user intents. As such, each $\mathcal{P}_j$ describes a unique set of *prerequisites* for a particular system action $\mathcal{A}_i$ (s.t., $\mathcal{A}_i$ is the corresponding action for system node $\mathcal{V}_i^s$). If all of the user intent prerequisites describes by

$\mathcal{P}_j$ are satisfied in the dialog history, then the appropriate system action is $\mathcal{A}_i$.

Consider all of the unique paths $\mathcal{P}$ described by a task specification. Assume that $\text{action}(\mathcal{P}_i)$ denotes the action corresponding to the system node that $\mathcal{P}_i$ ends at. Assume that $\text{intents}(\mathcal{P}_i)$ denotes the set of intents corresponding to the user nodes that $\mathcal{P}_i$ passes through. Given M^p, a matrix of dimensions $|\mathcal{P}| \times |\mathcal{I}|$ (i.e., a row for each unique path and a column for each intent), the values of M^p are defined as follows:

$$M^p_{i,j} = \begin{cases} 1 & \mathcal{I}_j \in \text{intents}(\mathcal{P}_i) \\ 0 & \text{otherwise.} \end{cases} \tag{6.10}$$

Though defining M^p in this manner will generally work in practice, it does have a key limitation. Consider the following example, wherein there are 4 possible intents and a linear policy graph. Assume that 3 out of the 4 intents have occurred (1,3,4) — $v_i = \langle 1, 0, 1, 1 \rangle$. Since the 2nd user intent has not been provided, the appropriate system action should be the system node directly preceding the 2nd user node. The row in M^p corresponding to the correct system action will be $\langle 1, 0, 0, 0 \rangle$. However, the row with the highest probability will be $\langle 1, 1, 1, 1 \rangle$. This example highlights a weakness of this definition: the absence of intents does not sufficiently influence the predicted action.

To address this issue, we make a simple modification to the definition of M^p that ensures that *all* the user intents described by a path must be present. This is done by concatenating M^p and $-\alpha M^p$, such that α is a scalar with value greater than or equal to the length of the maximum path. The new dimension of M^p is now $|\mathcal{P}| \times 2|\mathcal{I}|$. To determine the appropriate system action, we now perform: $M^p \cdot [v_i; (1 - v_i)]$. In the example provided above, the correct answer will be predicted. The value of the correct row in M^p will now be $\langle 1, 0, 0, 0, -4, 0, 0, 0 \rangle$, and the value of the incorrect row will be $\langle 1, 1, 1, 1, -4, -4, -4, -4 \rangle$. The value of $[v_i; (1 - v_i)]$ is $\langle 1, 0, 1, 1, 0, 1, 0, 0 \rangle$. Therefore, the score for the correct action will be 1 and the score for the previously predicted, incorrect action will be -1. In practice, setting α to be too high, results in worse performance because v_i is predicted by a zero-shot intent prediction model and is therefore prone to false negatives. As such, $\alpha = 2$ is used throughout the experiments, despite the theoretical disadvantage. Given stronger intent prediction models, higher α can be leveraged to better enforce adherence to the task specification. In certain experiments, the $|\mathcal{P}| \times |\mathcal{I}|$ dimension version of M^p is used. This is referred to as *half-policy*.

Schema Attention Model for Intent Prediction

The Schema Attention Model (Section 6.3) facilitates zero-shot next action prediction by using the task specification as an input. SAM predicts system actions by attending to the policy graph. SAM-I adapts SAM to facilitate zero-shot intent prediction using the task specification as the input. Two modifications are made to SAM. First, only the *last turn* of the dialog history (i.e., last system action and last user action) is used as the input. Next, to account for *multiple* intents being satisfied by a single utterance, binary cross entropy loss is leveraged. In this manner, SAM-I determines the intents that are satisfied by performing token-level attention between the last turn and the nodes of the policy graph.

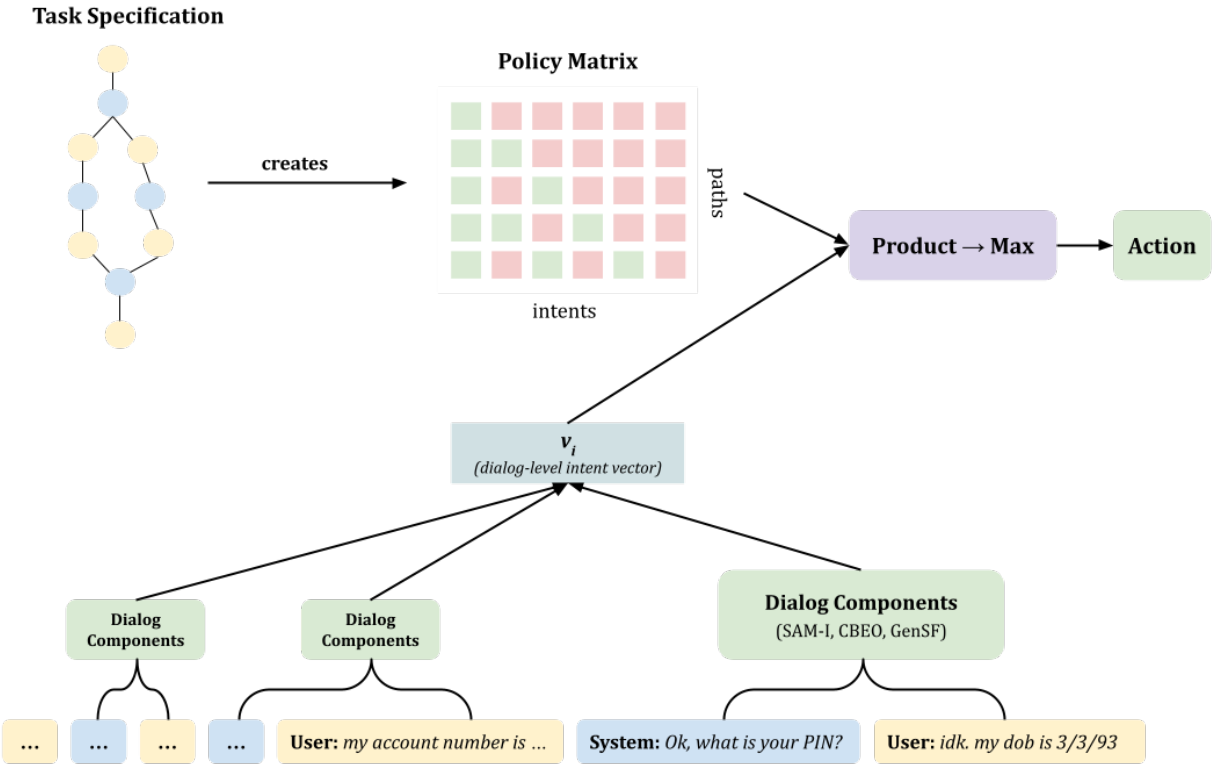


Figure 6.6: Visualization of Schema-Guided Fusion. The task specification is used to create the policy matrix by considering all of the different paths in the policy graph. Dialog components produce predictions at the turn level, which are then aggregated to produce v_i. For visual clarity, the interpolation with SAM is omitted from this image.

Schema-Guided Fusion

Schema-Guided Fusion (SGF) is introduced with the goal of leveraging the task specification as an inductive bias in the model architecture. The policy matrix, constructed by considering paths in the policy graph, is used as a parameter in the model architecture of a next action prediction model. SGF predicts the appropriate system action using a three-stage pipeline. First, pre-trained *dialog components* are leveraged to predict utterance-level intents and slots. These predictions are aggregated into a vector, v_i that represents the user intents in the dialog history. Next, the policy matrix is used to determine the system actions with *satisfied prerequisites*. Finally, the probability distribution produced by the policy matrix is interpolated with the output of SAM. Through this three-stage pipeline, SGF (1) leverages the strengths of zero-shot intent prediction and slot filling (described in Section 6.4), (2) leverages the task specification as an inductive bias in the model architecture through the policy matrix and (3) leverages the task specification as an input.

Throughout the experiments, three zero-shot dialog components are used. All of these models are trained with the synthetic data produced by LAD (Section 6.4). The three dialog components are: (1) ConvBERT + Example-Driven Training (CBEO – Section 4.3), (2) Generative Slot Filling (GENSF – Section 5.3)

and (3) Schema Attention Model (SAM-I). The two intent prediction models, CBEO and SAM-I, are both trained in the same manner with a binary cross entropy loss function. The key difference between the two models is that CBEO attends to the training examples while SAM-I attends to the policy graph. The intent prediction models consider each user utterance u_i in the dialog history and produce utterance-level intent probabilities, p_i^{intent}, such that $\forall j \in \mathcal{I}, 0 \leq p_{i,j}^{intent} \leq 1$. The utterance-level intent probabilities p_i^{intent} are scaled to ensure that all intent probabilities $p_{i,j}^{intent} > \frac{1}{2} \max p_i^{intent}$ has a value of at least 1. As such, to produce the set of intents in the dialog history, we aggregate over the scaled utterance-level intent probabilities: $v_i = \sum_{k \leq i} p_k^{intent} / 2 \max p_k^{intent}$. Though the intent prediction models are able to independently produce v_i, the slot predictions produced by GENSF are insufficient for this because not every user node has corresponding slots. As such, GENSF is only used in *combination* with CBEO and SAM-I. For each predicted slot in the dialog history, the corresponding intent (e.g., `user-inform-date` for slot `date`) has its value $v_{i,j}$ incremented.

Once v_i is constructed using the pre-trained dialog components, the policy matrix M^p. The probability distribution over the different paths in the graph is produced by softmax$(M^p \cdot [v_i; (1 - v_i)])$. To produce P_{pm}, the probability distribution over the system actions, the maximum probability for a particular action is used. Using $\mathcal{A}_k$ to denote the action for a particular path and P^p to denote the probabilities over paths, $P_{pm}(a) = \max_{k;\mathcal{A}_k=a} P_k^p$. In this manner, the task specification is being used as an inductive bias in the model architecture to define the relationship between the dialog components and the system actions. In this probability distribution, the system actions that have all of their prerequisites satisfied will have higher probabilities. If multiple actions have all of their prerequisites satisfied, the action with the *most* prerequisites will be favored. This process is visualized in Figure 6.6.

If the pre-trained dialog components were perfectly accurate, the probabilities produced by the aforementioned product would be sufficient. However, we are using zero-shot intent prediction and slot filling approaches and there may be errors. To ensure that errors in the predictions produced by the dialog component do not propagate to next action prediction, we combine the probabilities produced by the policy matrix and SAM. This is done by predicting a value p_{pm} (s.t., $0 \leq p_{pm} \leq 1$) and combining the log-probabilities (denoted as $log(P_{pm})$ and $log(P_{sam})$) of the two models: $p_{pm} log(P_{pm}) + (1 - p_{pm}) log(P_{sam})$. The value of p_{pm} is produced by a linear classifier which takes as input $log(P_{pm})$, $log(P_{sam})$ and h_i, where h_i represents the hidden state representation of the dialog history. The linear classifier which produces p_{pm} can potentially learn to account for common errors in the dialog components or default to SAM in particularly complex dialog histories (e.g., user repetition or looping).

Several loss functions are used in SGF. First, a cross entropy loss can potentially be applied in three places: (1) the predictions produced by the policy matrix, (2) the predictions produced by SAM and (3) the combined probability distribution. The latter two are done consistently across the experiments. The loss is only applied to the probabilities produced by the policy matrix if either the policy matrix or the dialog components are being fine-tuned. Second, a binary cross entropy loss can be applied to v_i when the dialog components are being fine-tuned. This loss is only used when SAM-I is being fine-tuned, to ensure that the intent probabilities remain accurate.

Policy Learning

The policy matrix is generally initialized according to the task specification, as described in the previous section. This policy matrix can be fine-tuned. However, in some experiments – the policy matrix is randomly initialized. In such cases, the policy matrix is over-parameterized (such that there are 1000 rows for each possible system action, rather than 1-2). This over-parameterization accounts for the fact that there may be multiple possible paths. When being learned from scratch, the policy matrix and forward pass resembles convolutional filters and max-pooling in CNNs. Each row of the policy matrix represents a unique set of prerequisites (i.e., path in the policy graph) for the corresponding system action.

6.5.2 Experiments

To ascertain the effectiveness of using the task specification as an inductive bias in the model architecture, several experiments are carried out for the task of next action prediction using the STAR corpus [Mosig et al., 2020]. First, experiments are performed with a fixed policy matrix and fixed dialog components. These experiments demonstrate that SGF is able to leverage zero-shot dialog components using the policy matrix defined according to the task specification. Next, SAM-I is fine-tuned to demonstrate that the policy matrix can be used to train the dialog components. Finally, two sets of experimental results are presented to demonstrate that SGF can learn a task-specific dialog policy.

Fixed Policy

Table 6.12 shows the results on the STAR corpus using both fixed dialog components and a fixed policy matrix which is defined according to the task specification. We show the results of using the pre-trained intent prediction models (SAM-I and CBEO) independently, as well as in combination with GENSF. Since all of the dialog components are trained on synthetic data produced by LAD, they produce very accurate predictions for the next action prediction training data (i.e., because the same dialogs were used for training). As such, for efficiency purposes and to avoid dealing with implementation details (e.g., ensuring that the set of examples for CBEO does not overlap with the training examples), the round-truth intent labels are used during training. At inference time, since the STAR corpus does not consist of any intent or slot labels, the predictions produced by the dialog components are used. SAM-I, which predicts intents by attending to the policy graph, is less likely (especially relative to CBEO) to over-fit on the training dialogs. As such, for SAM-I, we experiment with both using the ground-truth labels and the predictions during training.

The results in Table 6.12 demonstrate that SGF results in moderate performance gains over prior approaches. SGF with SAM-I and GENSF attains a **+1.18** improvement in F-1 over SAM+LAD and a **+2.52** improvement over the re-trained SAM+LAD. The re-trained model is slightly worse because the batch size is reduced (from 64 to 16) to maintain consistency with all of the SGF models. Though the effective batch size remains the same due with gradient accumulation, the number of negative examples from the policy graph is proportional to the batch size and thereby the reduction results in a small performance decrease.

Model	F-1
Previous Results	
BERT+S [Mosig et al., 2020]	28.12
SAM (Section 6.3)	53.31
SAM + LAD (Section 6.4)	64.36
New Results	
SAM + LAD (re-trained)	63.02
SGF + LAD + SAM-I (trained with ground-truth)	63.01
SGF + LAD + SAM-I (trained with SAM-I)	63.97
SGF + LAD + CBEO (trained with ground-truth)	63.39
SGF + LAD + SAM-I + GENSF (trained with ground-truth)	62.65
SGF + LAD + SAM-I + GENSF (trained with SAM-I)	**65.54**
SGF + LAD + CBEO + GENSF (trained with ground-truth)	64.87

Table 6.12: Experimental results on the STAR corpus. All results shown here are zero-shot. Methods trained with LAD have observed no human-written data, aside from the task specification.

Nonetheless, SGF improves over all previous methods, demonstrating the effectiveness of using the task specification as an inductive bias in the model architecture.

Using the predictions produced by SAM-I during consistently attains higher performance than using the ground-truth annotations. This is most likely because the prediction errors of SAM-I remain consistent between training and testing, and SGF learns to ignore the action probabilities produced by the policy matrix in certain contexts. Using the slot predictions produced by GENSF generally improves performance, demonstrating that SGF is effectively leveraging the strong performance of the zero-shot dialog components.

Fine-Tuning Intent Prediction

The policy matrix can be used to fine-tune the dialog components for the task of next action prediction. To demonstrate this, SAM-I is fine-tuned while SGF is being trained for next action prediction. The policy matrix is held fixed, however the gradient still flows to SAM-I during backpropagation through v_i. Two additional losses are leveraged: (1) a cross entropy loss which directly optimizes the probabilities that are produced by the policy matrix (i.e., before combination with SAM) and (2) a binary cross entropy loss on v_i that ensures the dialog-level intents are accurate. Table 6.13 shows four experimental settings: (1) the baseline SAM+LAD, (2) SGF + SAM-I without fine-tuning of SAM-I, (3) SGF + SAM-I wherein SAM-I is fine-tuned only with the binary cross entropy intent loss and (4) SGF+SAM-I wherein SAM-I is fine-tuned with both the binary cross entropy intent loss and with the next action prediction loss. To facilitate better gradient flow, the *half-policy* policy matrix is used when SAM-I is fine-tuned.

Table 6.13 shows that fine-tuning SAM-I results in moderate performance gains. Particularly, the dif-

Model	F-1
SAM + LAD (re-trained)	63.02
SGF + LAD + SAM-I (fixed SAM-I)	63.97
SGF + LAD + SAM-I (fine-tuned SAM-I, no backprop from action)	64.15
SGF + LAD + SAM-I (fine-tuned SAM-I)	**65.10**

Table 6.13: Experimental results on the STAR corpus demonstrating the performance gains from fine-tuning SAM-I during the training of SGF. For the setting where SAM-I is fine-tuned, the *half-policy* is used.

ference between the 3rd and 4th experimental setting (i.e., when SAM-I is fine-tuned with the action loss) validates the claim that SGF can facilitate improved performance in the dialog components.

Learned Policy

By representing the task-specific dialog policy as an explicit parameter in the model architecture, SGF facilitates learning the dialog policy from data. To demonstrate this, two sets of experiments are carried out. First, in Table 6.14 the policy is learned from the synthetic data produced by LAD. The resulting models perform similarly to the fixed policy experiments (Table 6.12) demonstrating that the policy has been effectively learned. Next, in Table 6.15, the policy is learned from noisy human-annotated dialogs that do not perfectly adhere to the task-specific policy.

Table 6.14 randomly initializes an over-parameterized policy matrix and fine-tunes it for the task of next action prediction. The resulting models perform slightly worse than the fixed policy experiments, however are still competitive. The results in this table signifies that the policy can effectively be learned from the synthetic dialogs produced by LAD which are perfectly adherent to the task-specific dialog policy.

However, learning the task-specific dialog policy from synthetic data is not particularly challenging. Especially when using ground-truth intent labels – learning a reasonable policy is trivial. Table 6.15 attempts to learn a meaningful task-specific dialog policy from the human-human dialogs in the STAR corpus, using the outputs of the task-specific dialog components. To facilitate learning of the policy matrix, the combination with the probabilities from SAM are removed. Furthermore, the half-policy policy matrix is learned. Table 6.15 shows the action prediction performance on a held-out test set from the STAR corpus, as well as the percentage of *paths* in the policy graph that the policy matrix correctly classifies. Using SAM-I and GENSF results in the highest recall with **59.06** of the paths in the policy graph being correctly detected.

While it would be ideal to produce an interpretable task-specific policy graph from the learned policy matrix — the nature of the learned policy matrix makes this difficult. The policy matrix learns to classify the correct action for a predicted v_i. There are no constraints on the policy matrix which ensure that well-formed sets of prerequisites are learned for each system action. Furthermore, because some intents never co-occur in the training data, it is impossible to identify what the correct behavior is for certain combinations of intents which impedes the creation of an interpretable policy graph. In addition to these problems, the over-

Model	F-1
Previous Results	
BERT+S [Mosig et al., 2020]	28.12
SAM (Section 6.3)	53.31
SAM + LAD (Section 6.4)	64.36
SGF + LAD + SAM-I + GENSF (trained with SAM-I)	**65.54**
SGF + LAD + CBEO + GENSF (trained with ground-truth)	64.87
Learned Policy Results	
SAM + LAD (re-trained)	63.02
SGF + LAD + SAM-I (trained with ground-truth)	63.53
SGF + LAD + SAM-I (trained with SAM-I)	64.31
SGF + LAD + CBEO (trained with ground-truth)	63.89
SGF + LAD + SAM-I + GENSF (trained with ground-truth)	63.87
SGF + LAD + SAM-I + GENSF (trained with SAM-I)	64.45
SGF + LAD + CBEO + GENSF (trained with ground-truth)	**64.57**

Table 6.14: Zero-shot experimental results on the STAR corpus wherein the policy matrix is randomly initialized and learned from data. All models are trained on data produced by LAD.

Model	F-1	Policy Path Recall
SGF + CBEO	61.15	50.27
SGF + SAM-I	45.60	45.61
SGF + CBEO + GENSF	62.51	54.39
SGF + SAM-I + GENSF	60.03	**59.06**

Table 6.15: Experimental results demonstrating the ability of SGF to learn the task-specific policy from noisy human-annotated dialogs. All models are trained with the STAR corpus (i.e., without LAD). The combination with SAM is removed for these experiments — as such, the probabilities produced are only from the policy matrix. A half-policy version of the policy matrix is used. All models are trained with the v_i produced by the dialog components (rather than ground-truth intents).

parameterization of the learned policy matrix and the potentially erroneous intent predictions also make this difficult. As such, any attempt at constructing the task-specific policy graph (i.e., the set of prerequisites for each action) from the learned policy matrix results in low precision. Future work should explore potential constraints or training formulations that facilitate better learning of the policy.

Your task is to create a task specification for a dialog system.

[Step 1] First, brainstorm a potential task-oriented dialog system that you might want to build. Be creative and unique (think of ideas related to your hobbies/interests). Your dialog system should have:

(1) At least 3 slots. A slot is a piece of information in an utterance (e.g., "I want a reservation at 5pm" –> slot: "time = 5pm"). Feel free to have as many slots as you'd like!
(2) At least 1 form of dialog flow logic (e.g., branching or looping).
(3) An API request. You don't need to implement the API, but the dialog flow should include at least one request to an API. The "decision making" part of the system is generally contained in the API (e.g., if you're building a movie recommender dialog system – the recommender functionality can be put into the API).

[Step 2] Describe the desired functionality for your dialog system. Make sure your description is sufficiently detailed and provides an overview of the (1) slots, (2) user intents, (3) system actions, (4) dialog flow logic, (5) API request.

[Step 3] Working off the provided example (https://pastebin.com/jJqjTmcs), construct a task specification for your dialog system. Ensure that the system behavior is deterministic (i.e., each user node only points to a single system node – however system nodes can point to multiple user nodes). Feel free to draw the graph out on paper if it helps.

Table 6.16: Instructions for the creation of the task specification.

6.5.3 Human Evaluation

Throughout this chapter, several approaches are presented for leveraging the task specification as an inductive bias in order to facilitate zero-shot generalization to new tasks. To validate the effectiveness of the proposed methods (i.e., SAM, LAD and SGF), a human evaluation is carried out.

Three individuals are tasked with (1) constructing a task specification for a new dialog task and (2) evaluating three zero-shot dialog systems in an interactive setting. In the first stage of this evaluation, each individual is instructed to brainstorm a potential dialog application and design a task specification. In the second stage, which occurred the next day, each individual had 5 conversations with three different dialog systems. They assessed the quality of each dialog and provided a relative ranking of the different systems.

Creating A Task Specification

The first stage of this human evaluation required each annotator to brainstorm a potential dialog system and create the task specification. The instructions are shown in Table 6.16. During the creation of the task specification, annotators were allowed to ask clarification questions about the requirements of the task specification. All of the annotators have a background in computer science. Two of the three annotators have no experience with building dialog systems.

The task specifications produced by each of the three annotators are shown below. The time it took an annotator create a task specification was tracked and is also indicated below. On average, it took the three annotators 14:18 minutes to create the task specifications. A well-designed user interface could potentially reduce this time further as it would reduce confusion pertaining to the adjacency list representation of the policy graph.

Annotator 1 created a dialog system which aims to suggest an appropriate weapon to purchase in the video game Valorant. The dialog system asks different questions depending on the number of credits a user has. This task specification took Annotator 1 a total of 21:03 minutes to create. The task specification is shown below.

```
{
"task": "weapon_recommendation",
"slots": {
    "[AgentName]": ["Viper", "Yoru", "Chamber", "Killjoy", "Phoenix"],
    "[Rank]": ["Iron", "Bronze", "Silver", "Gold", "Platinum",
               "Diamond","Immortal", "Radiant"],
    "[GunName]": ["Specter", "Judge", "Phantom", "Vandal"]
},
"utterances": {
    "user_hello": "Hello.",
    "hello": "Hello, how can I help?",
    "user_ask_recommendation": "Could I get a Valorant gunrecommendation?",
    "ask_agent": "What agent are you currently playing?",
    "user_inform_agent": "The agent I am playing is [AgentName]",
    "ask_rank": "What rank are you?",
    "user_inform_rank": "This season I am [Rank]",
    "ask_skins": "What guns do you have skins for?",
    "user_inform_skins": "I have skins for [GunName], [GunName], and [GunName]",
    "ask_credits": "Do you have below 1400, between 1400 and 3900, or
                    above 3900 credits?",
    "user_inform_below": "I have below 1400 credits this round",
    "user_inform_between": "I have between 1400 and 3900 credits this round",
    "user_inform_above": "I have above 3900 credits this round",
    "inform_below": "You can only afford a pistol this round",
    "ask_style": "Do you prefer precision or firing rate?",
    "user_inform_precision": "I prefer precision",
    "user_inform_firing_rate": "I prefer faster firing rate",
    "ask_shotgun": "Would you like to use a shotgun?",
    "user_inform_yes": "Yes I would prefer to use a shotgun",
    "user_inform_no": "No I would prefer not to use a shotgun",
    "query": "[QUERY]",
    "db_inform_recommendation": "[RESULT] APIName = weapon_recommendation;
```

```
                                    GunName = [GunName]",
    "inform_recommendation": "The recommended gun for you is [GunName]",
    "user_goodbye": "Alright, thanks goodbye",
    "system_goodbye": "Goodbye"
},
"graph": {
    "user_hello": ["hello"],
    "hello": ["user_ask_recommendation"],
    "user_ask_recommendation": ["ask_agent"],
    "ask_agent": ["user_inform_agent"],
    "user_inform_agent": ["ask_rank"],
    "ask_rank": ["user_inform_rank"],
    "user_inform_rank": ["ask_skins"],
    "ask_skins": ["user_inform_skins"],
    "user_inform_skins": ["ask_credits"],
    "ask_credits": ["user_inform_below", "user_inform_between","user_inform_above"],
    "user_inform_below": ["inform_below"],
    "user_inform_between": ["ask_shotgun"],
    "user_inform_above": ["ask_style"],
    "inform_below": ["user_goodbye"],
    "ask_shotgun": ["user_inform_yes", "user_inform_no"],
    "user_inform_yes": ["query"],
    "user_inform_no": ["query"],
    "ask_style": ["user_inform_precision", "user_inform_firing_rate"],
    "user_inform_precision": ["query"],
    "user_inform_firing_rate": ["query"],
    "query": ["db_inform_recommendation"],
    "db_inform_recommendation": ["inform_recommendation"],
    "inform_recommendation": ["user_goodbye"],
    "user_goodbye": ["system_goodbye"]
}
}
```

Annotator 2 created a dialog system that aims to coach players for the game League of Legends. This dialog system asks several generic questions, and asks specific follow-up questions depending on whether the user wants to focus on improving their micro or macro gameplay. Annotator 2 took 11:38 minutes to create the task specification shown below:

```
{
"task": "lol_coach",
"slots": {
    "[Username]": ["abaf183857", "epicgamer420", "player won", "jl917", "rayxero"],
    "[Gametime]": ["yesterday at noon", "yesterday at 1pm",
```

```
                    "yesterday at 1", "first game 7 days ago",
                    "the last game I played", "the second last game I played",
                    "april 24, 2020 at 1:30", "monday at 5pm"],
    "[AveragePing]": ["240", "24 milliseconds", "100", "10ms", "one second"],
    "[CurrentRank]": ["iron", "bronze", "bronze 3", "gold", "challenger",
                      "challenger 150lp", "150lp", "grandmaster", "plat 4",
                      "plat iii"]
},
"utterances": {
    "user_hello": "Hello.",
    "ask_username": "Hello, I am a robot coach. What is your username?",
        "user_inform_username": "My username is [Username].",
        "ask_gametime": "When was the game that you want to review?",
        "user_inform_gametime": "The time of the game is [Gametime]",
        "ask_intent": "What do you want me to focus on for this game?",
        "user_inform_macro": "I want to focus on my macro.",
        "user_inform_micro": "I want to focus on my micro.",
        "ask_avg_ping": "What is your typical ping?",
        "user_inform_ping": "My ping is [AveragePing].",
        "ask_current_rank": "What is your current rank?",
        "user_inform_rank": "My rank is [CurrentRank].",
        "query": "[QUERY]",
        "db_inform_analysis": "[RESULT] APIName = lolcoach ; Message = [Message]",
        "give_result": "[Message]. Goodbye!"
},
"graph": {
    "user_hello": ["ask_username"],
    "ask_username": ["user_inform_username"],
    "user_inform_username": ["ask_gametime"],
    "ask_gametime": ["user_inform_gametime"],
    "user_inform_gametime": ["ask_intent"],
    "ask_intent": ["user_inform_micro", "user_inform_macro"],
    "user_inform_micro": ["ask_avg_ping"],
    "user_inform_macro": ["ask_current_rank"],
    "ask_avg_ping": ["user_inform_ping"],
    "ask_current_rank": ["user_inform_rank"],
    "user_inform_ping": ["query"],
    "user_inform_rank": ["query"],
    "query": ["api_inform_analysis"],
    "db_inform_analysis": ["give_result"],
}
}
```

Annotator 3 created a dialog system to suggest a new chess opening to learn. The system asks several questions, and branches depending on whether the user wants a chess opening for the white or the black pieces. Annotator 3 created this task specification in 10:30 minutes.

```
{
"task": "chess_opening_recommender",
"slots": {
    "[Elo]": ["100", "667", "800", "1200", "1800", "2500"],
    "[Theory]": ["a little bit", "lots", "not a lot", "medium amount", "any amount"],
    "[PositionalTactical]": ["position", "positional", "tactics", "tactical"],
    "[ClassicHypermodern]": ["classic", "classical", "modern", "hypermodern"]
},
"utterances": {
    "user_hello": "Hello.",
    "hello": "Hello, how can I help?",
    "user_chess_opening_recommend": "Can you recommend me a chess opening?",
    "ask_elo": "Could I get your current chess elo, please?",
    "user_elo": "My current elo is [Elo].",
    "ask_theory": "Do you want an opening with little or lots of theory to memorize?",
    "user_theory": "I would like an opening with [Theory] of theory.",
    "ask_colour": "Great. Which colour would you want this opening to be for?",
    "user_colour_white": "I'd like this chess opening to be for the white pieces.",
    "user_colour_black": "I'd like this chess opening to be for the black pieces.",
    "ask_positional_tactical": "Do you want a positional opening or a
                                tactical opening?",
    "user_positional_tactical": "I would like a [PositionalTactical]
                                 opening.",
    "ask_classic_hypermodern": "Would you like a classic style or
                                hypermodern opening?",
    "user_classic_hypermodern": "I would like a [ClassicHypermodern] opening.",
    "query": "[QUERY]",
    "db_recommend_opening": "[RESULT] APIName = chess_opening_recommender ;
                             RecommendedOpenings = [Openings];",
    "inform_opening": "Here are some recommended openings: [Openings].",
    "user_thanks": "Thank you.",
    "anything_else": "Is there anything else that I can do for you?",
    "user_nothing_else": "No, I don't need anything else.",
    "goodbye_1": "Thank you and goodbye."
},
"graph": {
    "user_hello": ["hello"],
    "hello": ["user_chess_opening_recommend"],
    "user_chess_opening_recommend": ["ask_elo"],
```

```
    "ask_elo": ["user_elo"],
    "user_elo": ["ask_theory"],
    "ask_theory": ["user_theory"],
    "user_theory": ["ask_colour"],
    "ask_colour": ["user_colour_white", "user_colour_black"],
    "user_colour_white": ["ask_positional_tactical"],
    "ask_positional_tactical": ["user_positional_tactical"],
    "user_colour_black": ["ask_classic_hypermodern"],
    "ask_classic_hypermodern": ["user_classic_hypermodern"],
    "user_positional_tactical": ["query"],
    "user_classic_hypermodern": ["query"],
    "query": ["db_recommend_opening"],
    "db_recommend_opening": ["inform_opening"],
    "inform_opening": ["user_thanks"],
    "user_thanks": ["anything_else"],
    "anything_else": ["user_nothing_else"],
    "user_nothing_else": ["goodbye_1"]
}
```

The set of task specifications (summary statistics in Table 6.17) produced by the three human annotators cover domains that are significantly different from those in the STAR corpus. As such, this human evaluation is a realistic and challenging test of the zero-shot generalization abilities of the methods proposed throughout this chapter.

Interactive Evaluation

In the second stage of this human evaluation, the three annotators interact with three different dialog systems. The three systems evaluated are: (1) SAM (Section 6.3), (2) SAM+LAD (Section 6.4) and (3) SGF+LAD+SAM-I+GENSF (Section 6.5). For the latter two systems, LAD is used to generate 1000 dialogs for each task specification. SAM is trained on the STAR corpus and generalizes to the new tasks by using the task specification as an input. SAM+LAD generates synthetic data with LAD and uses it to train SAM. SGF+LAD+SAM-I+GENSF uses the task specification as an input (through SAM and SAM-I), for synthetic data creation (through LAD) and in the model architecture (through the policy matrix used in SGF). The third system also leverages GENSF, which uses an inductive bias in the problem formulation to improve zero-shot slot filling.

Annotators are instructed to have five dialogs with each of the three systems. The systems are presented to them in random order. Annotators are instructed to maintain the difficulty of the dialogs consistent across the three different systems. The task instructions and interface are identical to the interactive human evaluation conducted in Section 6.4. Three questions are asked following each dialog:

1. Did the system successfully complete the dialog?

Task	Time Taken	Num User Intents	Num Slots
Annotator 1: Valorant Weapon Recommendation	21:03	13	3
Annotator 2: League of Legends Coach	11:38	8	4
Annotator 3: Chess Opening Recommendation	10:13	10	4

Table 6.17: Summary of the task specifications created by each of the three human annotators.

Annotator 1 (Valorant Weapon Recommendation). Relative ranking: SAM+LAD > SGF > > SAM			
Model	**COMPLETE %**	**ASKS ALL %**	**AVOIDS REDUNDANCY %**
SAM	80	60	20
SAM+LAD	100	100	80
SGF+LAD+SAM-I+GENSF	80	100	80
Annotator 2 (League of Legends Coach). Relative ranking: SGF > SAM+LAD > SAM			
Model	**COMPLETE %**	**ASKS ALL %**	**AVOIDS REDUNDANCY %**
SAM	100	40	0
SAM+LAD	100	40	60
SGF+LAD+SAM-I+GENSF	100	60	80
Annotator 3 (Chess Opening Recommendation). Relative ranking: SGF > SAM+LAD > SAM			
Model	**COMPLETE %**	**ASKS ALL %**	**AVOIDS REDUNDANCY %**
SAM	100	0	60
SAM+LAD	100	40	100
SGF+LAD+SAM-I+GENSF	100	80	100

Table 6.18: Results of the interactive human evaluation.

2. Did the system ask for all of the necessary information?

3. Did the system ask for information that you had already provided it?

After interacting with the three systems, each annotator is asked to provide a relative ranking of the three systems. The results of both the dialog-level and system-level interactive evaluation for each annotator are shown in Table 6.18. It is evident from these results, particularly the relative rankings, that each proposed method in this chapter further improves zero-shot generalization to unseen tasks. All three annotators agree that SAM (i.e., only using the task specification as an input) is the worst of the three systems. Two of the three annotators prefer SGF above SAM+LAD. SGF is also shown to perform best according to the dialog-level results. Example dialogs for each of the three tasks are shown in Tables 6.19, 6.20 and 6.21.

System: SAM **Annotation:** **Incomplete**, **Did Not Ask All**, **Redundant**
User: hi **System:** Hello, how can I help? **User:** can i get a gun recommendation? i have skins for the phantom and vandal, and i have between 1400 and 3900 credits **System:** Do you have below 1400, between 1400 and 3900, or above 3900 credits? **User:** between 1400 and 3900 **System:** Do you prefer precision or firing rate? **User:** firing rate **System:** [QUERY] **DB:** [RESULT] APIName = weapon_recommendation ; GunName = [GunName] **System:** The recommended gun for you is [GunName].
System: SAM+LAD **Annotation:** **Complete**, **Asked All**, **Avoided Redundancy**
User: hi **System:** Hello, how can I help? **User:** could i get a gun recommendation? **System:** What agent are you currently playing? **User:** i'm on viper and i have above 3900 credits **System:** What rank are you? **User:** i'm gold and i have skins for the vandal **System:** Do you prefer precision or firing rate? **User:** probably precision **System:** [QUERY] **DB:** [RESULT] APIName = weapon_recommendation ; GunName = [GunName] **System:** The recommended gun for you is [GunName].
System: SGF+LAD+SAM-I+GENSF **Annotation:** **Complete**, **Asked All**, **Avoided Redundancy**
User: Hi **System:** Hello, how can I help? **User:** could i get a gun recommendation? i'm playing viper and i want to use a shotgun **System:** What rank are you? **User:** i'm gold and i have between 1400 and 3900 credits **System:** What guns do you have skins for? **User:** phantom **System:** [QUERY] **DB:** [RESULT] APIName = weapon_recommendation ; GunName = [GunName] **System:** The recommended gun for you is [GunName].

Table 6.19: Example interactive dialogs by Annotator 1 for the task of Valorant weapon recommendation.

System: SAM **Annotation:** **Complete**, **Did Not Ask All**, **Redundant**
User: hi I need help with gold macro for a game yesterday **System:** What is your typical ping? **User:** 150 ms **System:** What is your typical ping? **User:** 150 ms **System:** What is your typical ping? **User:** 150 **System:** What is your typical ping? **User:** one hundred **System:** [QUERY] **DB:** [RESULT] APIName = lolcoach ; Message = [Message] **System:** [Message]. Goodbye!
System: SAM+LAD **Annotation:** **Complete**, **Did Not Ask All**, **Avoided Redundancy**
User: hi my username is jl917, I want help with a game yesterday **System:** What do you want me to focus on for this game? **User:** macro **System:** What is your current rank? **User:** gold **System:** [QUERY] **DB:** [RESULT] APIName = lolcoach ; Message = [Message] **System:** [Message]. Goodbye!
System: SGF+LAD+SAM-I+GENSF **Annotation:** **Complete**, **Did Not Ask All**, **Avoided Redundancy**
User: hi i had a bad game yesterday at 1 **System:** What do you want me to focus on for this game? **User:** macro **System:** What is your current rank? **User:** gold **System:** [QUERY] **DB:** [RESULT] APIName = lolcoach ; Message = [Message] **System:** [Message]. Goodbye!

Table 6.20: Example interactive dialogs by Annotator 2 for the task of League of Legends coaching.

System: SAM **Annotation:** **Complete**, **Did Not Ask All**, **Avoided Redundancy**
User: I want a chess opening for White pieces **System:** Do you want a positional opening or a tactical opening? **User:** maybe tactical, I do tactics training every day :D **System:** [QUERY] **DB:** [RESULT] APIName = chess_opening_recommender ; RecommendedOpenings = [Openings]; **System:** Here are some recommended openings: [Openings].
System: SAM+LAD **Annotation:** **Complete**, **Asked All**, **Avoided Redundancy**
User: hi **System:** Hello, how can I help? **User:** ches open **System:** Could I get your current chess elo, please? **User:** 1400 but true skill is more like 1800 if you know what I mean **System:** Do you want an opening with little or lots of theory to memorize? **User:** none **System:** Great. Which colour would you want this opening to be for? **User:** blac **System:** Would you like a classic style or hypermodern opening? **User:** hypmodem **System:** [QUERY] **DB:** [RESULT] APIName = chess_opening_recommender ; RecommendedOpenings = [Openings]; **System:** Here are some recommended openings: [Openings].
System: SGF+LAD+SAM-I+GENSF **Annotation:** **Complete**, **Asked All**, **Avoided Redundancy**
User: I want a positional hypermodern opening for the black pieces at around 1674 elo **System:** Do you want an opening with little or lots of theory to memorize? **User:** Lots! **System:** [QUERY] **DB:** [RESULT] APIName = chess_opening_recommender ; RecommendedOpenings = [Openings]; **System:** Here are some recommended openings: [Openings].

Table 6.21: Example interactive dialogs by Annotator 3 for the task of chess opening recommendation.

6.5.4 Discussion

This section demonstrates the effectiveness of using the task specification as an inductive bias in the model architecture. Schema-Guided Fusion (SGF) leverages the task specification construct a policy matrix, which explicitly defines the relationship between pre-trained dialog components and system actions. Through experiments on the STAR corpus, SGF is shown to attain moderate improvements in zero-shot generalization by leveraging the policy matrix and zero-shot intent prediction and slot filling models. Furthermore, SGF is shown to be valuable as a mechanism for fine-tuning intent prediction models for the task of next action prediction. By randomly initializing the policy matrix, SGF is able to effectively learn the task-specific dialog policy from both synthetic data produced by LAD and from noisy human-human dialogs. These experiments validate the primary claims of this section. By using the task specification as an inductive bias in the model architecture, SGF (1) facilitates stronger adherence to the task specification, (2) successfully leverages the strengths of zero-shot intent prediction and slot filling models to improve next action prediction and (3) demonstrates promising results in learning the policy from data.

Human evaluation is conducted with the goal of assessing SGF, and more broadly the use of the task specification as an inductive bias. Human annotators are tasked with (1) creating a novel task specification and (2) interacting with multiple dialog systems which leverage the constructed specification. The results of this human evaluation demonstrate effective zero-shot generalization to unseen and unforeseen tasks. The strong performance of SGF throughout this human evaluation highlights the effectiveness of using the task specification as an inductive bias in the model architecture.

6.6 Conclusion

This chapter studies the long-standing and challenging problem of zero-shot generalization to new tasks. In order to facilitate this class of generalization, the *task specification* is used as an inductive bias. The task specification aims to define a task, particularly the task-specific dialog policy and the dialog ontology. The task specification is a minimal expression of the task-specific properties and this chapter argues that the task specification is a necessary condition for zero-shot generalization. Through three different proposed methods, this chapter demonstrates using the task specification can effectively facilitate zero-shot generalization.

First, the Schema Attention Model (SAM) uses the task specification as an input. The appropriate system action action is predicted by attending between the dialog history and the task-specific dialog policy. SAM achieves significant performance gains in zero-shot generalization on the STAR corpus over baseline models. Second, Language Models as Data (LAD) uses the task specification as an inductive bias during synthetic data creation. LAD leverages the strengths of large language models, in combination with the task specification, to generate *diverse* and *accurate* synthetic data. By using the task specification as an inductive bias, this synthetic data effectively conveys the necessary task-specific properties to a downstream dialog model. Through the synthetic data produced by LAD, significant gains are induced in zero-shot intent prediction, slot filling and next action prediction. Finally, Schema-Guided Fusion (SGF) uses the task speci-

fication as an inductive bias in the model architecture. SGF uses the task specification to create a parameter in the model architecture, which explicitly defines the relationship between intent prediction, slot filling and next action prediction. In this manner, SGF is shown to (1) improve adherence to the task specification, (2) effectively leverage zero-shot intent and slot prediction models and (3) facilitate learning the task-specific dialog policy from human-human dialogs.

To further validate the effectiveness of using the task specification as an inductive bias for inducing zero-shot generalization, two human evaluation studies are performed. In Section 6.4, over 1500 interactive dialogs are carried out to assess the performance of SAM and LAD. In Section 6.5, human annotators create novel task specifications and later interact with dialog systems that are conditioned on the specifications. These human evaluations highlight the effectiveness of the three proposed approaches for leveraging the task specification as an inductive bias.

This chapter progresses the state of zero-shot generalization to unseen tasks by using the task specification as an inductive bias through SAM (as an input), LAD (for synthetic data creation) and SGF (in the model architecture). The inductive biases introduced in this chapter disentangle the task-specific and task-agnostic abstractions in data-driven models of dialog and succesfully facilitate zero-shot generalization to unseen and unforeseen tasks.

Chapter 7

Conclusion

Generalization is imperative in dialog research. No finite dialog corpus can contain every phenomenon necessary for modelling dialog due to the complexity of human communication, the compositional nature of language, the vast number of possible tasks and the constantly evolving nature of these tasks. Given ample training examples, data-driven models can effectively learn specific tasks in constrained settings. However, such models struggle to generalize to unseen and unforeseen phenomena.

There are four classes of generalization necessary in dialog modelling: (1) generalization to new inputs, (2) generalization to new problems, (3) generalization to new outputs and (4) generalization to new tasks. This thesis studies these four classes of generalization across a variety of different dialog problems, including intent prediction, slot filling, dialog evaluation, response generation and next action prediction. To facilitate these classes of generalization, it is imperative to *prescribe* specific learned abstractions and behaviors in data-driven models. This is done through **inductive biases** which define the set of assumptions used by an intelligent system to predict outputs for unobserved inputs, i.e., to generalize.

Inductive biases are motivated by *knowledge* of the domain, the problem, the models and the desired generalization. An inductive bias allows a system developer, with knowledge, to modify a component of a data-driven model in order to prescribe specific behavior and thereby induce a desired generalization. In an effort to facilitate the aforementioned classes of generalization, this thesis studies four types of inductive biases: (Chapter 3) through self-supervised training, (Chapter 4) inductive biases in the model architecture, (Chapter 5) inductive biases in the problem formulation and (Chapter 6) the task specification as an inductive bias. Across nine different studies, inductive biases (motivated by knowledge of the domains, problems, models and desired generalizations) are constructed to prescribe specific learned abstractions in data-driven models and thereby facilitate generalization.

Chapter 3 demonstrates how the choice of self-supervised training data and self-supervised training objectives can be used to facilitate generalization to new inputs, to new outputs and to new problems. CONVBERT + MLM leverages task-adaptive self-supervised training to adapt pre-trained models to specific task-oriented dialog domains, facilitating strong performance gains across multiple problems in the DialoGLUE benchmark. USR uses specific self-supervised training objectives to approximate different dialog

qualities, and achieves stronger correlations with human judgements of dialog quality than existing dialog evaluation metrics. These two studies demonstrate how well-motivated inductive biases in self-supervised training can facilitate generalization.

Chapter 4 studies inductive biases in the model architecture. Structured Fusion Networks incorporate the traditional dialog pipeline into neural models of dialog. This inductive bias in the model architecture prescribes specific behavior which results in better generalization to new inputs/domains and improved robustness. Example-Driven Training presents a new architecture and training algorithm for intent prediction which predicts intents by attending to examples. This inductive bias makes the representation-to-intent mapping an explicit non-parametric process rather than an implicitly learned set of weights in a classification layer. As such, the resulting model is able to generalize to new outputs and can predict unseen intents, given the corresponding examples. This inductive bias further maintains consistency with the capabilities of the pre-trained model, thereby avoiding catastrophic forgetting and facilitating generalization to new domains and new outputs without any additional training. In these two studies, specific behavior is prescribed by incorporating well-motivated inductive biases into the model architecture. This specific behavior (e.g., making the representation-to-intent mapping a non-parametric process) facilitates generalization.

Chapter 5 explores inductive biases in the problem formulation as a means of achieving stronger alignment between the requirements of a downstream problem and the pre-existing capabilities of a pre-trained model. By reformulating both dialog evaluation and slot filling as response generation, the implicitly learned abilities of DialoGPT are effectively leveraged to induce zero-shot generalization. FED evaluates eighteen different qualities of dialog without any training whatsoever, by evaluating the likelihood of various follow-up utterances using DialoGPT. Through this inductive bias in the problem formulation, multiple dialog qualities are measured in a zero-shot setting. GENSF similarly reformulates slot filling as a response generation task, in addition to incorporating inductive biases into the model architecture. In this manner, GENSF achieves a strong alignment between the capabilities of DialoGPT (e.g., implicitly learned notion of certain slots) and the requirements of the slot filling task – thereby achieving significant performance gains in zero-shot slot filling. These two studies demonstrate the effectiveness of inductive biases in the problem formulation as a means of inducing zero-shot generalization.

Chapter 6 studies the fourth and most challenging class of generalization in dialog: generalization to new tasks. Zero-shot generalization to unseen tasks is a long-standing challenge that is central to the notion of generalization in dialog and is particularly desirable in practical applications. Given a dialog model which has been trained on a set of tasks, this class of generalizations aims to leverage this model for an unseen task *without any additional training data*. A data-driven model does not and can not have any notion of the task-specific dialog policy and dialog ontology for an unseen task. As such, this thesis uses the *task specification* as an inductive bias. The task specification is a minimal expression of the necessary task-specific properties (i.e., dialog policy and ontology) for a particular dialog task. The task specification is necessary for generalization. To demonstrate that the task specification is sufficient for inducing zero-shot generalization, three studies are carried out. The Schema Attention Model (SAM) uses the task specification as an input, and attends to a task-specific policy graph in order to predict system actions. Through this inductive bias,

SAM attains significant gains in zero-shot task transfer and domain transfer on the STAR corpus. Language Models as Data (LAD) extends the use of the task specification as an inductive bias, by using it to create synthetic data. In combination with large language models, this inductive bias facilitates the creation of accurate and diverse synthetic data, which effectively conveys the necessary task-specific properties to a downstream dialog model. LAD facilitates improved zero-shot generalization for intent prediction, slot filling and next action prediction. Next, Schema-Guided Fusion (SGF) uses the task specification as an inductive bias in the model architecture. SGF uses the task specification to construct a policy matrix, which explicitly defines the relationship between intent prediction, slot filling and next action prediction. Through this inductive bias, SGF (1) better enforces adherence to the task specification, (2) leverages zero-shot intent prediction and slot filling models to improve next action prediction and (3) facilitates *learning* of the dialog policy. Across these three studies, using the task specification as an inductive bias facilitates significant gains in zero-shot generalization. The three different next action prediction models are evaluated in interactive settings and with new task specifications — demonstrating effective zero-shot generalization.

This thesis studies four classes of generalization in dialog (new inputs, new outputs, new problems and new tasks) and presents four types of inductive biases (self-supervised training, in the model architecture, in the problem formulation and the task specification). Inductive biases, motivated by knowledge, are used to prescribe specific learned abstractions and behavior in data-driven models of dialog, thereby inducing the desired generalizations.

List of Figures

List of Tables

Bibliography

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. *arXiv preprint arXiv:1707.02363*.

Aditya Bhargava, Asli Celikyilmaz, Dilek Hakkani-Tür, and Ruhi Sarikaya. 2013. Easy contextual intent prediction and slot detection. In *2013 ieee international conference on acoustics, speech and signal processing*, pages 8337–8341. IEEE.

Dan Bohus and Alexander I Rudnicky. 2009. The ravenclaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Paweł Budzianowski and Ivan Vulić. 2019. Hello, it's gpt-2–how can i help you? towards the use of pretrained language models for task-oriented dialogue systems. *arXiv preprint arXiv:1907.05774*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Inigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*.

Giuseppe Castellucci, Valentina Bellomaria, Andrea Favalli, and Raniero Romagnoli. 2019. Multi-lingual intent detection and slot filling in a joint bert-based model. *arXiv preprint arXiv:1907.02884*.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019a. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019b. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. *arXiv preprint arXiv:1905.12866*.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6045–6049. IEEE.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Sam Coope, Tyler Farghly, Daniela Gerz, Ivan Vulić, and Matthew Henderson. 2020. Span-convert: Few-shot span extraction for dialog with pretrained conversational representations. *arXiv preprint arXiv:2005.08866*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Peter Dayan, Maneesh Sahani, and Grégoire Deback. 1999. Unsupervised learning. *The MIT encyclopedia of the cognitive sciences*, pages 857–859.

Anoop Deoras and Ruhi Sarikaya. 2013. Deep belief network based semantic taggers for spoken language understanding. In *Interspeech*, pages 2713–2717.

Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. 2019. Survey on evaluation methods for dialogue systems. *arXiv preprint arXiv:1905.04071*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

Arash Einolghozati, Panupong Pasupat, Sonal Gupta, Rushin Shah, Mrinal Mohit, Mike Lewis, and Luke Zettlemoyer. 2019. Improving semantic parsing for task oriented dialog. *arXiv preprint arXiv:1902.06000*.

Mihail Eric and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Maxine Eskenazi, Shikib Mehri, Evgeniia Razumovskaia, and Tiancheng Zhao. 2019. Beyond turing: Intelligent agents centered on the user. *arXiv preprint arXiv:1901.06613*.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, volume 2.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 445–450, Beijing, China. Association for Computational Linguistics.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational ai. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1371–1374.

Sarik Ghazarian, Johnny Tian-Zheng Wei, Aram Galstyan, and Nanyun Peng. 2019. Better automatic evaluation of open-domain dialogue systems with contextualized embeddings. *arXiv preprint arXiv:1904.10635*.

Sarik Ghazarian, Ralph Weischedel, Aram Galstyan, and Nanyun Peng. 2020. Predictive engagement: An efficient metric for automatic evaluation of open-domain dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7789–7796.

Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, GP Bhargav, Dinesh Garg, and Avirup Sil. 2019. Span selection pre-training for question answering. *arXiv preprint arXiv:1909.04120*.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, Dilek Hakkani-Tür, and Amazon Alexa AI. 2019. Topical-chat: Towards knowledge-grounded open-domain conversations. *Proc. Interspeech 2019*, pages 1891–1895.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 554–559. IEEE.

Prakhar Gupta, Shikib Mehri, Tiancheng Zhao, Amy Pavel, Maxine Eskenazi, and Jeffrey P Bigham. 2019. Investigating evaluation of open-domain dialogue systems with human generated multiple references. *arXiv preprint arXiv:1907.10568*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. *arXiv preprint arXiv:1810.07942*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić. 2020. Trippy: A triple copy strategy for value independent neural dialog state tracking. *arXiv preprint arXiv:2005.02877*.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Matthew Henderson, Inigo Casanueva, Nikola Mrkšić, Pei-Hao Su, Tsung-Hsien Wen, and Ivan Vulić. 2019. Convert: Efficient and accurate conversational representations from transformers. *arXiv preprint arXiv:1911.03688*.

Matthew Henderson and Ivan Vulić. 2020. Convex: Data-efficient and few-shot slot labeling. *arXiv preprint arXiv:2010.11791*.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org.

Lishan Huang, Zheng Ye, Jinghui Qin, Liang Lin, and Xiaodan Liang. 2020. Grade: Automatic graph-enhanced coherence metric for evaluating open-domain dialogue systems. *arXiv preprint arXiv:2010.03994*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. 2019. A surprisingly robust trick for winograd schema challenge. *arXiv preprint arXiv:1905.06290*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. *arXiv preprint arXiv:1908.08593*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv preprint arXiv:1601.01530*.

Tian Lan, Xian-Ling Mao, Wei Wei, Xiaoyan Gao, and Heyan Huang. 2020. Pone: A novel automatic evaluation metric for open-domain generative dialogue systems. *ACM Transactions on Information Systems (TOIS)*, 39(1):1–37.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. 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 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Michael Lebowitz. 1983. Generalization from natural language text. *Cognitive Science*, 7(1):1–40.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020. Pre-training via paraphrasing. *arXiv preprint arXiv:2006.15020*.

Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016a. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016b. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Zekang Li, Jinchao Zhang, Zhengcong Fei, Yang Feng, and Jie Zhou. 2021. Conversations are not flat: Modeling the dynamic information flow across dialogue utterances. *arXiv preprint arXiv:2106.02227*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Hsien-chin Lin, Nurul Lubis, Songbo Hu, Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, and Milica Gašić. 2021. Domain-independent user simulation with transformers for task-oriented dialogue systems. *arXiv preprint arXiv:2106.08838*.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019a. Benchmarking natural language understanding services for building conversational agents. *arXiv preprint arXiv:1903.05566*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A coarse-to-fine approach for cross-domain slot filling. *arXiv preprint arXiv:2004.11727*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ryan Lowe, Michael Noseworthy, Iulian V Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. *arXiv preprint arXiv:1708.07149*.

Dhiraj Madan, Dinesh Raghu, Gaurav Pandey, and Sachindra Joshi. 2018. Unsupervised learning of interpretable dialog models. *arXiv preprint arXiv:1811.01012*.

Andrea Madotto. 2020. Language models as few-shot learner for task-oriented dialogue systems. *arXiv preprint arXiv:2008.06239*.

Andrea Madotto, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2021. Few-shot bot: Prompt-based learning for dialogue systems. *arXiv preprint arXiv:2110.08118*.

Gary Marcus. 2018. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Shikib Mehri, Mihail Eric, and Dilek Hakkani-Tur. 2020a. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. *arXiv preprint arXiv:2009.13570*.

Shikib Mehri, Mihail Eric, and Dilek Hakkani-Tur. 2020b. Example-driven intent prediction with observers. *arXiv preprint arXiv:2010.08684*.

Shikib Mehri and Maxine Eskenazi. 2020a. Unsupervised evaluation of interactive dialog with dialogpt. *arXiv preprint arXiv:2006.12719*.

Shikib Mehri and Maxine Eskenazi. 2020b. Usr: An unsupervised and reference free evaluation metric for dialog generation. *arXiv preprint arXiv:2005.00456*.

Shikib Mehri and Maxine Eskenazi. 2021. Schema-guided paradigm for zero-shot dialog. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 499–508.

Shikib Mehri, Evgeniia Razumovskaia, Tiancheng Zhao, and Maxine Eskenazi. 2019. Pretraining methods for dialog context representation learning. *arXiv preprint arXiv:1906.00414*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2014. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, pages 3771–3775.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tom M Mitchell. 1980. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research ....

Tom M Mitchell et al. 1997. Machine learning.

Kaixiang Mo, Yu Zhang, Qiang Yang, and Pascale Fung. 2018. Cross-domain dialogue policy transfer via simultaneous speech-act and slot alignment. *arXiv preprint arXiv:1804.07691*.

Johannes EM Mosig, Shikib Mehri, and Thomas Kober. 2020. Star: A schema-guided dialog dataset for transfer learning. *arXiv preprint arXiv:2010.11853*.

Bo Pang, Erik Nijkamp, Wenjuan Han, Linqi Zhou, Yixian Liu, and Kewei Tu. 2020. Towards holistic and automatic evaluation of open-domain dialogue generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3619–3629.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 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.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020a. Soloist: Few-shot task-oriented dialog with a single pretrained auto-regressive model. *arXiv preprint arXiv:2005.05298*.

Baolin Peng and Kaisheng Yao. 2015. Recurrent neural networks with external memory for language understanding. *arXiv preprint arXiv:1506.00195*.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020b. Few-shot natural language generation for task-oriented dialog. *arXiv preprint arXiv:2002.12328*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Vitou Phy, Yang Zhao, and Akiko Aizawa. 2020. Deconstruct to reconstruct a configurable evaluation metric for open-domain dialogue systems. *arXiv preprint arXiv:2011.00483*.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. *arXiv preprint arXiv:1906.03520*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. 2018. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*.

Owen Rambow, Srinivas Bangalore, and Marilyn Walker. 2001. Natural language generation in dialog systems. Technical report, AT AND T LABS-RESEARCH FLORHAM PARK NJ.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020a. Schema-guided dialogue state tracking task at dstc8. *arXiv preprint arXiv:2002.01359*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020b. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Let's go public! taking a spoken dialog system to the real world. In *Ninth European conference on speech communication and technology*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. 2020. Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. In *Proceedings of The Web Conference 2020*, pages 2962–2968.

Yu-Ping Ruan, Zhen-Hua Ling, Jia-Chen Gu, and Quan Liu. 2020. Fine-tuning bert for schema-guided zero-shot dialogue state tracking. *arXiv preprint arXiv:2002.00181*.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162. Association for Computational Linguistics.

Ananya B Sai, Akash Kumar Mohankumar, Siddhartha Arora, and Mitesh M Khapra. 2020. Improving dialog evaluation with a multi-reference adversarial dataset and large scale pretraining. *Transactions of the Association for Computational Linguistics*, 8:810–827.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Ruhi Sarikaya, Geoffrey E Hinton, and Bhuvana Ramabhadran. 2011. Deep belief nets for natural language call-routing. In *2011 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pages 5680–5683. IEEE.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? how controllable attributes affect human judgments. *arXiv preprint arXiv:1902.08654*.

Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Darsh J Shah, Raghav Gupta, Amir A Fayazi, and Dilek Hakkani-Tur. 2019. Robust zero-shot cross-domain slot filling with example values. *arXiv preprint arXiv:1906.06870*.

Yangyang Shi, Kaisheng Yao, Hu Chen, Yi-Cheng Pan, Mei-Yuh Hwang, and Baolin Peng. 2015. Contextual spoken language understanding using recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5271–5275. IEEE.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. 2021. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *arXiv preprint arXiv:2104.06644*.

Koustuv Sinha, Prasanna Parthasarathi, Jasmine Wang, Ryan Lowe, William L Hamilton, and Joelle Pineau. 2020. Learning an unreferenced metric for online dialogue evaluation. *arXiv preprint arXiv:2005.00583*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and William B. Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *HLT-NAACL*.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2017. Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Zoltán Gendler Szabó. 2004. Compositionality.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sebastian Thrun and Tom M Mitchell. 1994. Learning one more thing. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

Jörg Tiedemann. 2009. News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248.

Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Alan Turing and J Haugeland. 1950. *Computing machinery and intelligence*. MIT Press Cambridge, MA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Anu Venkatesh, Chandra Khatri, Ashwin Ram, Fenfei Guo, Raefer Gabriel, Ashish Nagar, Rohit Prasad, Ming Cheng, Behnam Hedayatnia, Angeliki Metallinou, et al. 2018. On evaluating and comparing open domain dialog systems. *arXiv preprint arXiv:1801.03625*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. *arXiv preprint cmp-lg/9704004*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

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

Chien-Sheng Wu, Steven Hoi, Richard Socher, and Caiming Xiong. 2020. Tod-bert: pre-trained natural language understanding for task-oriented dialogue. *arXiv preprint arXiv:2004.06871*.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. *arXiv preprint arXiv:1905.08743*.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 ieee workshop on automatic speech recognition and understanding*, pages 78–83. IEEE.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2020. Ubar: Towards fully end-to-end task-oriented dialog systems with gpt-2. *arXiv preprint arXiv:2012.03539*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 189–194. IEEE.

Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. 2013. Recurrent neural networks for language understanding. In *Interspeech*, pages 2524–2528.

Yi-Ting Yeh, Maxine Eskenazi, and Shikib Mehri. 2021. A comprehensive assessment of dialog evaluation metrics. *arXiv preprint arXiv:2106.03706*.

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

Steve Young. 2010. Still talking to machines (cognitively speaking). In *Eleventh Annual Conference of the International Speech Communication Association*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Chen Zhang, Yiming Chen, Luis Fernando D'Haro, Yan Zhang, Thomas Friedrichs, Grandee Lee, and Haizhou Li. 2021a. Dynaeval: Unifying turn and dialogue level evaluation. *arXiv preprint arXiv:2106.01112*.

Chen Zhang, Luis Fernando D'Haro, Rafael E Banchs, Thomas Friedrichs, and Haizhou Li. 2021b. Deep am-fm: Toolkit for automatic dialogue evaluation. In *Conversational Dialogue Systems for the Next Decade*, pages 53–69. Springer.

Jian-Guo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. 2019a. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *arXiv preprint arXiv:1910.03544*.

Linhao Zhang, Dehong Ma, Xiaodong Zhang, Xiaohui Yan, and Houfeng Wang. 2020. Graph lstm with context-gated mechanism for spoken language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9539–9546.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019b. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019c. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*.

Lin Zhao and Zhe Feng. 2018. Improving slot filling in spoken language understanding with joint pointer and attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 426–431.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. *arXiv preprint arXiv:1805.04803*.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. *arXiv preprint arXiv:1804.08069*.

Tiancheng Zhao, Allen Lu, Kyusong Lee, and Maxine Eskenazi. 2017a. Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. *arXiv preprint arXiv:1706.08476*.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. 2019. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. *arXiv preprint arXiv:1902.08858*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017b. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.