**McGill University**

# Learning and evaluating neural network models for human-machine communication

by

Ryan Lowe

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Science
School of Computer Science, McGill University, Montreal

October 2, 2020

McGill University, Montreal

# *Abstract*

Faculty of Science
School of Computer Science, McGill University, Montreal

Doctor of Philosophy

by Ryan Lowe

In this thesis, I investigate several approaches to training and evaluating machine learning algorithms, where the ultimate goal is to have the algorithm interact and communicate with humans (i.e. a dialogue system). I consider two separate paradigms for producing such agents: supervised learning of dialogue agents from large text corpora, and multi-agent reinforcement learning for grounded communication tasks. In this first paradigm, I propose a dataset, the Ubuntu Dialogue Corpus, which represented one of the first large-scale, publicly-available datasets in this area. I then train various neural network models on this dataset to benchmark performance. Next, I conduct an extensive examination of evaluation metrics for dialogue systems, critiquing existing metrics based on word-overlap scores and proposing a new method, ADEM, that learns to evaluate dialogue responses from an evaluation dataset. In the second paradigm, I develop a new set of environments for multi-agent research called particle world, and propose a new algorithm called MADDPG that uses a centralized critic to improve agent performance across a variety of tasks, including a cooperative emergent communication task. I then examine some common pitfalls of measuring emergent communication, and propose two categories of metrics that must both be measured. Finally, I attempt to bridge the gap between these two paradigms: I investigate how self-play and supervised learning can be combined to produce effective language learning agents using an image-based referential game with English captions.

Université McGill

# *Abrégé*

Faculté des Sciences
School of Computer Science, Université McGill, Montréal

Doctor of Philosophy

par Ryan Lowe

Dans cette thèse, j'étudie plusieurs approches pour la formation et l'évaluation des algorithmes d'apprentissage, où le but ultime est de faire interagir l'algorithme avec les humains. Je considère deux paradigmes distincts pour produire de tels agents : l'apprentissage supervisé d'agents de dialogue à partir de grands corpus de texte, et l'apprentissage par renforcement multi-agents pour des tâches de communication ancrées dans une environment. Dans ce premier paradigme, je propose un ensemble de données, l'Ubuntu Dialogue qui représentait l'un des premiers ensembles de données à grande échelle accessibles au public dans ce domaine. Ensuite, je critique plusieus métriques d'évaluation pour les systèmes de dialogue, et propose une nouvelle méthode, ADEM, qui apprend à évaluer les réponses du dialogue à partir d'un ensemble de données d'évaluation. Dans le deuxième paradigme, je développe un nouvel ensemble d'environnements pour la recherche multi-agents appelé 'Particle World', et je propose un nouvel algorithme appelé MADDPG qui utilise une critique centralisée pour améliorer les performances des agents à travers une variété de tâches, y compris une tâche de communication émergente coopérative. J'examine ensuite certains pièges courants de la mesure de la communication émergente et propose deux catégories de paramètres qui doivent tous deux être mesurés. Enfin, j'essaye d'unifier ces deux paradigmes : j'étudie comment l'apprentissage par renforcement multi-agents et l'apprentissage supervisé peuvent être combinés pour produire des agents qui apprennent les langues efficacement.

# Contribution to Original Knowledge

This thesis contributes to the understanding of how we might use, and how we should evaluate, neural network models for human-machine communication. Specifically, we make the following set of contributions:

1. A dataset for training large-scale dialogue systems, the Ubuntu Dialogue Corpus, along with benchmarked results from several generative and retrieval neural network models trained on this dataset.
2. Several analyses of methods for evaluating responses from non-task-oriented dialogue systems, including:
   - an analysis of the correlation of word-overlap and vector-based metrics with human judgements;
   - a study evaluating human performance on the task of Next Utterance Classification across two datasets;
   - a model that learns to evaluate dialogue responses based on a dataset of collected human judgements on Twitter.
3. A configurable environment for training and evaluating reinforcement learning agents in a multi-agent setting, and an algorithm for training them more efficiently.
4. An analysis of methods for evaluating emergent communication, and a proposal for how to categorize these metrics.
5. An investigation into how supervised learning can be combined with self-play to train agents to play an image-based referential game with natural language.

# Contribution of Authors

- Chapters 1 and 2 providing the introduction and background material for this thesis were written by me, with inspiration from various sources including the theses of Pierre-Luc Bacon, Phil Bachman, Jiwei Li, and Jakob Foerster.

- Chapter 3 is based on (Lowe et al., 2017b), which is a journal paper in *Dialogue & Discourse* based in turn on earlier results in (Lowe et al., 2015), a conference paper at SIGDIAL. In the original work, my equal co-author Nissan Pow and I led the project. Nissan and I wrote the code for extracting and formatting the Ubuntu Dialogue Corpus, and Nissan devised and ran the experiments for testing the Dual Encoder model. Nissan and I analyzed the results, and I wrote the first iterations of the paper. Iulian Serban and Joelle Pineau provided technical direction and leadership, and contributed to the writing. In the updated journal version, I performed additional experiments along with Iulian Serban and Chia-Wei Liu for the generative models. Rudolf Kadlec and Martin Schmid led the development of the Ubuntu Dialogue Corpusv2, along with Nissan and me. I conducted the additional quantitative and qualitative analysis and wrote the paper, with contributions from Iulian, Joelle, and Laurent Charlin.

- Chapter 4 incorporates work from several papers, described below.

  – Chapter 4.1 is based on the work in (Liu et al., 2016), a conference paper at EMNLP. My equal co-authors Chia-Wei Liu, Iulian Serban, Michael Noseworthy and I led the work. Chia-Wei started the investigation of word-overlap metrics for dialogue evaluation. Iulian, Michael, and I continued this work with him and we each contributed significantly to the analysis. Iulian and I wrote the first version of the paper, with subsequent iterations by all authors including Laurent Charlin and Joelle Pineau. Laurent and Joelle provided technical guidance.

  - Chapter 4.2 is based on (Lowe et al., 2016), a conference paper at SIGDIAL. I led this work, including designing and performing the human experiments and writing the paper. Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau all provided guidance throughout the project and helped with writing.
  - Chapter 4.3 is based on the work in (Lowe et al., 2017a), a conference paper at ACL. I led the initial work, including designing the ADEM model and performing the core experiments using representations from a pre-trained VHRED model provided by Iulian Serban. I wrote the first version of the paper, with help from all co-authors. My equal co-author Michael Noseworthy helped with the experiment design, ran many more experiments on the ADEM model to analyze its performance, and produced the final version of the paper. Nicolas Angelard-Gontier also ran some experiments and helped with the analysis. Iulian, Michael, Nicolas, Laurent Charlin, and Joelle Pineau helped with the technical direction and writing of the paper. Yoshua Bengio provided ideas and inspiration for some of the project, including the idea of learning a discriminator which was ultimately not included in the final version.

- Chapter 5 originates from a conference paper at NeurIPS (Lowe et al., 2017c). The project was primarily led by myself, Igor Mordatch, and Yi Wu. Igor wrote most of the base code for the Particle World environments. I worked on various algorithms to solve the cooperative communication task over the course of several months, with Igor and Pieter Abbeel providing technical direction. Igor came up with the idea of using a centralized critic in MADDPG and implemented the first version. I ran most of the experiments and devised most of the tasks in Particle World, along with Igor. In parallel, my equal co-author Yi Wu implemented and ran experiments for both the policy ensembles and learning policies of other agents, and wrote this section of the paper. Yi also came up with the keep-away environment. Igor and I wrote the rest of the paper with help from Yi, Pieter, and Aviv Tamar. Aviv also provided some technical guidance and derived the proof of Proposition 1. Jean Harb helped run some of the experiments.

- Chapter 6 is based on a conference paper at AAMAS (Lowe et al., 2019). I led the project, with technical guidance from Yann Dauphin, Y-Lan Boureau, Jakob Foerster, and Joelle Pineau. I ran the experiments and wrote the paper, with input from my co-authors along the way.

- Chapter 7 is based on a paper to be presented at ICLR (Lowe et al., 2020). Jakob Foerster and I had the original idea for how to bridge the gap from emergent communication to natural language in 2017. My equal co-author Abhinav Gupta started working on this idea in January 2020, and I joined him shortly thereafter. Abhinav and I co-led the project, including devising and running the experiments and writing the paper, with technical guidance from Jakob, Douwe Kiela, and Joelle Pineau. I wrote the first version of the paper, with help from Abhinav and the other co-authors. Abhinav made significant later revisions to the paper. The paper's direction was influenced by a meeting with Angeliki Lazaridou at ICML 2019.

# *Acknowledgements*

First off, I'd like to thank my supervisor, Joelle Pineau, for continually putting me in positions to succeed, for being a role model for how to conduct scientific research, and for being extremely kind and patient with me.

I'd also like to thank my collaborators and co-authors for their help in conducting the research that makes up this thesis: Pieter Abbeel, Nicolas Angelard-Gontier, Yoshua Bengio, Y-Lan Boureau, Laurent Charlin, Yann Dauphin, Jakob Foerster, Abhinav Gupta, Jean Harb, Douwe Kiela, Chia-Wei Liu, Igor Mordatch, Michael Noseworthy, Nissan Pow, Iulian Serban, Aviv Tamar, and Yi Wu, along with many others at McGill, OpenAI and Facebook AI.

My PhD life slowly split into two separate and yet partially overlapping worlds. I'm grateful for all my friends from my base in Montreal who made it feel like home for almost 5 years, including: Phil Bachman, Pierre-Luc Bacon, Emmanuel Bengio, Genevieve Fried, Neil Ghirdar, Jean Harb, Jad Kabbara, Leo Long, Josh Romoff, Harsh Satija, and the entire RL Lab, the Allez Up climbing crew, the McGill Mafia club, Liam Fedus, David Krueger, Tegan Maharaj, Michael Noukhovitch, Evan Racah, and my other friends at Mila, Mido Assran and the rest of Facebook AI Montreal, along with Jeff Cai, Sophie Davis, Mathieu Harb, Evgeny Naumov, and Victor Thibert.

No less important are the people who've welcomed me to my new home in San Francisco and expanded my horizons in exciting and unpredictable ways, starting with Diogo Almeida, Michael Azorin, Brian and Alexi Basham, Natasha Jensen, Michael Nagle, Janine Parziale, Sasha Sheng, Morgan Sutherland, Leo Urbina, Clayton Ward, Alex Warren, and the rest of the beautiful people that make up the BePo universe, Topos, Starcats, the members of all the board game crews and improv troupes, Matthias Plappert and my new and old friends at OpenAI, and of course the entire gold group and ESF family who each have a special place in my heart.

There are many friends and machine learning colleagues who don't fit neatly into these categories who have added joy to my life over the past few years. A special shout-out goes to the entire CBN crew, as well as David Krueger, Jan Leike, Jelena Luketina, CFAR, and others who slowly nudged me towards working on AI safety.

Finally, I’m grateful for my entire family, especially my parents, Tom and Lynn, and my brother Rourke. They’ve been my staunchest cheerleaders for a long time. Love you guys.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Communication between humans and machines has long been a fascination of both artificial intelligence (AI) researchers and the general public. Indeed, building systems that can naturally and meaningfully communicate with humans has been a central goal of AI since the formulation of the Turing test (Turing, 1950). The creation of the first chatbot, ELIZA, by Joseph Weizenbaum in 1966 spurred further excitement in computers that can have conversations with humans (Weizenbaum, 1966). Despite only consisting of a few rules that mostly parroted back the users' input (in the form of a Rogerian psychotherapist), ELIZA was nonetheless able to deceive some interlocutors at the time into thinking it was human (Colby et al., 1972).

In the last decade, conversational systems (also known as *dialogue systems*) have moved from idle curiosity to become increasingly prevalent in commercial applications. Major software companies are now developing chatbots that interact with millions of people. For example, Microsoft has developed a bot, called Xiaoice, which holds regular conversations with tens of millions of users in China and around the world (Markoff and Mozur, 2015). More recently, Google has created a product called Duplex that makes voice calls to businesses ranging from restaurants to hair stylists in order to make reservations, book movie tickets, and perform other tasks for human users (Chen and Metz, 2019). Chatbots are even showing promise in mental therapy applications (Brodwin, 2018).

Moving forward, language promises to be an important component to developing increasingly capable AI systems. Language allows humans to communicate their intentions, disseminate knowledge, and ask questions to learn about the world. Similarly, we would like AI systems to be able to receive instructions, request clarification, and share information with humans. This will become increasingly true as AI systems become more powerful and present in our everyday lives. Thus, it seems imperative that we teach these machines to communicate via human language so that they can be integrated productively and safely into our society.

## 1.2 Setting the stage

To further motivate the research questions tackled in this thesis, we briefly describe the state of research on dialogue systems and deep neural networks applied to natural language processing (NLP) in 2015, the year in which the research that makes up this thesis commenced.

Prior to 2015, the landscape of dialogue systems contained few deep neural networks. Most dialogue systems were *modular*, in that they were composed of a set of modules each trained roughly independently using a separate objective function. This contrasts with the *end-to-end* approach to training dialogue systems, where all of the parameters of the model are trained with a single objective function. This is often accomplished by using a single neural network to replace all of the modules in a modular system (see a more detailed discussion of modular dialogue systems in Chapter 2).

While there has been significant work on implementing and testing modular systems in a variety of domains (Wen et al., 2016), research on modular dialogue systems can also focus on optimizing the performance of individual modules. For example, dialogue state tracking is the task of predicting a user's goal as the dialogue progresses (Nagata and Morimoto, 1994; Young, 2000; Williams et al., 2013). This is usually done by collecting a dataset of goal annotations for a certain domain (say, movie ticket booking), and training models via supervised learning to predict these annotations. In 2015, dialogue state tracking was a major component of dialogue research, as exemplified by the Dialogue State Tracking Challenges (DSTC), a sequence of research competitions to determine the state-of-the-art state trackers (Williams et al., 2013; Kim et al., 2017; Henderson et al., 2013). The DSTC was held 5 times between 2013 and 2016, before being re-branded as the Dialogue System Technology Challenge to encapsulate a broader range of dialogue-related tasks. Around this time, dialogue systems research was most successful on small, well-defined datasets, such as restaurant and movie booking, and public transportation information (Williams et al., 2014; Wen et al., 2016).

Concurrently, neural network models were starting to gain traction on a variety of hard supervised learning problems. For example, neural networks were becoming useful on tasks such as speech recognition (Hinton et al., 2012), object recognition (Krizhevsky et al., 2012), and machine translation (Bahdanau et al., 2014; Cho et al., 2014), as well as in game-playing tasks (Mnih et al., 2015) with reinforcement learning. In many of these domains, neural networks were approaching or exceeding the previous state-of-the-art.

In most of these cases, neural networks were being applied differently than via a set of distinct modules described above for dialogue. Rather than deciding beforehand which components their system needed, researchers would instead simply define the desired input and output of their system and model it in its entirety with a single neural network. This simple end-to-end approach has proven to be surprisingly powerful performance-wise, and has the added benefit of not requiring labeled data for each sub-component of the system. The success of these large neural network-based approaches gave rise to a new term to describe them, called *deep learning* (LeCun et al., 2015). Although most of the tools of deep learning were invented several decades earlier (Rumelhart et al., 1988; Elman, 1990; LeCun et al., 1998; Hochreiter and Schmidhuber, 1997), the era of deep learning was distinguished by a dramatic increase in the available data and compute (Amodei and Hernandez, 2018), and some new techniques to train neural networks efficiently (Glorot and Bengio, 2010; Glorot et al., 2011).

In the beginning of 2015, when the research that makes up this thesis was started, modern deep learning techniques had not yet been applied to learning dialogue models. However, the idea was 'in the air': many papers would come out later that year on the subject from a variety of research labs (Sordoni et al., 2015b; Shang et al., 2015; Vinyals and Le, 2015; Serban et al., 2016; Li et al., 2015). The work in this thesis, starting from (Lowe et al., 2015), tracks some of the evolving thought in our understanding of models for human-machine communication: from supervised learning on dialogue datasets, to grounded language learning via emergent communication, and finally to a modern take on bridging the gap between these perspectives.

## 1.3 Focus of this thesis

In this thesis, we investigate neural network models for producing agents that can communicate with humans, and study how to evaluate them. We tackle this problem from two different perspectives: supervised learning of dialogue agents from large English text corpora, and training agents to communicate from scratch in simulated environments via emergent communication. In this thesis, for pragmatic reasons we will refer to the communication protocol used by either humans or artificial agents as a ‘language’, while remaining agnostic to the philosophical and linguistic question of whether emerged artificial communication protocols can be considered a true ‘language’.

In this section, we discuss the motivation for studying both dialogue systems and emergent communication, and our focus on both learning and evaluation. We conclude with a list of research questions investigated in the following chapters.

### 1.3.1 Dialogue systems and emergent communication

A natural way for humans to communicate with machines is through language. After all, this is how most humans communicate with each other. The study of how to build artificial systems that interact with humans through natural language has long been the purview of the field of *dialogue systems*. Much of this thesis is focused on building dialogue systems by training neural networks on large-scale dialogue corpora.

However, the reason that language is useful for humans is that we can use it to accomplish tasks in our environment. That is, words in a language aren’t just abstract entities but *refer* to things in the real world. This is the notion of *grounding*, which has been much discussed in the literature on linguistics, natural language processing (NLP), and other fields (Siskind, 1994; Mikolov et al., 2016; Gauthier and Mordatch, 2016; Steels, 2012; Glenberg and Kaschak, 2002). While training neural networks to imitate human dialogue on large text corpora can result in models that generate plausible English, they are not grounded in the sense that they do not refer to any external world other than the language itself.

An alternate approach to training agents to learn language is to place them in an environment with other agents, where the objective is to solve a particular task requiring communication between agents. In this scenario, language (usually through the form of a dedicated, discrete communication channel) is one of many tools that agents can use to solve problems in their environment. Since in this setting agents must often develop a language to solve a task from scratch, we refer to this set of approaches as *emergent communication*. Work on simulated language emergence has a long history (Murray, 1999; Steels, 2012; Wagner et al., 2003). More recent work on emergent communication uses reinforcement learning (RL) to maximize the environment reward, with agents' policies represented by neural networks (Sukhbaatar et al., 2016; Foerster et al., 2016; Jaques et al., 2018a; Lazaridou et al., 2018). The motivation for using emergent communication as a testbed for developing language learning agents was recently articulated in (Gauthier and Mordatch, 2016).

In this thesis, we make contributions to both the 'supervised learning from large-scale dialogue datasets' and 'emergent communication in grounded environments' approaches to teaching machines language. In Chapter 7, we show how these two approaches can be complementary, and potentially combined. We also discuss our current perspective on the strengths and weaknesses of these two paradigms in Chapter 8.

### 1.3.2 Learning and evaluation

The goal of this thesis is to make contributions to the field's understanding of human-machine communication, through the lens of large-scale dialogue modeling and emergent communication. Part of these contributions come from developing and designing models and learning algorithms to solve various tasks. This style of contribution makes up the bulk of Chapters 3, 5, and 7. Focusing on developing better learning algorithms is important, as it's the main way the field makes progress to solving increasingly interesting and relevant problems.

But in this thesis we also place significant focus on *proposing new datasets and environments* and *understanding and improving evaluation methods*. Specifically, the focus of Chapters 4 and 6 is on the evaluation of dialogue systems and emergent communication, respectively, and we also propose a new dialogue dataset (the Ubuntu Dialogue Corpus) in Chapter 3 and a new multi-agent environment (Particle World) in Chapter 5.

In the era of deep learning (LeCun et al., 2015), data (along with compute) are among the most crucial factors driving progress. For example, the historic ImageNet dataset (Deng et al., 2009) became a focal point for computer vision researchers, and led to the development of large-scale convolutional neural networks (Krizhevsky et al., 2012) which sparked the recent wave in neural network applications. Thus, work on large-scale datasets often ends up being more useful than work on improving models, as without a suitably large task-relevant dataset it is often difficult for large-scale neural networks to make much progress. More generally, the set of datasets available in a given sub-field shapes how researchers think about the problem they are tackling, and the types of approaches they consider.

The state of dialogue systems research in early 2015 was marked by a dearth of large-scale datasets that were publicly available; the first efforts to apply neural networks to train a dialogue model end-to-end used mostly proprietary datasets (Vinyals and Le, 2015). The Ubuntu Dialogue Corpus, which we proposed in (Lowe et al., 2015) and detail in Chapter 3, was among the first large-scale publicly available datasets for training dialogue models, and in retrospect helped kick-start the surge in dialogue research with end-to-end neural network models of the past few years. Our Particle World environment (Lowe et al., 2017c) has also seen significant use in multi-agent RL research. Thus, we consider new datasets and environments to be an important part of this thesis' contributions.

However, large datasets are useless without an appropriate way to measure performance that correlates with the behaviour we actually care about. We therefore dedicate a significant portion of this thesis to the study of *evaluation metrics* for benchmarking performance. Evaluation is particularly difficult in chitchat-style dialogue systems, where the question of what makes a 'good' dialogue is very open ended. In cases where evaluation metrics diverge from the actual desired characteristics of a model (for example, usefulness on a downstream task), improving a model according to these metrics is not helpful and simply adds noise to the field.

It is often the case that evaluation metrics used to measure progress in a field turn out to correlate poorly with the desired model behaviour. For example, there have been challenges in evaluating generative models of images, and it has been shown that Parzen window estimates (once used to evaluate generative model quality) exhibit undesirable properties, such as giving higher scores to model samples than actual images from the dataset (Theis et al., 2015). Detecting these irregularities is important for the overall health of a field, especially since evaluation metrics determine which models are considered state-of-the-art, which in turn informs many researchers' decisions on what ideas to pursue. One of our most important contributions in this thesis is from (Liu et al., 2016) (detailed in Chapter 4.1), where we provide evidence that the BLEU score (and other word overlap-based metrics) correlate poorly to human judgements of dialogue response quality. This is important because BLEU was previously the most commonly used evaluation metric for chit-chat dialogue systems using neural networks — our work helped seed more recent work on methods to evaluate dialogue systems (Tao et al., 2018), although the problem remains unsolved.

### 1.3.3 Research questions

In this section, we list the main research questions that are tackled in this thesis. In Part I, we address the following questions:

- How can we build end-to-end dialogue systems using neural networks? (Chapter 3)
- What kinds of errors do these models make? (Chapter 3)
- How well do existing automatic evaluation metrics in dialogue correlate with human judgements? (Chapter 4.1)

- Can we create better automatic evaluation metrics for dialogue? (Chapters 4.2-4.3)

In Part II, we ask the following questions:

- How can we improve multi-agent RL algorithms to get them to solve some simple emergent communication tasks? (Chapter 5)
- How should we be evaluating emergent languages? (Chapter 6)
- How can we bridge the gap between work on language learning from supervised datasets and emergent communication? (Chapter 7)

We provide a preview of some of our results in the next section, give more detailed answers throughout the thesis, and end with a summary and extended discussion in Chapter 8.

## 1.4 Preview of results

This thesis is divided into two parts. In Part I (Chapters 3-4), we investigate methods for training and evaluating dialogue systems trained on large datasets. In Part II (Chapters 5-7) we detail work on emergent communication and multi-agent reinforcement learning. At the end of Part II, we describe how these two frameworks could fit together.

We first present necessary background information referenced throughout the thesis in Chapter 2. In Chapter 3, we introduce the Ubuntu Dialogue Corpus, which at the time of its release was among the first large-scale dialogue datasets that was publicly available. We use this dataset to train a variety of end-to-end dialogue models using neural networks, along with some baseline models. We consider both *retrieval* models, which are trained to select the correct next response of a conversation from a list of candidates, and *generative* models that are trained to generate a response conditioned on the context. When evaluating these models on the task of next utterance classification (NUC), a retrieval-based metric, we find that the (at the time) state-of-the-art models, the LSTM dual encoder (DE) and hierarchical recurrent encoder-decoder (HRED), outperform the baselines on all metrics. We also conduct a qualitative analysis to determine the main sources of errors for the DE model, and find that the most common errors are a lack of understanding of the semantics of the responses, indicating that these models are not close to solving a domain as complex as Ubuntu.

In Chapter 4, we analyze various approaches to evaluating dialogue systems. First, in Section 4.1, we critique commonly used word overlap-based metrics, such as BLEU and ROUGE, as well as vector-based metrics. Specifically, we find that these metrics do not correlate well with human judgements on the Twitter and Ubuntu datasets. In Section 4.2, we conduct a human study of a retrieval-based task that we call next utterance classification (NUC), and find that humans outperform state-of-the-art (at publication time) machine learning models on Twitter and Ubuntu, suggesting that the task is useful for benchmarking progress of dialogue systems. Finally, in Section 4.3 we propose to *learn* a dialogue evaluation model to predict human judgements of Twitter dialogues. We find that our model, which we call ADEM, performs well on our test set (in terms of correlating with human judgements) and even generalizes to models that it hasn't been trained on. However, more recent evidence suggests that ADEM does not generalize to other subsets of Twitter data, perhaps due to the bias in the data collection procedure. Finding automatic evaluation metrics that correlate well with human judgements remains an important area of future work.

We begin Part II by describing a new multi-agent reinforcement learning (RL) algorithm for improved emergent communication in Chapter 5. Specifically, we propose an actor-critic method where the critic, which is trained to evaluate the expected value of an agent's action, is given information from all agents during training time. To test our algorithm, we also propose a new benchmark for multi-agent learning called Particle World, which consists of small environments with simulated physics where agents (represented as circles) can carry out simple tasks. We find that our algorithm leads to significant performance gains on several environments in our Particle World, including a simple cooperative communication task where an agent needs to signal the color of a landmark to another agent, which we found challenging to solve with other methods.

We turn our focus in Chapter 6 to the evaluation of emergent communication. Specifically, we aim to provide a set of evaluation tools for detecting and measuring emergent communication. To aid our analysis, we categorize emergent communication metrics into those that measure *positive signaling* (whether the agent is sending messages that are related to its observation or action), and those that measure *positive listening* (whether the agent's messages are affecting the behaviour of other agents). Both of these properties are desirable for communicative agents, since we want them to speak about their environment and react to what others are saying. In the chapter, we first find that positive signaling can emerge without positive listening. Then, we describe a family of metrics (some existing, some novel) according to whether they measure positive signaling or positive listening. In particular, we note that many existing metrics and analyses did not detect the presence of positive listening.

Finally, in Chapter 7 we address how we might *bridge the gap* between the work on emergent communication in Part II, and the work on natural language dialogue in Part I. Specifically, we investigate how to combine two categories of training signals: imitating human data via supervised learning, and self-play to maximize reward in a multi-agent environment (the training signal used in most work on emergent communication). We find evidence that, contrary to our initial assumption, it is not beneficial to emerge a language from scratch and subsequently fine-tune on human data. Rather, it is best to start with learning from human data, and schedule intermittent self-play updates. We provide three reasons for why this could be the case, and study various learning schedules in an image-based referential game with natural language. We also propose a population-based method in this setting that leads to improved performance.

We conclude in Chapter 8 with a summary of our contributions, and a candid discussion of the strengths and limitations of this line of work. Specifically, we highlight that this thesis only begins to address the questions surrounding how to incorporate methods from large-scale dialogue modeling and emergent communication. We reflect on this and other interesting future directions, most notably the use of very large-scale language modeling and how it interacts with having agents use language to accomplish tasks in situated environments that may contain objects and even other agents.

# Chapter 2

# Technical background

In this section, we review the fundamentals required for understanding the contributions of this thesis, including artificial neural networks, sequence-to-sequence generation, dialogue systems, reinforcement learning (RL), and multi-agent RL.

## 2.1 Supervised learning

We first briefly review the paradigm of supervised learning. In supervised learning, the goal is to learn a target function $f^* : \mathcal{X} \rightarrow \mathcal{Y}$, using experience sampled from a dataset $\mathcal{D}_{\mathcal{X}\mathcal{Y}}$. $\mathcal{D}_{\mathcal{X}\mathcal{Y}}$ usually consists of (input, output) pairs of the form $(x, y), x \in \mathcal{X}, y \in \mathcal{Y}$. To do this, the learning algorithm is tasked with minimizing a scalar loss function $\mathcal{L}(\mathcal{D}_{xy}, f)$, where $f$ is the function learned by the learning algorithm to approximate $f^*$.

Two prominent forms of supervised learning that we will be using throughout this thesis are *classification* and *regression*. In classification, the target function $f^*$ has the form $f^* : \mathbb{R}^n \rightarrow \{1, ..., k\}$. In other words, $f^*$ maps from a real-valued vector $\mathbf{x} \in \mathcal{X}$ to a set of $k$ discrete categories $y$. An example classification task is object recognition, where the input $\mathbf{x}$ is an image and the output $y$ is a numeric code identifying the object in the image. Language modeling, described in Section 2.3 and used frequently throughout this thesis, is also a classification task.

In regression problems, on the other hand, the target function $f^* : \mathbb{R}^n \to \mathbb{R}$ maps from a vector $\mathbf{x} \in \mathcal{X}$ to a scalar numerical value. An example regression task is predicting tomorrow's temperature, given the temperature and other weather conditions from the previous week.

## 2.2 Artificial neural networks

### 2.2.1 Feed-forward neural networks

Deep feed-forward neural networks, often called multi-layer perceptrons (MLPs), are a fundamental building block of deep learning research, and they are useful for a wide variety of supervised learning problems (Ivakhnenko and Lapa, 1966; Schmidhuber, 2015; Goodfellow et al., 2015). The goal of an MLP is to approximate a function $f^*(x) = y$ with a learned *non-linear* function $f(x;\theta)$, where $x \in \mathbb{R}^k$ are the inputs and $y \in \mathbb{R}^l$ are the labels for some dataset, and $\theta$ are the parameters of the MLP. An MLP has multiple layers in the sense that it is formed from the composition of multiple functions, i.e. $f(x) = f^{(n)}(f^{(n-1)}(\cdots f^{(1)}(x))\cdots))$ for a network with $n$ layers. The final layer $f^{(n)}$ is often called the *output layer*, while layers $f^{(1)}, \ldots, f^{(n-1)}$ are called *hidden layers* as there is no target for the output of these layers.

When using MLPs in practice, a very standard formulation is used for each layer $f^{(i)}$ in the network: $f^{(i)}(x) = g(W[i] \cdot x + b[i])$ where $W[i]$ is a matrix of real-valued parameters, $b[i]$ is a vector-valued bias ($W[i]$ and $b[i]$ are learned separately for each layer), and $g(\cdot)$ is a non-linear *activation function*. Usually, a single fixed activation function $g(\cdot)$ is chosen and used throughout the network. Popular choices for $g(\cdot)$ include the hyperbolic tangent function $\tanh = \frac{e^{2x}-1}{e^{2x}+1}$, the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, and the rectified linear unit (ReLU) $ReLU(x) = \max(0, x)$ (Glorot et al., 2011; Goodfellow et al., 2015; Schmidhuber, 2015).

The output layer of a neural network depends on the task. For regression tasks, where the goal is to predict a real-valued outcome, the final layer is often linear, omitting $g(\cdot)$: $f^{(n)}(x) = W \cdot x + b$. For classification tasks where each output neuron represents a class, a softmax distribution is often used for the

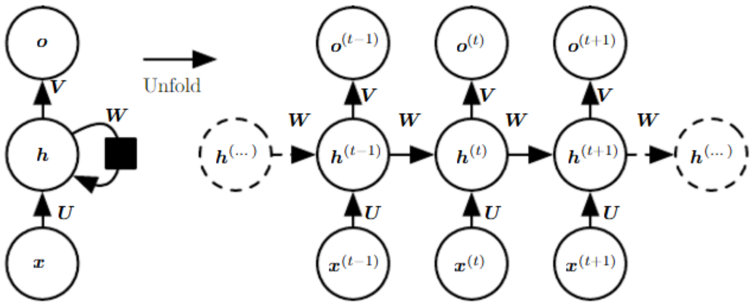



FIGURE 2.1: A computation graph of an RNN; the unrolled RNN is shown on the right. Figure modified from (Goodfellow et al., 2015).

non-linearity: $softmax(x) = \frac{e^{x_j}}{\sum_j e^{x_j}}$ where the sum in the denominator is taken over all of the neurons in the output layer (i.e. over all classes).

### 2.2.2 Recurrent neural networks

Recurrent neural networks (RNNs) augment feed-forward neural networks with feedback loops. Most often, these are self loops from hidden neurons to themselves and to other neurons in the same layer (Elman, 1990; Jordan, 1990). The idea is that we want to be able to model data that is correlated in time, and does not follow the independent and identically distributed (i.i.d.) assumption. This can take the form of sequences of input data $x_1, \ldots, x_m$.

The addition of feedback loops to the hidden units of an MLP can be thought of as forming a *hidden state* of the network $h(x)$, corresponding to the vector of activations (value after the non-linearity) of the hidden neurons, that evolves over time as we present inputs to the network. The hidden state is updated at each time step according to some function $f_h$: $h_t = f_h(h_{t-1}, x_t)$. A crucial point is that the parameters of $f(\cdot)$ are the same for all time steps. More specifically, an RNN is governed by the following update equations at time $t$:

$$h_t = g(Wh_{t-1} + Ux_t + b) \tag{2.1}$$
$$o_t = softmax(Vh_t + c) \tag{2.2}$$

where $W$ is the hidden-to-hidden matrix of parameters, $U$ is the input matrix, $V$ is the output matrix, $b$ is the hidden state bias and $c$ is the output bias. The most common non-linearity $g(\cdot)$ used in RNNs is the tanh function.

The configuration of RNNs can be modified to fit a wide variety of tasks. For example, an RNN with a single output layer at the last time step can be used to make a categorical prediction when taking sequential data as input, such as in sentiment analysis (Tang et al., 2015). An RNN can also be used as a language model, as described in the next section.

### 2.2.3 Training via backpropagation

We have described the basic architecture of MLPs and RNNs, and how they produce outputs given some input. Neural networks are almost always trained to minimize some (usually differentiable) loss function with gradient descent on the parameters. This is applied to neural networks using the *backpropagation* algorithm.

Backpropagation (Rumelhart et al., 1988), often called backprop, is an algorithm that describes how to update the parameters $\theta$ of a neural network to minimize a loss function $\mathcal{L}$ with gradient descent. It is essentially an application of the chain rule: for each computation layer $f^{(i)}(x) = g(W_i x + b_i)$ that composes a neural network, the gradients are calculated by taking first taking the gradient of the output of layer $f^{(i)}$ in the network with respect to the loss, and then taking the gradient of the output of this layer with respect to the parameters $\theta_i = \{W_i, b_i\}$:

$$\nabla_{\theta_i}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial f^{(i)}} \nabla_{\theta_i} f^{(i)} = \frac{\partial \mathcal{L}}{\partial f^{(n)}} \prod_{j=i:n-1} \frac{\partial f^{(j+1)}}{f^{(j)}} \nabla_{\theta_i} f^{(i)}. \tag{2.3}$$

The parameters $\theta_i$ are then updated a small distance in the direction of the negative of the gradient, as determined by a learning rate $\alpha$:

$$\theta_i \leftarrow \theta_i - \alpha \nabla_{\theta_i} \mathcal{L}. \tag{2.4}$$

Recurrent neural networks are most often trained using the *backpropagation through time* (BPTT) algorithm (Werbos, 1990). This is the same as the usual backpropagation algorithm, but the gradient of the hidden state at each time step additionally considers the gradient of the hidden state at the next time step. BPTT can also be seen as normal backpropagation on the unrolled RNN computation graph in Figure 2.1.

### 2.2.4 Long short-term memory units

One problem that arises when applying BPTT to RNNs is that the gradients to earlier parts of the network can become arbitrarily small or large. This is because the hidden-to-hidden weight matrix $W$ is the same at every time step, and this matrix gets multiplied iteratively (via the chain rule) to calculate the gradient at earlier time steps (Hochreiter, 1991; Bengio et al., 1994; Hochreiter et al., 2001). By analogy with scalar numbers, if you take a number and multiply it by another number $k \in \mathbb{R}_{>0}$ a large number of times, the resulting number will either become exponentially large (if $k > 1$) or exponentially small (if $0 < k < 1$). In the case of BPTT, $W$ plays the role of $k$, and the gradient will become very large or small depending on the spectral radius of $W$ (Pascanu et al., 2013).

One way to alleviate this problem in RNNs is by introducing long short-term memory units (Hochreiter and Schmidhuber, 1997). LSTMs augment RNNs with a cell state, which stores information from previous time steps, and a set of multiplicative gates that determine how much information is written and forgotten from the cell state at each time step. The cell state allows the LSTM to propagate gradient information further back in time (so long as information from earlier time steps was kept on the cell state), as backpropagation through the cell state requires a number of matrix multiplications that's constant in the number of time steps.

More formally, given a sequence of input data $x_1, ..., x_m$ (with each $x \in \mathbb{R}$), an LSTM contains at each step an input gate $i_t$, a forget gate $f_t$, and an output gate $o_t$. The cell state at time $t$ is denoted $c_t$. Then the hidden state (used to calculate the output of the network) $h_t$ is calculated as follows:

$$i_t = \sigma(W^i \cdot [h_{t-1}, x_t]) \tag{2.5}$$

$$f_t = \sigma(W^f \cdot [h_{t-1}, x_t]) \tag{2.6}$$

$$o_t = \sigma(W^o \cdot [h_{t-1}, x_t]) \tag{2.7}$$

$$l_t = \tanh(W^l \cdot [h_{t-1}, x_t]) \tag{2.8}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \tag{2.9}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{2.10}$$

where $\sigma$ is the sigmoid function, $W^i, W^f, W^o, W^l \in \mathbb{R}^{n \times 2n}$, $\cdot$ represents the element-wise dot product between two vectors, $[a, b]$ represents the concatenation of vectors $a$ and $b$, and $l_t$ can be interpreted as the information from the input and hidden state that is to be 'written' to the cell state $c_t$.

Although LSTMs were originally proposed in (Hochreiter and Schmidhuber, 1997), they have stood the test of time and have seen significant use in a variety of deep learning applications, such as phoneme classification, (Graves and Schmidhuber, 2005), handwriting synthesis (Graves, 2013), machine translation (Sutskever et al., 2014; Cho et al., 2014), image captioning (Xu et al., 2015), polyphonic music modeling (Chung et al., 2014), and others. In fact, although there have been several LSTM variants that have been proposed with different gating arrangements (Cho et al., 2014), these modifications do not consistently outperform the original LSTM architecture across all tasks (Greff et al., 2015).

## 2.3 Language modeling with neural networks

After having described the basic architecture of neural networks, we now briefly describe how they can be applied to the task of *language modeling*, as this forms the basis of work on generative neural dialogue models.

### 2.3.1 Language modeling

Language modeling is the task of predicting the next word in a sequence, given the previous words. Variants of language modeling have been used in a wide variety of NLP tasks, from speech recognition (Jelinek, 1990) to machine translation (Jozefowicz et al., 2016).

More concretely, given a sequence $x = \{x_1, ..., x_m\}$, in language modeling we decompose the joint probability over all sequences $p(x)$ into a product of conditional probabilities:

$$p(x) = \prod_{t=1}^{m} p(x_t | x_1, ..., x_{t-1}). \tag{2.11}$$

Different language modeling methods vary in how they model the conditional probabilities $p(x_t|x_{<t})$. One of the simplest and most common approaches is called n-gram modeling, which uses a simple statistical count to predict the probability of the next token given the $n-1$ preceding tokens. N-gram language models, and more complex variants that incorporate smoothing, have historically seen frequent use (Kneser and Ney, 1995; Rosenfeld, 2000; Federico, 1996). However, n-gram models are limited as they require significant memory (the number of n-grams grows exponentially in the number of words in the vocabulary), consider only a very limited context window, and suffer from data sparsity issues.

### 2.3.2 Neural language models

Another way to approximate the conditional probability $p(x_t|x_{<t})$ is using a neural network. To do this, one usually represents each word $x_t$ as a vector $\mathbf{x}_t \in \mathbb{R}^k$. A commonly used vector representation for words is *one-hot vectors*, where each $\mathbf{x}_t$ has dimension equal to the size of the vocabulary $|V|$, and consists of all 0 entries except for a single 1 corresponding to the numerical index of word $x_t$. Using a feed-forward network, one can then take the previous $n$ tokens as input, and output a softmax distribution over all words in the vocabulary. This was first attempted in (Bengio et al., 2003), and it helps overcome the problems of memory (since only the parameters of the neural network need to be stored) and data sparsity (as the model can learn to associate similar words). However, the length of context that can be considered is limited by the width of the input of the network.

The data sparsity issue can be further alleviated by using *dense pre-trained word vectors* as inputs, as opposed to one-hot vectors the size of the vocabulary. These word vectors, also called *word embeddings*, are commonly trained in an unsupervised way by leveraging distributional semantics, the idea that "you can know a word by the company it keeps" (Firth, 1957). There have been a wide variety of approaches to learning pre-trained word embeddings, the most popular being word2vec (Mikolov et al., 2013a), Glove (Pennington et al., 2014), and, more recently, innovations in contextual word embeddings (Peters et al., 2018).

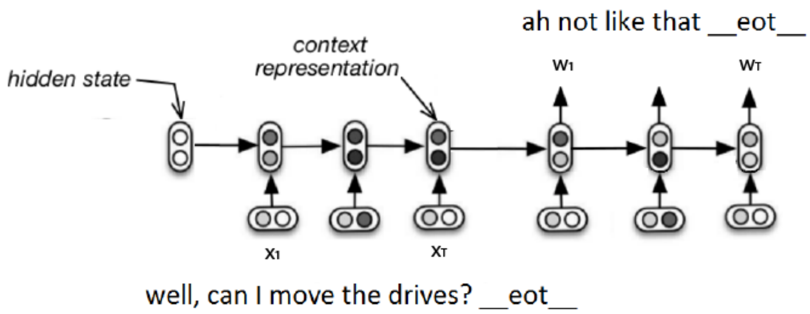



FIGURE 2.2: Diagram of an RNN language model in the dialogue domain. If the RNN used to produce the context representation (left) and the RNN used to decode the next sentence (right) have different parameters, then this becomes an encoder-decoder architecture (described in Section 2.3.3).

To allow for longer contexts, language models can be implemented using a recurrent neural network, which we call an RNN language model (RNN-LM) (Mikolov et al., 2010). The model observes the sentence word-by-word and updates its hidden state $h_t$ at time step (word index) $t$. Given a hidden state $h_t$ the model then outputs a probability distribution over all words in the vocabulary. This allows the prediction of the next word to implicitly depend on all previous words seen by the model (as moderated by the hidden state $h_t$). We depict an RNN language model in the dialogue domain in Figure 2.2.

We will take $\hat{y}_n = \{w_1, ..., w_T\}$ as the target output sequence for the $n^{\text{th}}$ training example, given the input sequence $\hat{x}_n = \{x_1, ..., x_T\}$. The conditional distribution over each output symbol is computed in a similar manner, and depends on the current hidden state $h_t$:

$$p(w_t | w_{t-1}, ..., w_1) = \frac{\exp(W^o_{w_t} \cdot h_t)}{\sum_w \exp(W^o_w \cdot h_t)}, \tag{2.12}$$

where $W^o$ is the output matrix, and $W^o_w$ indicates the row of the $W^o$ matrix corresponding to the output index for word $w$. The model is trained with *teacher forcing*, meaning the input $x_t$ to the network is the previous ground-truth output word $w_{t-1}$.

The model is trained in an end-to-end fashion using the BPTT algorithm to maximize the conditional log-likelihood of input-output pairs from the training set, $\{\hat{x}_n, \hat{y}_n\}$:

$$\max_\theta \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(\hat{y}_n | \hat{x}_n), \tag{2.13}$$

where $\theta$ are the parameters of the model, including $W^x, W^h, W^o$, and the corresponding biases. Thus, the model learns a probability distribution over all output sequences, $p(\hat{y}_1, ..., \hat{y}_T)$.

### 2.3.3 Encoder-decoders

The basic idea of the encoder-decoder is simple: we first use an RNN (called the *encoder*) to map the input sequence to a fixed-length vector, the final hidden state of the RNN. We then use this fixed-length vector to condition a separate RNN with different parameters (called the *decoder*) to map this vector back to another sequence. This conditioning can be done by using the output of the encoder as the initial hidden state of the decoder, or by feeding it as input to the decoder at each time step. We consider the case where the output sequence consists of tokens from some vocabulary $V$, as is the case for dialogue. Thus, the output of the decoder RNN at each time step is a categorical distribution defined by a softmax, where the output dimension is the size of the vocabulary $|V|$. Each component of the output vector thus represents the probability that a certain token $y \in V$ should be the next token output by the decoder RNN. Together, this sequence-to-sequence model is called an *encoder-decoder* (Cho et al., 2014; Sutskever et al., 2014).

A key innovation of the encoder-decoder framework is to augment the vocabulary $V$ with a special *end-of-sequence symbol*, $EOS$. When this token is output by the decoder, the sequence is terminated. The parameters of the encoder and decoder are trained jointly to minimize the loss function, which for dialogue is most often the negative log-likelihood. Encoder-decoder methods have become prevalent for a variety of NLP tasks, including machine translation (Cho et al., 2014; Bahdanau et al., 2014) and automatic summarization (Rush et al., 2015).

## 2.4 Dialogue systems

In this section, we introduce some of the terminology that we use to refer to dialogue systems, which is used throughout the remainder of this thesis.

### 2.4.1 Task-oriented vs. non-task-oriented dialogue systems

We first make an important distinction between what we call *task-oriented* and *non-task-oriented* dialogue systems. It is important to distinguish between these kinds of dialogue systems as they have different goals, and are often evaluated differently.

The focus of our dialogue work in this thesis is on non-task-oriented dialogue systems, often called *chit-chat* dialogue systems (Weizenbaum, 1966; Parkinson et al., 1977; Ritter et al., 2011; Li, 2020). In non-task-oriented systems, the goal is to entertain the user in some way, either by engaging them, comforting them, or simply talking about what the user wants to talk about. In chit-chat systems, there is no specific task that the user wants to accomplish (such as book a hotel); instead, they are simply engaging with the system for the fun of it. As such, evaluation of these systems is not straightforward; ideally, users can be asked at the end of the dialogue to give a score relevant to the goal at hand (e.g. engagingness). In general, non-task-oriented systems are usually better the longer they are able to converse with a user (which means the user is more engaged). In practice, chit-chat systems are often evaluated at the response-level, based on how reasonable a response it produced given the context of the conversation (Li et al., 2015, 2016a; Sordoni et al., 2015b).

Task-oriented systems, on the other hand, are designed to accomplish a specific task of the user, such as booking a hotel, finding a restaurant, or looking up transportation information (Walker and Whittaker, 1990; Seneff, 1992; Young, 2000; Williams and Young, 2007; Young et al., 2010, 2013; Wen et al., 2016). As such, these systems can be concretely evaluated by whether or not they accomplished the user's task. Contrary to chit-chat systems, task-oriented systems are generally doing a better job the *less* they interact with the user, given that they are able to solve the user's task (since, when a human interlocutor has a specific task in mind, they don't want their conversation to drag out). Further, simply measuring the 'reasonableness' of a task-oriented system at the response level is not an appropriate metric for evaluation, as there are usually reasonable responses that do not solve the user's task (such as 'I don't know').

### 2.4.2 Retrieval vs. generative dialogue systems

Whereas the task-oriented vs. non-task-oriented distinction separates dialogue systems based on their goal when interacting with a user, *retrieval* and *generative* systems are ways to describe how dialogue systems are trained and how they perform inference. Both retrieval and generative systems can be used for either task-oriented or non-task-oriented systems. We study both retrieval and generative models in Chapter 3.

A retrieval-based dialogue system is one that produces a conversational response by searching through the set of responses available in the dialogue dataset (Isbell et al., 2000; Jafarpour et al., 2010; Al-Rfou et al., 2016). Thus, a retrieval system cannot generate new responses that have not been seen in the dataset. On the other hand, responses from retrieval systems are generally coherent (as they were generated by humans), and are easier to curate to eliminate potentially offensive or harmful responses. Retrieval models can be trained in a variety of ways, e.g. by learning to classify, given a context, whether a response is the actual next response to the conversation or whether it is drawn randomly from elsewhere in the dataset (Bordes et al., 2014; Yu et al., 2014). In the rest of this thesis, we refer to the task of distinguishing between the actual next response and a set of random responses as *next utterance classification* (NUC).

A generative dialogue system, on the other hand, generates a response from scratch to each conversational context (Ritter et al., 2011; Sordoni et al., 2015b; Serban et al., 2016; Ghazvininejad et al., 2018). This provides significantly more flexibility compared to retrieval systems, in terms of ability to customize a response to a given input. However, generative systems are at risk of generating less coherent or more generic responses (Li et al., 2015). Generative systems can be trained in many ways, but when using neural networks they are generally trained via teacher forcing to predict the next token in the response conditioned on the previous tokens (Sordoni et al., 2015b; Serban et al., 2016).

Note that this retrieval vs. generative distinction does not describe the full space of possible dialogue systems. For example, historically many dialogue systems were *rule-based*, as they used a sequence of rules to determine how to construct a response (which usually combines some sentence stems pre-defined by the dialogue system creator). For example, this is how ELIZA was built (Weizenbaum, 1966), along with many extensions (Parkinson et al., 1977). There

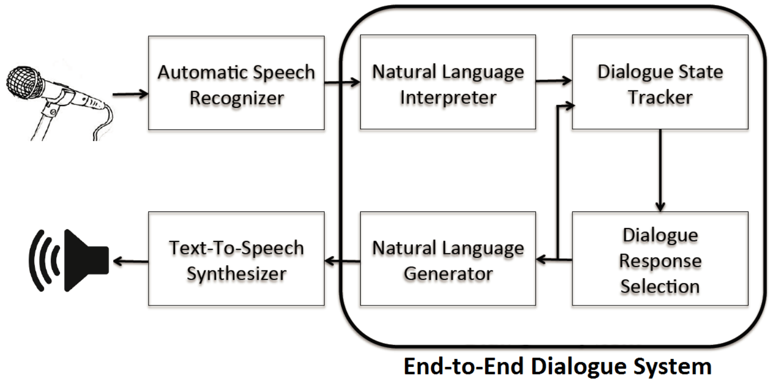



FIGURE 2.3: An end-to-end dialogue system replaces the components of a dialogue system with a single model.

has also been recent work on combining generative and retrieval based methods, for example by first retrieving a relevant response from the dataset and then making some local edits to customize the response to the given context (Chen et al., 2017; Song et al., 2016).

### 2.4.3 Modular vs. end-to-end dialogue systems

Much of the work presented in Chapter 3 is a benchmarking of end-to-end approaches to training dialogue systems with neural networks, which was a fairly novel area at the time of first publication. Thus, it is important to define specifically what is meant by an 'end-to-end' dialogue system. We begin with the standard architecture for a dialogue system, shown in Figure 2.3, which incorporates a Speech Recognizer, Language Interpreter, State Tracker, Response Generator, Natural Language Generator, and Speech Synthesizer. In the case of text-based (written) dialogues, the Speech Recognizer and Speech Synthesizer can be left out. Although some previous literature on dialogue systems identifies only the State Tracker and Response Selection components as belonging inside the dialogue manager (Young, 2000), we adopt a broader view where the Language Interpreter and Natural Language Generator are also part of the dialogue manager, consistent with more recent literature (Serban et al., 2015).

When we speak of an 'end-to-end' dialogue system, we mean a single system that can be used to solve each of these four aspects simultaneously. Typically this is a system that takes as input the history of the conversation and is trained to optimize a single objective, which is a function of the textual output produced by the system and the correct (ground truth) response. This is in contrast to the 'modular' system approach to dialogue systems, where each component of Figure 2.3 is trained separately, and either takes a more structured input, such as a set of dialogue acts, or is trained to maximize an intermediary objective, such as slot-filling. More formally, we define a modular dialogue system as a system where two or more elements (sub-components or system parameters) are optimized with respect to two or more different objective functions (e.g. where the State Tracker is trained to minimize the cross-entropy error of predicting the slot-value pairs, and where the Response Generator is trained to maximize the conditional log-likelihood of the correct response given the slot-value pairs).

Note that, according to this definition, whether a system is end-to-end is *independent of how it is evaluated*. Both retrieval and generative models can be end-to-end so long as they are trained using a single objective function. Similarly, end-to-end models can be evaluated using intermediary tasks such as NUC, which do not evaluate the ability of the models to generate new utterances unseen in the training set. Of course, in order to evaluate the full capability of the models it is best to evaluate their outputs in a setting as realistic as possible; however, this is difficult to do automatically when there is no notion of task completion (Liu et al., 2016).

Examples of end-to-end dialogue systems in the recent literature involve neural network-based approaches that are fully differentiable, and are usually trained to maximize the log-likelihood of the generated utterance conditioned on some conversational context (Serban et al., 2016; Vinyals and Le, 2015). These systems learn off-line through examples of human-human dialogues, and thus learn to emulate the behaviour of agents in the training corpus. However, differentiability and off-line learning are not strict prerequisites for end-to-end dialogue systems, and other methods could be devised.

Modular dialogue systems have been historically preferred over end-to-end systems (Raux et al., 2005; Young et al., 2013). This is because such modular systems are easier to train, require less data, and so far have been shown to achieve better results in practice, albeit typically for highly structured tasks. It is also easier to manually program each component specifically to obey certain task constraints or to solve one or more isolated tasks. However, there are significant advantages to end-to-end dialogue systems that make investigating them worthwhile. In particular, modular dialogue systems are restricted to task-specific domains, and often require significant human feature engineering, including pre-defining the state and action spaces of the model. Although this can work well for narrow domains, it does not necessarily generalize to general-purpose dialogue.

On the other hand, end-to-end models do not require a pre-defined state or action space representation; instead, these representations are learned directly from conversational data. Once an end-to-end model architecture is specified, all that is needed to have the system learn to converse about another domain is to provide new training data for that domain. As the amount of available dialogue data grows and more general-purpose conversational systems are desired, we believe that training end-to-end models without hand-crafted features will yield better performance.

## 2.5 Reinforcement learning

In this section we describe reinforcement learning (RL), a framework which we use for our work on multi-agent communication, including MDPs and popular RL algorithms.

### 2.5.1 Markov Decision Processes (MDP)

An MDP (Bellman, 1957) is a tuple $< \mathcal{S}, \mathcal{A}, P, r >$ consisting of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a state transition probability distribution $P : \mathcal{A} \times \mathcal{S} \rightarrow (\mathcal{S} \rightarrow [0,1])$, and a reward function $r : \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. Agents acting in an MDP will use either a stochastic policy $\boldsymbol{\pi} : \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ or a deterministic policy $\boldsymbol{\pi} : \mathcal{S} \rightarrow \mathcal{A}$ in order to maximize their future discounted reward (the *return*), $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ where $\gamma \in (0,1)$ is a discount factor and $T$ is the time horizon. Reinforcement learning methods (Sutton and Barto, 1998) are an approach to solving MDPs when the environment dynamics $P$ are unknown, and we only have sample-based access to the environment.

### 2.5.2 Partially Observed Markov Decision Processes (POMDPs)

A POMDP (Åström, 1965; Kaelbling et al., 1998) is a generalization of MDPs to include the case where an agent only observes a subset of the state at each time step. More specifically, a POMDP is a tuple $< \mathcal{S}, \mathcal{A}, P, \Omega, O, r >$, where $\Omega$ is the set of *observations*, and $O$ is the conditional observation probability $O : \mathcal{A} \times \mathcal{S} \to (\Omega \to [0, 1])$ that provides the probability of an agent observing observation $o_t \in \Omega$ at time step $t$ given the current state $s_t \in \mathcal{S}$ and previous action $a_{t-1} \in \mathcal{A}$. An agent's policy then maps from the set of observations to actions.

### 2.5.3 Q-Learning and Deep Q-Networks (DQN).

Q-Learning and DQN are popular methods in reinforcement learning, as they have lead to a variety of impressive results in game-playing (Mnih et al., 2015) and other domains. Q-learning makes use of an *action-value function* $Q^{\boldsymbol{\pi}}(s, a)$ for policy $\boldsymbol{\pi}$, defined as the expected future reward if the agent takes action $a$ in state $s$ at time $t$, and follows policy $\boldsymbol{\pi}$ thereafter: $Q^{\boldsymbol{\pi}}(s, a) = \mathbb{E}[R|s^t = s, a^t = a]$. In Q-learning, we learn an approximation to this Q function $Q_\phi$ using a function approximator with parameters $\phi$ (in DQN, this is a deep neural network). The Q function can be recursively rewritten using the *Bellman equation* (Bellman, 1957): $Q^{\boldsymbol{\pi}}(s, a) = \mathbb{E}_{s'}[r(s, a) + \gamma \mathbb{E}_{a' \sim \boldsymbol{\pi}}[Q^{\boldsymbol{\pi}}(s', a')]]$. In DQN, we use this formulation to learn our approximate Q function $Q_\phi^{\boldsymbol{\pi}}$ (the approximate Q function $Q_\phi$ for policy $\boldsymbol{\pi}$) by minimizing the loss:

$$\mathcal{L}(\phi) = \mathbb{E}_{s,a,r,s'}[(Q_\phi^{\boldsymbol{\pi}}(s, a) - y)^2], \qquad \text{where} \qquad y = r + \gamma \max_{a'} Q_{\bar{\phi}}^{\boldsymbol{\pi}}(s', a'), \tag{2.14}$$

where $Q_{\bar{\phi}}^{\boldsymbol{\pi}}$ is a target Q function that uses old parameters $\bar{\phi}$ from earlier in training, which helps stabilize learning; these parameters are periodically updated with the most recent $\phi$.

Another crucial component of stabilizing DQN is the use of an experience replay buffer $\mathcal{D}$ containing tuples $(s, a, r, s')$. To encourage exploration, an *$\epsilon$-greedy* approach is usually used, where the agent choose a random action with probability $\epsilon$, and the action with the highest Q value with probability $1 - \epsilon$. Because of this, Q-learning is an *off-policy* learning algorithm, meaning that the policy used during learning is not the same as the policy executed at runtime (which does not use $\epsilon$-greedy exploration).

### 2.5.4 Policy Gradient (PG) Algorithms.

Policy gradient methods are another popular choice for a variety of RL tasks, from continuous control (Lillicrap et al., 2015) to Atari game-playing (Mnih et al., 2016). In policy gradient methods, rather than learning an action-value function $Q_\phi$, we directly adjust the parameters $\theta$ of the policy $\boldsymbol{\pi}$ in order to maximize the cumulative reward $J(\theta) = \mathbb{E}_{s \sim p^\pi, a \sim \boldsymbol{\pi}_\theta}[R]$ by taking steps in the direction of $\nabla_\theta J(\theta)$. Using the Q function defined previously, the gradient of the policy can be written as (Sutton and Barto, 1998):

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim p^\pi, a \sim \boldsymbol{\pi}_\theta}[\nabla_\theta \log \boldsymbol{\pi}_\theta(a|s) Q^{\boldsymbol{\pi}}(s,a)], \tag{2.15}$$

where $p^{\boldsymbol{\pi}}$ is the state distribution. The policy gradient theorem has given rise to several practical algorithms, which often differ in how they estimate $Q^{\boldsymbol{\pi}}$. For example, one can use a sample return $R^t = \sum_{i=t}^T \gamma^{i-t} r_i$, which leads to the REINFORCE algorithm (Williams, 1992). Alternatively, one could learn an approximation of $Q^{\boldsymbol{\pi}}(s,a)$ by e.g. temporal-difference learning (Sutton and Barto, 1998); this $Q^{\boldsymbol{\pi}}(s,a)$ is called the *critic* and leads to a variety of *actor-critic* algorithms (Sutton and Barto, 1998).

### 2.5.5 Deterministic Policy Gradient (DPG) Algorithms.

It is also possible to extend the policy gradient framework to deterministic policies $\boldsymbol{\mu}_\theta : \mathcal{S} \mapsto \mathcal{A}$. We can write the gradient of the objective $J(\theta) = \mathbb{E}_{s \sim p^{\boldsymbol{\mu}}}[R(s,a)]$ as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \mathcal{D}}[\nabla_\theta \boldsymbol{\mu}_\theta(a|s) \nabla_a Q^{\boldsymbol{\mu}}(s,a)|_{a=\boldsymbol{\mu}_\theta(s)}] \tag{2.16}$$

Since this theorem relies on $\nabla_a Q^{\boldsymbol{\mu}}(s,a)$, it requires that the action space $\mathcal{A}$ (and thus the policy $\boldsymbol{\mu}$) be continuous.

*Deep deterministic policy gradient* (DDPG) is a variant of DPG where the policy $\boldsymbol{\mu}$ and critic $Q^{\boldsymbol{\mu}}$ are approximated with deep neural networks. DDPG is an off-policy algorithm, and samples trajectories from a replay buffer of experiences that are stored throughout training. DDPG also makes use of a target network, as in DQN (Mnih et al., 2015).

## 2.6 Multi-agent RL

In most work on emergent communication, agents are placed in environments where at least one other agent is present — otherwise, there is no need to communicate at all! When these agents are trained with reinforcement learning, this leads to a special set of considerations and methods that are the domain of *multi-agent reinforcement learning* (MARL). In this section, we give some background on multi-agent RL, including a formalization of the setting and a distinction between different kinds of multi-agent training.

### 2.6.1 Markov games

The multi-agent environments that we develop in Chapters 5 and 6 can be considered multi-agent extensions of partially observable Markov decision processes (POMDPs), called partially observable Markov games (Littman, 1994). A Markov game for $N$ agents is defined by a set of states $\mathcal{S}$, $N$ sets of actions $\mathcal{A}_1, ..., \mathcal{A}_N$ and $N$ sets of observations $\mathcal{O}_1, ..., \mathcal{O}_N$, one each per agent. To choose actions, each agent $i$ uses a stochastic policy $\boldsymbol{\pi}_{\theta_i} : \mathcal{O}_i \times \mathcal{A}_i \mapsto [0, 1]$, which produces the next state according to the state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \mapsto \mathcal{S}$.

Each agent $i$ obtains rewards as a function of the state and agent's action $r_i : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \mapsto \mathbb{R}$, and receives a private observation correlated with the state $o_i : \mathcal{S} \mapsto \mathcal{O}_i$. The initial states are determined by a distribution $\rho : \mathcal{S} \mapsto [0, 1]$. Each agent $i$ aims to maximize its own total expected return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ where $\gamma$ is a discount factor and $T$ is the time horizon.

### 2.6.2 Extension to communication games

In the games we consider that involve communication, the action space $\mathcal{A}_i$ for each agent $i$ can be subdivided into disjoint *environment actions* $\mathcal{A}_i^e$, and *communication actions* $\mathcal{A}_i^m$, such that $\mathcal{A}_i^e \cup \mathcal{A}_i^m = \mathcal{A}_i$ and $\mathcal{A}_i^e \cap \mathcal{A}_i^m = \emptyset$. Environment actions are those that have a direct effect on the environment dynamics and the rewards obtained by the agent. We model each agent's communication action as a sequence of discrete symbols sent over a dedicated communication channel, which are observed by the other agents at the next time step. Communication actions do not affect the environment dynamics (other than being observed by the other agent), and incur a fixed cost $r_c \in \mathbb{R}_{<0}$. Throughout this thesis, we consider the *cheap talk* setting (Farrell and Rabin, 1996) where $r_c = 0$, and we do not examine the domain of costly signaling (Zahavi, 1975; Gintis et al., 2001).

### 2.6.3 Centralized vs. decentralized learning and execution

Traditional RL approaches can be naïvely applied to the multi-agent case, by having each agent use a separate policy to compute their actions, and run a separate RL algorithm during training. The former paradigm is called *decentralized execution* (or sometimes *decentralized control*), while the latter is called *decentralized learning*. The canonical example of decentralized learning and decentralized execution is independent Q-learning, where each agent $i$ learns an independently optimal function $Q_i$ (Tan, 1993) and independently performs Q-learning. However, because agents are independently updating their policies as learning progresses, the environment appears non-stationary from the view of any one agent, violating Markov assumptions required for convergence of Q-learning. This is a general problem for decentralized learning in multi-agent environments, as there is no way to take into account the fact that the other agents in the environment are also learning. Another difficulty observed in (Foerster et al., 2017b) is that an experience replay buffer, which is commonly used in many deep RL algorithms, cannot be used in such a setting since in general, $P(s'|s, a, \boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_N) \neq P(s'|s, a, \boldsymbol{\pi}'_1, ..., \boldsymbol{\pi}'_N)$ when any $\boldsymbol{\pi}_i \neq \boldsymbol{\pi}'_i$.

One way to circumvent these problems is to perform *centralized learning* with *centralized execution*. Centralized learning refers to training agents using additional information other than simply their observations and reward, such as the observations or actions of other agents in the environment. Centralized execution (or centralized control) refers to having a single policy that maps from states to the join actions of all agents in the environment. While centralized execution allows agents to overcome the non-stationarity issues mentioned previously, it is infeasible in many practical applications of interest where agents only have access to their own local observations at execution time. Further, with centralized execution there is no need to communicate via sending discrete messages (i.e. language) as all agents have access to the observations of all other agents.

Thus, a common approach in modern deep multi-agent RL is centralized learning with decentralized execution. In this setting, agents have separate policies (and thus, only use their own observations to produce actions at execution time), but are trained using extra information about the environment or other agents' policies. This is feasible in many real-world scenarios, such as when we train an agent using a simulator and deploy it in the real world, and permits agents to learn to communicate with each other through language. This setting has become quite popular in recent work on multi-agent RL (Foerster et al., 2017a). In this thesis, we consider the centralized learning and decentralized execution paradigm when proposing a new multi-agent RL algorithm in Chapter 5; however, we stick to the simpler decentralized learning and execution setting when studying evaluation metrics in Chapter 6.

Finally note that, for practical purposes, centralized execution necessarily implies centralized learning, and thus we do not consider the case of decentralized learning and centralized execution.

# Part I

# Learning and evaluating dialogue systems

# Chapter 3

# Training end-to-end dialogue models with the Ubuntu Dialogue Corpus

## 3.1 Motivation

Historically, dialogue models required significant hand-engineering of features, and thus could only generate a limited number of responses and be deployed in constrained situations (Walker and Whittaker, 1990; Seneff, 1992; Nagata and Morimoto, 1994). In 2015, advances in neural network-based language models had begun to make feasible the idea of learning an entire dialogue model directly from conversational data, with humans only specifying the model hyper-parameters. The promise of these *end-to-end* systems is that they are more general, can be easily re-trained on new domains, and their performance scales directly with the amount of training data given to them. However, at the time these kind of models were relatively untested for dialogue, and significant work needed to be done before they could be implemented in practice with high confidence.

In this chapter we consider the problem of building dialogue agents in an end-to-end manner. We define end-to-end systems, contrary to *modular* systems, as those that are trained directly from conversational data to optimize a single objective function (see Section 2.4.3). To do this, we introduce a dataset called the Ubuntu Dialogue Corpus. This dataset consists of almost one million two-person (dyadic) conversations extracted from the Ubuntu chat logs, which provide technical support for various Ubuntu-related problems. Dialogues in the corpus are *multi-turn* and *unstructured*, as there is no *a priori* logical representation for the information exchanged during the conversation. This is in contrast to recent systems which focus on structured dialogue tasks, using slot-filling representations (Williams et al., 2013; Henderson et al., 2014a; Singh et al., 2002).

The creation of such a large, unstructured dialogue dataset was motivated by observations of progress in various sub-fields of AI. In particular, it has been argued that this progress can be attributed to three major factors: 1) the public distribution of very large rich datasets (Deng et al., 2009), 2) the availability of substantial computing power, and 3) the development of new training methods for neural architectures, in particular leveraging unlabeled data. In 2015, the Ubuntu Dialogue Corpus was among the first large-scale dialogue datasets that was publicly available, and thus it can be argued that this dataset spurred some of the subsequent work on large-scale end-to-end dialogue systems Serban et al. (2017); Bordes and Weston (2016); Chen et al. (2017); Wu et al. (2016).

In this chapter, we conduct an analysis of several dialogue models that can be used in conjunction with the Ubuntu Dialogue Corpus. We first consider *retrieval* or *classification* models, which are trained to select the correct next response of a conversation from a list of candidate responses. We use a baseline model that calculates term frequency-inverse document frequency (TF-IDF) between the context and each response, and compare it to a Dual Encoder (DE) model using both recurrent neural networks (RNNs) and long short-term memory (LSTMs). Next, we present *encoder-decoder* models that are trained to generate an utterance given the context. We consider both the traditional LSTM language model, which corresponds to the encoder and decoder having tied weights, and the recently proposed Hierarchical Recurrent Encoder-Decoder (HRED) (Serban et al., 2016), which has a second recurrent network that encodes utterance-specific information, and is thus able to model longer-term dependencies in the context.

We evaluate these models on the task of next utterance classification (NUC), where the model ranks a list of candidate responses by how likely they are to have followed the context. We also evaluate using vector-based metrics to determine the quality of generated responses, in terms of semantic similarity to the ground-truth next utterance. We observe that the (at the time) state-of-the-art models, the LSTM dual encoder and HRED, outperform the baselines on all metrics.

Finally, we conduct a qualitative analysis to determine the main sources of error for the DE model. We find that the most common errors are a lack of understanding of the semantics of the responses, which includes missing key words that are copied between the context and target response, and a lack of higher-level inference. There are also a number of cases where the model would benefit from explicitly incorporating the turn-taking structure of dialogue, and using some source of external knowledge for Ubuntu terminology. An examination of the responses produced by the generative models reveals similar shortcomings; while the models are able to generate reasonable responses, they are often generic or lack a semantic understanding of the context. It is clear that end-to-end systems are not close to solving a domain as complex as Ubuntu. We hope that this analysis can help guide future research on the Ubuntu Dialogue Corpus, and the development of end-to-end dialogue systems.

## 3.2 Related work: Learning architectures for end-to-end dialogue systems

Most dialogue research has historically focused on structured slot-filling tasks (Schatzmann et al., 2005). Various approaches were proposed, yet few attempts leverage more recent developments in neural learning architectures. A notable exception is the work of Henderson et al. (2014b), which proposes an RNN structure, initialized with a denoising autoencoder, to tackle the DSTC 3 domain.

Work on end-to-end dialogue systems was recently pioneered by Ritter et al. (2011), who proposed a response generation model for Twitter data based on ideas from Statistical Machine Translation. In particular, they consider a model that 'translates' from the context of a conversation to the associated response. This is shown to give superior performance to previous information retrieval (e.g. nearest neighbour) approaches (Jafarpour et al., 2010). This idea was further developed by Sordoni et al. (2015a) to exploit information from a longer context, using a structure similar to the Recurrent Neural Network Encoder-Decoder model (Cho et al., 2014). This achieves rather poor performance on A-B-A Twitter triples when measured by the BLEU score (a standard for machine translation), yet performs comparatively better than the model of Ritter et al. (2011). Their results were also verified with a human-subject study.

A similar Encoder-Decoder framework for dialogue is presented by Shang et al. (2015) and Vinyals and Le (2015). This model also uses one RNN to transform the input to some vector representation, and another RNN to 'decode' this representation to a response by generating one word at a time. The model from Shang et al. (2015) was also evaluated in a human-subject study, although on a smaller scale compared to Sordoni et al. (2015a).

A hierarchical version of the encoder-decoder framework has also recently been proposed (Serban et al., 2016). This model consists of two RNNs stacked on top of each other: one 'sentence-level' RNN encodes each utterance into a fixed length vector, while a 'conversation-level' RNN takes as input each utterance vector and outputs a vector that summarizes the conversation so far. This is mapped back to text using a recurrent decoder. This improves over the traditional Encoder-Decoder frameworks in both word perplexity and word error rate, particularly when bootstrapped with word embeddings derived from distributional semantics. However, the model has not been evaluated in any human-subject studies.

Another approach, taken in Traum et al. (2015), uses information retrieval techniques to map user questions to systems responses in the domain of time-offset interaction. Since the natural language interpreter, dialogue response selection, and natural language generator model are all combined, this can also be seen as a form of end-to-end dialogue system. Inaba and Takahashi (2016) also propose an end-to-end retrieval model, however they use neural networks to select a response from a fixed dataset. This is similar to the model used by Lowe et al. (2015). Our work is also inspired by Nio et al. (2014a) whose model, although rule-based, is not composed of modules, as it retrieves a response to the context based on cosine similarity. This is in turn related to the work on example-based dialogue modeling (Lee et al., 2009).

There has been some work on combining end-to-end dialogue models with auxiliary information regarding the persona or participant role of each person in the dialogue. Luan et al. (2016) investigate several models that incorporate participant roles, using topic-modelling based approaches with LDA. Li et al. (2016a) use an embedding for each separate speaker in the conversation, which is used to condition the decoder in an LSTM model. They achieve improvements in both perplexity and BLEU on a Twitter dataset.

There has also been interesting work using deep reinforcement learning for end-to-end dialogue generation. Li et al. (2016b) propose using a deep Q-network (DQN) for dialogue generation, using a set of reward functions designed to increase the diversity of generated responses. Zhao and Eskenazi (2016) similarly use a deep recurrent Q-network (DRQN) to replace the conventional NLU, state tracking, and dialogue policy modules for task-oriented dialogue.

One of the most effective task-oriented end-to-end systems is presented by Wen et al. (2016), who train an end-to-end system on a small dataset of restaurant recommendations. They show that they are able to achieve a higher task completion rate than a modular baseline, and have significantly higher scores in naturalness, comprehension, preference, and performance.

Overall, these models highlight the potential of end-to-end learning architectures for interactive systems. However, much work remains before these can be implemented with confidence in a variety of settings.

## 3.3 The Ubuntu Dialogue Corpus

### 3.3.1 Motivation

There are several factors that motivated the creation of the Ubuntu Dialogue Corpus. In particular, there was a lack of *large*, *multi-turn*, *publicly available* dialogue datasets. In addition to providing a dataset that satisfied these constraints, we wanted the dataset to be two-way (dyadic), as opposed to multi-participant chat, and we desired a task-specific domain. All of these characteristics are satisfied by the Ubuntu Dialogue Corpus.

Note that there are two versions of the Ubuntu Dialogue Corpus: v1, first proposed in Lowe et al. (2015), and the updated v2 proposed in Lowe et al. (2017b). v2 of the corpus contains an updated procedure for generating the test set and fixes some other inconsistencies. In this chapter, we describe the Ubuntu Dialogue Corpus v2.

### 3.3.2 Ubuntu Chat Logs

The Ubuntu Chat Logs refer to a collection of logs from Ubuntu-related chat rooms on the Freenode Internet Relay Chat (IRC) network. This protocol allows for real-time chat between a large number of participants. Each chat room, or channel, has a particular topic, and every channel participant can see all the messages posted in a given channel. Many of these channels are used for obtaining technical support with various Ubuntu issues.

As the contents of each channel are moderated, most interactions follow a similar pattern. A new user joins the channel, and asks a general question about a problem they are having with Ubuntu. Then, another more experienced user replies with a potential solution, after first addressing the 'username' of the first user. This is called a *name mention* (Uthus and Aha, 2013), and is done to avoid confusion in the channel — at any given time during the day, there can be between 1 and 20 simultaneous conversations happening in some channels. In the most popular channels, there is almost never a time when only one conversation is occurring; this renders it particularly problematic to extract dyadic dialogues. A conversation between a pair of users generally stops when the problem has been solved, though some users occasionally continue to discuss a topic not related to Ubuntu.

Despite the nature of the chat room being a constant stream of messages from multiple users, it is through the fairly rigid structure in the messages that we can extract the dialogues between users. Figures 3.1 and 3.2 show an example chat room conversation from the #ubuntu channel as well as the extracted dialogues, which illustrates how users usually state the username of the intended message recipient before writing their reply (we refer to all initial questions and replies as 'utterances'). For example, it is clear that users 'Taru' and 'kuja' are engaged in a dialogue, as are users 'Old' and 'bur[n]er', while user '_pm' is asking an initial question, and 'LiveCD' is perhaps elaborating on a previous comment.

| Time | User | Utterance |
|---|---|---|
| 03:44 | Old | I dont run graphical ubuntu, I run ubuntu server. |
| 03:45 | kuja | Taru: Haha sucker. |
| 03:45 | Taru | Kuja: ? |
| 03:45 | bur[n]er | Old: you can use “ps ax" and “kill (PID#)" |
| 03:45 | kuja | Taru: Anyways, you made the changes right? |
| 03:45 | Taru | Kuja: Yes. |
| 03:45 | LiveCD | or killall speedlink |
| 03:45 | kuja | Taru: Then from the terminal type: sudo apt-get update |
| 03:46 | _pm | if i install the beta version, how can i update it when the final version comes out? |
| 03:46 | Taru | Kuja: I did. |

| Sender | Recipient | Utterance |
|---|---|---|
| Old | | I dont run graphical ubuntu, I run ubuntu server. |
| bur[n]er | Old | you can use “ps ax" and “kill (PID#)" |
| kuja | Taru | Haha sucker. |
| Taru | Kuja | ? |
| kuja | Taru | Anyways, you made the changes right? |
| Taru | Kuja | Yes. |
| kuja | Taru | Then from the terminal type: sudo apt-get update |
| Taru | Kuja | I did. |

FIGURE 3.1: Example chat room conversation from the #ubuntu channel of the Ubuntu Chat Logs (left), with the disentangled conversations for the Ubuntu Dialogue Corpus (right).

| Time | User | Utterance |
|---|---|---|
| [12:21] | dell | well, can I move the drives? |
| [12:21] | cucho | dell: ah not like that |
| [12:21] | RC | dell: you can’t move the drives |
| [12:21] | RC | dell: definitely not |
| [12:21] | dell | ok |
| [12:21] | dell | lol |
| [12:21] | RC | this is the problem with RAID:) |
| [12:21] | dell | RC haha yeah |
| [12:22] | dell | cucho, I guess I could just get an enclosure and copy via USB... |
| [12:22] | cucho | dell: i would advise you to get the disk |

| Sender | Recipient | Utterance |
|---|---|---|
| dell | | well, can I move the drives? |
| cucho | dell | ah not like that |
| dell | cucho | I guess I could just get an enclosure and copy via USB |
| cucho | dell | i would advise you to get the disk |
| dell | | well, can I move the drives? |
| RC | dell | you can’t move the drives. definitely not. this is the problem with RAID :) |
| dell | RC | haha yeah |

FIGURE 3.2: Example of before (left) and after (right) the algorithm adds and concatenates utterances in dialogue extraction. Since `RC` only addresses `dell`, all of his utterances are added, however this is not done for `dell` as he addresses both `RC` and `cucho`.

### 3.3.3 Dataset Creation

In order to create the Ubuntu Dialogue Corpus, first a method had to be devised to extract dyadic dialogues from the chat room multi-party conversations. The first step was to separate every message into 4-tuples of *(time, sender, recipient, utterance)*. Given these 4-tuples, it is straightforward to group all tuples where there is a matching sender and recipient. Although it is easy to separate the time and the sender from the rest, finding the intended recipient of the message is not always trivial.

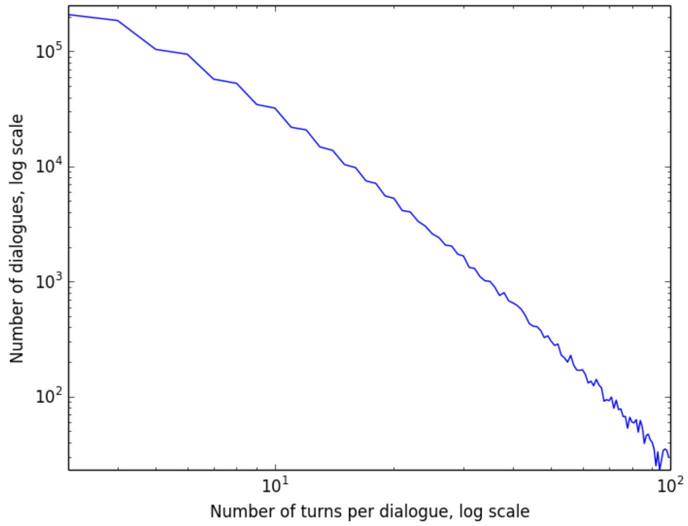



FIGURE 3.3: Plot of number of conversations with a given number of turns. Both axes use a log scale.

#### 3.3.3.1 Recipient Identification

While in most cases the recipient is the first word of the utterance, it is sometimes located at the end, or not at all in the case of initial questions. Furthermore, some users choose names corresponding to common English words, such as 'the' or 'stop', which could lead to many false positives. In order to solve this issue, we create a dictionary of usernames from the current and previous days, and compare the first word of each utterance to its entries. If a match is found, and the word does not correspond to a very common English word[1], it is assumed that this user was the intended recipient of the message. If no matches are found, it is assumed that the message was an initial question, and the recipient value is left empty.

#### 3.3.3.2 Utterance Creation

The dialogue extraction algorithm works backwards from the first response to find the initial question that was replied to, within a time frame of 3 minutes. A first response is identified by the presence of a recipient name (someone from the recent conversation history). The initial question is identified to be the most recent utterance by the recipient identified in the first response.

[1]We use the GNU Aspell spell checking dictionary.

All utterances that do not qualify as a first response or an initial question are discarded; initial questions that do not generate any response are also discarded. We additionally discard conversations longer than five utterances where one user says more than 80% of the utterances, as these are typically not representative of real chat dialogues. Finally, we consider only extracted dialogues that consist of 3 turns or more to encourage the modeling of longer-term dependencies.

To alleviate the problem of 'holes' in the dialogue, where one user does not address the other explicitly, as in Figure 3.2, we check whether each user talks to someone else for the duration of their conversation. If not, all non-addressed utterances are added to the dialogue. An example conversation along with the extracted dialogues is shown in Figure 3.2. Note that we also concatenate all consecutive utterances from a given user.

We do not apply any further pre-processing (e.g. tokenization, stemming) to the data as released in the Ubuntu Dialogue Corpus. However the use of pre-processing is standard for most NLP systems, and was also used in our analysis (see Section 3.4).

#### 3.3.3.3 Special Cases and Limitations

It is often the case that a user will post an initial question, and multiple people will respond to it with different answers. In this instance, each conversation between the first user and the user who replied is treated as a separate dialogue. This has the unfortunate side-effect of having the initial question appear multiple times in several dialogues. However the number of such cases is sufficiently small compared to the size of the dataset.

Another issue to note is that the utterance posting time is not considered for segmenting conversations between two users. Even if two users have a conversation that spans multiple hours, or even days, this is treated as a single dialogue. However, such dialogues are rare. We include the posting time in the corpus so that other researchers may filter as desired.

| | |
|---|---|
| # dialogues (human-human) | 936,000 |
| # utterances (in total) | 7,100,000 |
| # words (in total) | 100,000,000 |
| Min. # turns per dialogue | 3 |
| Avg. # turns per dialogue | 7.71 |
| Avg. # words per utterance | 10.34 |
| Median conversation length (min) | 6 |
| Training set dialogues | 898,000 |
| Validation/test set dialogues | 19,000 |
| Training set examples | unspecified |

TABLE 3.1: Properties of Ubuntu Dialogue Corpus. Note that any number of training examples can be specified during creation of the training set. Depending on the desired number of examples, multiple passes are made through the dataset, where each pass samples a new context stochastically from each dialogue. Very large training sets are possible, yet they will have overlapping examples.

### 3.3.4 Dataset Statistics

Table 3.1 summarizes properties of the Ubuntu Dialogue Corpus. One of the most important features of the Ubuntu chat logs is its size. This is crucial for research into building dialogue managers based on neural architectures. Another important characteristic is the number of turns in these dialogues. The distribution of the number of turns is shown in Figure 3.3. It can be seen that the number of dialogues and turns per dialogue follow an approximate power law relationship.

### 3.3.5 Test Set Generation

We set aside 2% of the Ubuntu Dialogue Corpus conversations to form a test set that can be used for evaluation[2]. Compared to the rest of the corpus, for evaluating retrieval models this test set has been further processed to extract a pair of *(context, response, flag)* triples from each dialogue. The *flag* is a Boolean variable indicating whether or not the response was the actual next utterance

[2]Note that, contrary to the original Ubuntu Dialogue Corpus, the updated version separates the training, validation, and test sets by time. That is, the training set consists of conversations that started from 2004 to approximately April 27, 2012; the validation set consists of dialogues starting from April 27 to August 7, 2012; and the test set has dialogues from August 7 to December 1, 2012. This mimics the training of dialogue systems in practice, where we only have access to data in the past, and want to answer user queries in the future.

after the given context. The *response* is a target (output) utterance which we aim to correctly identify. The *context* consists of the sequence of utterances appearing in the conversation prior to the response.

We create a pair of triples, where one triple contains the correct response (i.e. the actual next utterance in the dialogue), and the other triple contains a false response, sampled randomly from elsewhere within the test set. The flag is set to 1 in the first case and to 0 in the second case. An example pair is shown in Table 3.2. To make the task harder, we can move from pairs of responses (one correct, one incorrect) to a larger set of wrong responses (all with flag=0). In our experiments below, we consider both the case of 1 wrong response and 10 wrong responses. For evaluating generative models, rather than constructing *(context, response, flag)* triples, we take *(context, reference response)* pairs (i.e. where flag=1), and compare the model-generated response to the reference response.

| Context | Response | Flag |
|---|---|---|
| well, can I move the drives? __eot__ ah not like that | I guess I could just get an enclosure and copy via USB | 1 |
| well, can I move the drives? __eot__ ah not like that | you can use “ps ax" and “kill (PID #)" | 0 |

TABLE 3.2: Test set example with (context, reply, flag) format. The ‘__eot__’ tag is used to denote the end of a user’s turn within the context, and the ‘__eou__’ tag is used to denote the end of a user utterance without a change of turn.

Since we want to learn to predict all parts of a conversation, as opposed to only the closing statement, we consider various portions of context for the conversations in the test set. The context size is determined stochastically by uniform sampling[3]:

$$c = \sim Unif(2, t-1).$$

Here, parameter $t$ is the actual length of that dialogue (thus the constraint that $c \leq t-1$). In practice, this leads to short test dialogues having short contexts, while longer dialogues are often broken into a combination of short, medium, and long contexts.

[3]Note that this is a different formula than the original Ubuntu Dialogue Corpus, which sampled from a decreasing distribution. The new formula is simpler and leads to longer sampled contexts, which we consider desirable.

Note that, except for the plot in Figure 3.6, all experiments, results, and analysis in this paper will refer to the updated Ubuntu Dialogue Corpus v2.

## 3.4 Response Classification Architectures

To provide further evidence of the value of our dataset for research into neural architectures for dialogue managers, we provide performance benchmarks using two different training and evaluation criteria: response classification, and response generation.

We first consider *response classification* architectures, which attempt to distinguish between valid and invalid next responses to the context of a conversation. These are trained on the task of best response selection, which we call next utterance classification (NUC). This can be achieved by processing the data as described in Section 3.3.5, without requiring any human labels. This classification task is an adaptation of the recall and precision metrics previously applied to dialogue datasets (Schatzmann et al., 2005).

Note that retrieval models trained on the task of NUC are still end-to-end, as the natural language understanding, dialogue planning, and generation modules are combined, and the system is learned with a single supervision signal. These models can be used to 'generate' the next utterance in a conversation by retrieving the most probable next utterance from the entire training set, given the context. Thus, we can also evaluate these models using several generative metrics, that compare the selected response to the ground-truth response. We carry this out in Section 3.4.4.

We consider one naive model and two neural network-based retrieval models. The approaches considered are: TF-IDF, and models using Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM). Prior to applying each method, we perform standard pre-processing of the data using the NLTK[4] library and Twitter tokenizer[5] to parse each utterance. We use generic tags for various word categories, such as names, locations, organizations, URLs, and system paths.

[4] `www.nltk.org/`
[5] `http://www.ark.cs.cmu.edu/TweetNLP/`

To train the RNN and LSTM architectures, we process the full training Ubuntu Dialogue Corpus into the same format as the test set described in Section 3.3.5, extracting *(context, response, flag)* triples from dialogues. For the training set, we sample the responses in the same way described in Section 3.3.5. One can generate any number of training examples by iterating several times through the training data. Negative responses are selected at random from the rest of the training data. We note that for all models presented in this paper, the entire context of the dialogue that is available (i.e. after the context length sampling procedure in Section 3.3.5 used to create the dataset) is taken into account, and not just the most recent utterance. For the response classification architectures, this is done by concatenating all context utterances together.

We note that the models proposed below do not explicitly take into account ordinal information. The reasons for doing this are two-fold. First, training neural networks using classification for ranking tasks is well-established in the literature (Bordes et al., 2014), and is both simple to implement and effective in practice. Second, in the Ubuntu Dialogue Corpus we do not have supervised ordinal data for the relative quality of next responses given a context. More advanced methods could consider some way to approximate this ordinal information, such that a neural network model could be explicitly trained as a ranking system; however, this is beyond the scope of this paper.

### 3.4.1 TF-IDF

Term frequency-inverse document frequency is a statistic that intends to capture how important a given word is to some document, which in our case is the context (Ramos, 2003). It is a technique often used in document classification and information retrieval. The 'term-frequency' term is simply a count of the number of times a word appears in a given context, while the 'inverse document frequency' term puts a penalty on how often the word appears elsewhere in the corpus. The final score is calculated as the product of these two terms, and has the form:

$$\mathrm{tfidf}(w,d,D) = f(w,d) \times \log \frac{|D|}{|\{d \in D : w \in d\}|}, \tag{3.1}$$

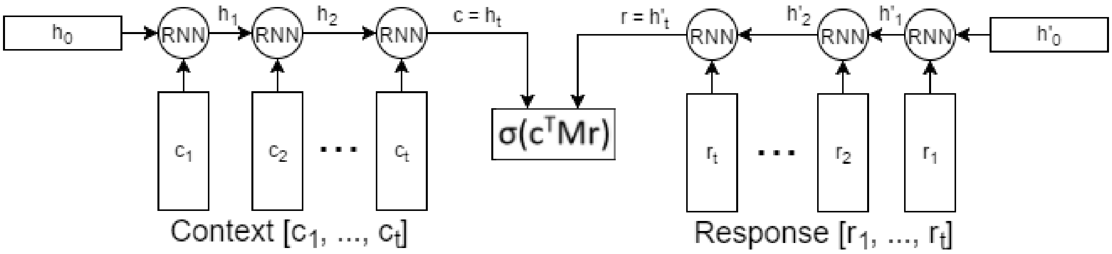



FIGURE 3.4: Diagram of the Dual Encoder (DE) model. The RNNs have tied weights. $c, r$ are the last hidden states from the RNNs. $c_i, r_i$ are word vectors for the context and response, $i < t$. We consider contexts up to a maximum of $t = 160$.

where $f(w, d)$ indicates the number of times word $w$ appeared in context $d$ and the denominator represents the number of dialogues in which the word $w$ appears. The TF-IDF vectors are constructed by calculating the TF-IDF score for each word $w$ in the vocabulary.

For classification, the TF-IDF vectors are first calculated for the context and each of the candidate responses. Given a set of candidate response vectors, the one with the highest cosine similarity to the context vector is selected as the output. For Recall@k, the top k responses are returned.

### 3.4.2 Dual Encoder

RNNs have been the primary building block of many current neural models for language-related tasks (Sutskever et al., 2014; Sordoni et al., 2015b), which use RNNs as encoders and decoders. We detail such models in Section 2.3.3. However, in this section, we are concerned with classification of responses, and thus using a decoder RNN for generation is not strictly necessary (and is not used in the model shown in Figure 3.4). In this section we build upon the approach in (Bordes et al., 2014), which has also been recently applied to the problem of question answering (Yu et al., 2014), and use RNNs for classification rather than generation.

We use a siamese network consisting of two RNNs with tied weights to produce the embeddings for the context and response, that we call the Dual-Encoder (DE) model. Given some input context and response, we compute their embeddings — $c, r \in \mathbb{R}^d$, respectively — by feeding the word embeddings one at a time into its respective RNN. Word embeddings are initialized using the pre-trained vectors (Common Crawl, 840B tokens from (Pennington et al., 2014)), and fine-tuned during training. The hidden state of the RNN is updated at each step, and the final hidden state represents a *summary* of the input utterance. Using the final hidden states from both RNNs, we then calculate the probability that this is a valid pair:

$$p(\text{flag} = 1|c, r, M) = \sigma(c^T M r + b), \tag{3.2}$$

where the bias $b$ and the matrix $M \in \mathbb{R}^{d \times d}$ are learned model parameters. This can be thought of as a generative approach; given some input response, we generate a context with the product $c' = Mr$, and measure the similarity to the actual context using the dot product. This is converted to a probability with the sigmoid function. The model is trained by minimizing the cross entropy of all labeled (context, response) pairs (Yu et al., 2014):

$$\mathcal{L} = -\sum_n \log p(\text{flag}_n|c_n, r_n, M) \tag{3.3}$$

A diagram of the DE model can be seen in Figure 3.4.

For training, we used a 1:1 ratio between true responses (flag = 1), and negative responses (flag=0) drawn randomly from elsewhere in the training set. The RNN architecture is set to 1 hidden layer with 100 neurons (optimized over $\{10, 50, 100, 200, 300\}$), and a learning rate of 0.0001 (optimized over $\{0.1, 0.01, 0.001, 0.0001\}$). The $W^h$ matrix is initialized using orthogonal weights (Saxe et al., 2013), while $W^x$ is initialized using a uniform distribution with values between -0.01 and 0.01. We use the first-order stochastic gradient optimization procedure Adam (Kingma and Ba, 2014) with the default parameters, using gradients clipped to 10 and a batch size of 512 (optimized over $\{128, 256, 512\}$). We found that weight initialization as well as the choice of optimizer were critical for training the RNNs.

In addition to the RNN model, we consider the same architecture but change the hidden units to long-short term memory (LSTM) units (Hochreiter and Schmidhuber, 1997), as described in Section 2.2.4. For this model, we used 1 hidden layer with 200 neurons, a learning rate of 0.001, and a batch size of 256 (optimized over the same values as the RNN). We again use the default Adam settings, and initialize the forget gate bias of the LSTM to 2.0. The hyper-parameter configuration (including number of neurons) was optimized independently for RNNs and LSTMs using a separate validation set extracted from the training data.

### 3.4.3 Evaluation Metrics

We consider two types of evaluation metrics: *retrieval* metrics, and *generative* metrics. These metrics are applicable to both models trained on the task of NUC, detailed in this section, and the generative models introduced in Section 3.5. In particular, they offer two ways of automatically evaluating dialogue systems trained in an end-to-end manner.

For retrieval, we evaluate using Recall@k (denoted R@1 R@2, R@5 below), which has often been used in language tasks. Here the agent is asked to select the $k$ most likely responses, and it is correct if the true response is among these $k$ candidates. Only the R@1 metric is relevant in the case of binary classification (as in the Table 3.2 example). Although a language model that performs well on these retrieval metrics is not guaranteed to achieve good performance on utterance generation, we hypothesize that improvements on a model with regards to the classification task will eventually lead to improvements for the generation task. See Section 3.6 for further discussion of this point.

We also consider generative metrics that compare the generated or retrieved utterance to the ground-truth next utterance. In general, this is a hard open problem (Liu et al., 2016). We use methods based on word embeddings that have recently been proposed for use in evaluating non-task-oriented dialogue systems, when no task completion signal is available. Methods such as Word2Vec (Mikolov et al., 2013b) calculate these embeddings using distributional semantics; that is, they approximate the meaning of a word by considering how often it co-occurs with other words in the corpus.[6] These embedding-based metrics usually approximate sentence-level embeddings using some heuristic to combine the vectors of the individual words in the sentence. The sentence-level embeddings between the candidate and target response are compared using a measure such as cosine distance.

[6] To maintain statistical independence between the task and each performance metric, it is important that the word embeddings used are trained on corpora which do not overlap with the task corpus.

We note that these metrics do not necessarily correlate strongly with human judgement (Liu et al., 2016); here, we consider them to be measures of the topicality of the retrieved responses. If the generated response and ground-truth response are semantically similar, then the vector-based metrics should be higher, as word embeddings themselves contain semantic information (Mikolov et al., 2013b). It is because of this interpretation that we prefer the vector-based metrics over word-overlap metrics such as BLEU.

**Greedy Matching** Greedy matching is the one embedding-based metric that does not compute sentence-level embeddings. Instead, given two sequences $r$ and $\hat{r}$, each token $w \in r$ is greedily matched with a token $\hat{w} \in \hat{r}$ based on the cosine similarity of their word embeddings ($e_w$), and the total score is then averaged across all words:

$$G(r,\hat{r}) = \frac{\sum_{w \in r;} \max_{\hat{w} \in \hat{r}} cos\_sim(e_w, e_{\hat{w}})}{|r|}$$
$$GM(r,\hat{r}) = \frac{G(r,\hat{r}) + G(\hat{r}, r)}{2}$$

This formula is asymmetric, thus we must average the greedy matching scores $G$ in each direction. This was originally introduced for intelligent tutoring systems (Rus and Lintean, 2012). The greedy approach favours responses with key words that are semantically similar to those in the ground truth response.

**Embedding Average** The embedding average metric calculates sentence-level embeddings using additive composition, a method for computing the meanings of phrases by averaging the vector representations of their constituent words (Foltz et al., 1998; Landauer and Dumais, 1997; Mitchell and Lapata, 2008). This method has been widely used in other domains, for example in textual similarity tasks (Wieting et al., 2015). The embedding average, $\bar{e}$, is defined as the mean of the word embeddings of each token in a sentence $r$:

$$\bar{e}_r = \frac{\sum_{w \in r} e_w}{|\sum_{w' \in r} e_{w'}|}.$$

To compare a ground truth response $r$ and retrieved response $\hat{r}$, we compute the cosine similarity between their respective sentence level embeddings: EA := $\cos(\bar{e}_r, \bar{e}_{\hat{r}})$.

**Vector Extrema** Another way to calculate sentence-level embeddings is using vector extrema (Forgues et al., 2014). For each dimension of the word vectors, take the most extreme value amongst *all word vectors in the sentence*, and use that value in the sentence-level embedding:

$$e_{rd} = \begin{cases} \max_{w \in r} e_{wd} & \text{if } e_{wd} > |\min_{w' \in r} e_{w'd}| \\ \min_{w \in r} e_{wd} & \text{otherwise} \end{cases}$$

where $d$ indexes the dimensions of a vector; $e_{wd}$ is the $d$'th dimensions of $e_w$ ($w$'s embedding). The min in this equation refers to the selection of the largest negative value, if it has a greater magnitude than the largest positive value.

Similarity between response vectors is again computed using cosine distance. Intuitively, this approach prioritizes informative words over common ones; words that appear in similar contexts will be close together in the vector space. Thus, common words are pulled towards the origin because they occur in various contexts, while words carrying important semantic information will lie further away. By taking the extrema along each dimension, we are thus more likely to ignore common words.

### 3.4.4 Experimental Results

We examine the performance of the models using both retrieval and vector based metrics, as shown in Tables 3.3 and 3.4. For NUC, the models were evaluated using both 1 (1 in 2) and 9 (1 in 10) false examples.[7]

| | **Retrieval Metrics** | | | |
|---|---|---|---|---|
| Method | 1 in 2 R@1 | 1 in 10 R@1 | 1 in 10 R@2 | 1 in 10 R@5 |
| TF-IDF | 74.9% | 48.8% | 58.7% | 76.3% |
| Dual Encoder (RNN) | 77.7% | 37.9% | 56.1% | 83.6% |
| Dual Encoder (LSTM) | **86.9%** | **55.2%** | **72.1%** | **92.4%** |

TABLE 3.3: Results for the three algorithms using various recall measures for binary (1 in 2) and 1 in 10 (1 in 10) next utterance classification %.

We observe that the Dual Encoder with LSTM units outperforms both the Dual Encoder with RNN units and TF-IDF on all evaluation metrics. It is interesting to note that TF-IDF actually outperforms the RNN on the Recall@1 case for the 1 in 10 classification. This is most likely due to the limited ability of the RNN

[7]The performance metrics Recall@2 and Recall@5 are not relevant in the binary classification case.

| | Generative Metrics | | |
|---|---|---|---|
| Method | Embedding Average | Greedy Matching | Vector Extrema |
| TF-IDF | 0.536 | 0.370 | 0.342 |
| Dual Encoder (LSTM) | **0.650** | **0.413** | **0.376** |

TABLE 3.4: Results for TF-IDF and the DE model with LSTM units on the embedding average, greedy matching, and vector extrema scores. These scores provide an estimate of the topic consistency of the generated responses.

| Context |
|---|
| “any apache hax around ? i just deleted all of __path__ - which package provides it ?", “reconfiguring apache do n’t solve it ?" |

| Ranked Responses | Flag |
|---|---|
| 1. “does n’t seem to, no" | 1 |
| 2. “you can log in but not transfer files ?" | 0 |

FIGURE 3.5: Example showing the ranked responses from the LSTM. Each utterance is shown after pre-processing steps.

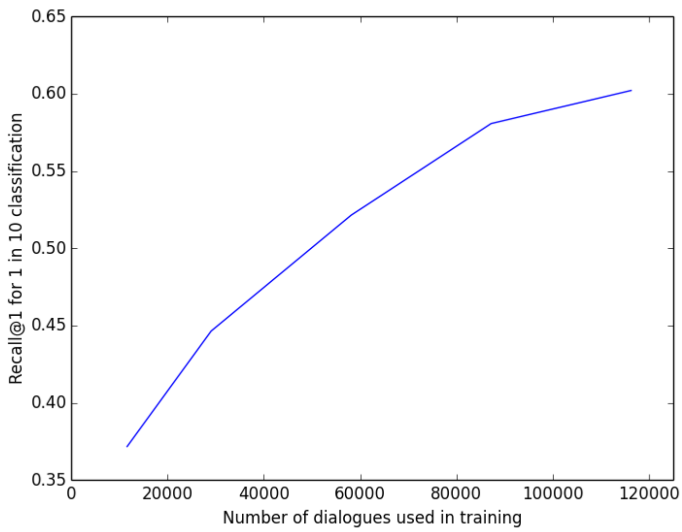



FIGURE 3.6: The LSTM (with 200 hidden units), showing Recall@1 for the 1 in 10 classification, with increasing dataset sizes up to 120k dialogues. Note that this was calculated using the old version of the Ubuntu Dialogue Corpus, and thus the Recall@1 values are higher than those in Table 3.3.

to take into account long contexts, which can be overcome by using the LSTM. An example output of the LSTM where the response is correctly classified is shown in Figure 3.5.

We also show, in Figure 3.6, the increase in performance of the LSTM as the amount of data used for training increases. This confirms the importance of having a large training set.

### 3.4.5 Qualitative Error Analysis

There are a large number of technical challenges that must be solved in order to construct a system that can provide adequate responses in a dialogue. In fact, almost all common challenges in natural language processing are present in some form or another in the dialogue problem. These include, but are not limited to: coreference resolution, lexical semantics, discourse coherence and cohesion, natural language understanding, natural language generation, compositional semantics, the turn taking structure of dialogue, and more. Further, it is often necessary to have some technical knowledge about the subject matter being discussed. It is clear that current end-to-end dialogue systems are not able to adequately address all these problems, yet precisely which aspects of conversation are the most prevalent sources of errors remains relatively unknown. This is particularly true for neural network models for dialogue, which have only recently come into prominence.

We undertake the task of evaluating an end-to-end dialogue system, the DE model with LSTM units, on the Ubuntu Dialogue Corpus for the NUC task. We hope that an understanding of the most common errors made by this model can help inform future work on neural dialogue systems, particularly on the Ubuntu Dialogue Corpus.

We conduct an error analysis with three participants[8] evaluating a total of 100 randomly chosen errors made by the Dual Encoder. For each error made by the model[9], we consider what abilities the model would need to have in order to answer the question correctly. We classify these into several categories: using knowledge, understanding tone and style of the responses, a better understanding of the semantic similarity of phrases, and explicitly considering the turn-taking nature of dialogue. Since we are evaluating a classification model, we do not consider problems associated with natural language generation. Note that an error can be classified into multiple categories, if they are each necessary to answer the question correctly.

[8]Participants were graduate students in computer science, who had familiarity with both dialogue systems and the Ubuntu domain.

[9]We consider an error to be any example where the correct response is not the top 1 response ranked by the model.

In addition to classifying the errors made by the model, we qualitatively evaluate the difficulty of the questions on a scale from 1-5. A rating of 1 on the difficulty scale means that the question is easily answerable by all humans. A 2 indicates moderate difficulty, which should still be answerable by all humans but only if they are paying attention. A 3 means that the question is fairly challenging, and may either require some familiarity with Ubuntu or the human respondent paying very close attention to answer correctly. A 4 is very hard, usually meaning that there are other responses that are nearly as good as the true response; many humans would be unable to answer questions of difficulty 4 correctly. A 5 means that the question is effectively impossible: either the true response is completely unrelated to the context, or it is very short and generic.

Finally, we evaluate the appropriateness of the response chosen by the model for each question on a scale from 1-3. A score of 1 indicates that the chosen response is completely unreasonable given the context. A 2 means that the response chosen was somewhat reasonable, and that it's possible for a human to make a similar mistake. A 3 means that the model's response was more suited to the context than the actual response.

We note that such an analysis is partially dependent on the model and the domain. The end-to-end system from Wen et al. (2016) achieves very strong performance, and thus would not face exactly the same problems as the models we present here. However, this is because the model was trained on a very narrow dataset of restaurant recommendations, and thus the space of generated responses is comparatively small. We believe that other conversational models trained on large, complex datasets are likely to encounter the same problems that we present here.

In the Ubuntu domain, questions where using **external knowledge** would be helpful for the model involve technical terminology. In most cases, the correct response contains the name of a command or process that is related to one stated in the context; however, the model is unable to link the two together. An example of such a question is shown in Figure 3.7. In this case, the context of the conversation is about file searching in Ubuntu, and the correct response (in italics) mentions the locate command. This response would have been assigned a higher probability if it was able to determine the meaning of the locate command.

| | |
|---|---|
| **Context:** | Speaker A: is there anything I can do to make ubuntus filesearching faster ? I am running from an ssd and it 's still painfully slow __eou__<br>Speaker B: searching how ? __eou__<br>Speaker A: hitting the search button from nautilus __eou__ searching systemwide __eou__ |
| **Binary Probability** | **Candidate Responses** |
| *0.10* | *i tend to use the locate command . __eou__* |
| 0.38 | I'm not that into it , but it has to be session in one , or track in one or something to have the rw funtion __eou__ |
| 0.06 | Np __eou__ |
| **0.56** | **probably just junky firefox __eou__ I bet you have a tonne of addons __eou__ that all takes resources __eou__ what other apps are you running ? __eou__ how much ram frees after you close skype ( if its convenient )** |
| 0.48 | except I get an invite __eou__ |
| 0.44 | installing from source on ubuntu isn't a great idea IMO . but look for a make uninstall option __eou__ |
| 0.16 | oh i get it , thanks a lot __eou__ |
| 0.21 | the python one __eou__ i think cron may be able to do that .. to restart a task if it dies out prematurely __eou__ well you can show your #python script and people may suggest the best way to *overcome and premature *unknown** .. __eou__ if it 's a buggy script then you'd expect it to be very problematic with anything starting it __eou__ |
| 0.22 | or killall ftl* __eou__ |
| 0.54 | any help with custom msg __eou__ |

FIGURE 3.7: Example where the model would benefit from using an external knowledge source. Correct answer is in italics, and the model's selected answer is in bold. Note that the probabilities do not sum up to 1, as they are binary probabilities – the model considers each candidate response independently.

There are other examples where the model may be incapable of taking into account the specific **tone or style** of the users in the conversation. For instance, a speaker may use many emoticons, have poor English grammar skills and be prone to misspelling words, use frequent abbreviations, or use a particularly formal tone. Being able to spot these distinct language features could lead the model to improve its performance in terms of selecting the actual next response. An example of this is shown in Figure 3.8. In the context, Speaker A appends his question with an (unnecessary) smiley face. Thus, it is more likely that the candidate response with multiple smiley faces is the correct response.

One of the most important challenges in natural language is understanding the semantics of phrases. Classification dialogue models can make errors due to an inability to detect **semantic similarities** between sentences, or due to the detection of spurious similarities. This category covers the general case where the topic of the model's response is clearly different than the topic of the context and true response. In this category, we also define two special cases: one where there is a direct **word copying** between the context and true response that the model failed to detect, and one where some **high-level inference** is required to answer correctly. An example of the latter case is shown in Figure 3.9; Speaker A asks how to install some programs automatically, and the correct response states that they had previously been 'doing it manually'. Thus, if the model was

| | |
|---|---|
| **Context:** | Speaker A: Whats the best RDP software for Ubuntu ? I want to be able to RDP into my ubuntu desktop from my ubuntu laptop :) __eou__<br>Speaker B: Then just use VNC . __eou__ |
| **Binary Probability** | **Candidate Responses** |
| *0.15* | *what software , do you have any links to show how to to it ? : ) __eou__ ur a beast ! TY again :) __eou__* |
| 0.04 | wrong place stop it __eou__ |
| 0.33 | could use puppet :) __eou__ |
| 0.29 | lol __eou__ |
| 0.17 | I've installed mine from " Additional drivers " __eou__ |
| 0.36 | xP yeah that its been a long time since i last used my vpn server __eou__ |
| 0.00 | yes , and where does it say it 's released , or you can buy it , in actually anything about it __eou__ **unknown** 's not released __eou__ it 's a concept canonical are working on / trying to create __eou__ |
| **0.75** | **it 's that " persistence " stuff ? what do you mean ? __eou__** |
| 0.00 | sudo chown root : root /tmp && sudo chmod 1777 /tmp __eou__ |
| 0.00 | python __eou__ python __eou__ |

FIGURE 3.8: Example where the model would benefit from understanding the tone and style of the speakers. Correct answer is in italics, and the model's selected answer is in bold.

| | |
|---|---|
| **Context:** | Speaker A: how do I move programs and all their dependencies automatically ? __eou__<br>Speaker B: you mean between two Ubuntu installations ? __eou__ |
| **Binary Probability** | **Candidate Responses** |
| *0.04* | *I have a chroot partition I've been using ldd and doing it manually but it is quite slow __eou__* |
| 0.00 | if your not a geek then you won't understand __eou__ would you advise your grandmother to try and install linux ? __eou__ |
| 0.02 | that 's where im stumped ... an older kernel made no difference , whilst an older release of Ubuntu did __eou__ |
| 0.71 | well , everything with indicators is basically dbus __eou__ |
| 0.01 | it 's cool that you help people who run free software ;) bye __eou__ |
| 0.00 | not by default , but it can . /var holds a lot of temp stuff like logs and debs , you don't need those cluttering your SSD and using write cycles __eou__ also , move your web cache to ramdisk to make it fast as well as not use your HDD at all :) __eou__ its a disk space ... in ram __eou__ |
| 0.59 | yes , but they speak http so I could use the Browser as a low-level access tool for browing repos and I would like to do that , but that doesn't seem to work . __eou__ |
| 0.61 | sure __eou__ |
| **0.97** | **I believe it uses gdm but I'm not sure . The login manager thing looks the same as the stock ubuntu 12.10 one __eou__** |
| 0.14 | ok , ty __eou__ |

FIGURE 3.9: Example where the model would benefit from the ability to conduct high-level inference to better understand the semantic similarity between context and correct response. Correct answer is in italics, and the model's selected answer is in bold.

able to infer that a person who has asked to perform an operation automatically could previously have been doing it manually, it would have assigned a higher probability to the correct response.

Finally, we consider errors where the model is unable to account for the **turn-taking structure** of dialogue. For the Ubuntu Dialogue Corpus, interactions between users usually take a certain form, where one user is asking for help and the other user is providing answers. Thus, it is important to consider the role of the current user when selecting the correct response; indeed, there has been preliminary work in this direction (Luan et al., 2016; Li et al., 2016a). We

| **Context:** | Speaker A: i can't seem to get audio working as a non-root user . has anyone ever had this problem ? __eou__<br>Speaker B: alsamixer to the rescue __eou__<br>Speaker A: alsamixer shows everything turned on , and looks exactly the same for my normal user as it does for root __eou__<br>Speaker B: no MM's? __eou__ |
|---|---|
| **Binary Probability** | **Candidate Responses** |
| *0.62* | *correct __eou__* |
| 0.92 | true but how will he find my new ip so easily if i get it changed ? All i do is programming c and check my mail usually __eou__ |
| 0.68 | yes strange . then omit the dash altogether , try giving set default sink :/ __eou__ |
| **0.93** | **thanks __eou__ where is the db app - i cannot locate it ( sorry to be such a noob ! ) __eou__** |
| 0.03 | I'm switching the location to my on board SSD drive that 's embedded to the laptops board . I just haven't been using the storage so I figure I could try and utilize the space while the ram being 8 gig 's itself I see no problem with the switch . Do you understand what I'm doing . I'm only asking here so I don't go screwing up and save myself hours of headaches __eou__ |
| 0.03 | you'll love it __eou__ I am joking , but you will probably enjoy learning about it __eou__ well it 's a step up from opening your hard drive up and using a magnet __eou__ |
| 0.44 | yeah , 512 is plenty __eou__ |
| 0.20 | i use it on a number of machines with no problems . just this one . __eou__ modprobe pulls up a variety of mouse drivers __eou__ |
| 0.62 | so the issue is **unknown** **unknown** . gz ' is different from the same file on the system " but i don't have any idea why/what that means , sorry . best of luck . __eou__ |
| 0.89 | http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-680m comes with optimus technology . so i think it has an onboard intel card __eou__ |

FIGURE 3.10: Example where the model selects an inappropriate response to a question. Correct answer is in italics, and the model's selected answer is in bold.

also consider a special case of this error, when the last utterance in the context **asks a question** and the response chosen by the model is not answering any question at all. For example, in Figure 3.10, the final utterance asks the question 'no MM's?'. The response selected by the model begins with 'thanks', which is clearly not a reasonable response to a question.

An example of the general turn-taking error is shown in Figure 3.11. This depicts a typical dialogue between two users in the Ubuntu Dialogue Corpus: Speaker A is having trouble with their brightness keys, and Speaker B is trying to help them. The model must predict the next response of Speaker A. In the first response, the user states that they are appreciative of the help being given, which fits with Speaker A's behaviour in the context; thus, it is more likely to be the correct response.

We examined 100 randomly selected errors of the DE model on the Ubuntu Corpus to compute the number of errors in each category; the results are shown in Table 3.12. We can first note that there is significant progress to be made for classification models on the Ubuntu Dialogue Corpus; over half (60%) of the errors made by the model can be considered feasible for the majority of humans (1-3 on the difficulty rating). However, the number of questions that

| | |
|---|---|
| **Context:** | Speaker A: hi __eou__ I have a problem with fn keys for brightness with my laptop and nvidia propertiary driver __eou__<br>Speaker B: what make and model laptop ? > __eou__<br>Speaker A: sony vaio vgn fz31z __eou__ and im using nvidia propertiary driver version current ( recommended one ) __eou__<br>Speaker B: try the boot option : acpi_backlight=vendor __eou__<br>Speaker A: i have added acpi backlight for vendor i have updated grub but the keys are not working __eou__ my grub<br>cmd line linxu default : quiet splash acpi_backlight=vendor __eou__<br>Speaker B: try the boot option : acpi_osi=LINUX __eou__<br>Speaker A: ok i must remove the acpi_backlihgt/ __eou__<br>Speaker B: I'd also report a bug __eou__ could try Quantal liveCD to see if the newer kernel plays nicer __eou__<br>Speaker A: I think that is a nvidia problem with the propertiary __eou__<br>Speaker B: possibly , or it could be acpi based __eou__<br>Speaker A: ok thank you i must remove the previous about the vendor ok ? __eou__<br>Speaker B: could try both and then just one __eou__ |
| **Binary Probability** | **Candidate Responses** |
| *0.38* | *thanks for the help . Trying now . Is there any other same bug report for vaio/ __eou__* |
| 0.20 | it 's actually ubuntu support , since i'm using ubuntu , isn't it ? __eou__ |
| 0.49 | yes __eou__ the usb disk will just be seen as a hard disk , install to it __eou__ |
| 0.50 | if you do unattended-upgrades -d , that might tell you a few things ? __eou__ |
| 0.58 | does this have ' open terminal here ' and ' 2pane mode ' options ? __eou__ found terminal option , just looking for 2pane __eou__ |
| **0.71** | **it 's cool** __eou__ |
| 0.02 | it 's like hotel internet __eou__ http://www.fdlinux.com/networksetuphowto.html __eou__ |
| 0.01 | I'll check what it means in google . Thank you . __eou__ |
| 0.36 | i never liked it ... for thin versions , i use fluxbox or some other window manager __eou__ |
| 0.49 | so do I just paste that code in to the beginning of the script ... ? __eou__ sorry experienced linux user , very very novice coder ;-P __eou__ |

FIGURE 3.11: Example where the model does not take into account the roles of the participants in the dialogue. Correct answer is in italics, and the model's selected answer is in bold.

| **Difficulty Rating** (1-5) | **Number of Errors** | **% of Errors** |
|---|---|---|
| Impossible (5) | 19 | 19% |
| Very difficult (4) | 21 | 21% |
| Difficult (3) | 22 | 22% |
| Moderate (2) | 25 | 25% |
| Easy (1) | 13 | 13% |
| **Model Response Rating** (1-3) | | |
| Very reasonable (3) | 14 | 14% |
| Somewhat reasonable (2) | 37 | 37% |
| Unreasonable (1) | 49 | 49% |
| **Error Category** | | |
| Tone and style | 8 | 9% |
| Knowledge | 18 | 20% |
| Semantic similarity | 45 | 49% |
| Word copying | 11 | 12% |
| High-level inference | 16 | 18% |
| Turn-taking structure | 20 | 22% |
| Answering questions | 6 | 7% |

FIGURE 3.12: Qualitative evaluation of the errors from the DE model. Note that counts for parent categories (semantic similarity and turn-taking structure) include the counts for the child categories. Error categories are not classified for impossible questions and are not mutually exclusive, thus totals may not add up to 100.

every human could answer unconditionally is small, as technical language can often be confusing for people who are unaccustomed to it. The other questions are roughly uniformly distributed over the remaining levels of difficulty, from moderate to impossible. We also note that there are a large number of cases (49%) where the response retrieved by the model was completely unreasonable given the context, which further indicates that there is room for improvement in these models.

It is also interesting to examine the distribution of errors across the examples. As can perhaps be expected, the most common form of error was a lack of understanding of the semantics of the responses. What is more surprising is that there is a significant number of examples where the model failed to observe that there was a key word shared between the context and the correct response; this could be because there are often common words between the context and false responses in the training set, and the model is unable to distinguish between words that are relatively unimportant and those that carry significant semantic meaning. Thus, there is much progress to be made in dialogue systems by working on the general problem of natural language understanding.

There are many examples where the model could be improved by explicitly accounting for the turn-taking structure of dialogue, as there were often instances where the model selected a response that was not suited to the current speaker. In several cases, the model also needed some form of external knowledge base in order to answer the question correctly. Note that the number of such examples in Table 3.12 refers to instances where the correct response mentions a Ubuntu term that is related but not identical to the terminology in the context; if this were to be extended to all questions where technical vocabulary is mentioned, the number would be significantly higher. Finally, there is a small number of cases where a better understanding of the tone of the dialogue would help the model, however this does not seem to be the best direction for future research.

## 3.5 Generative Response Architectures

In order to aid progress towards the goal of building fully generative conversational models, we present baseline models for generating responses conditioned

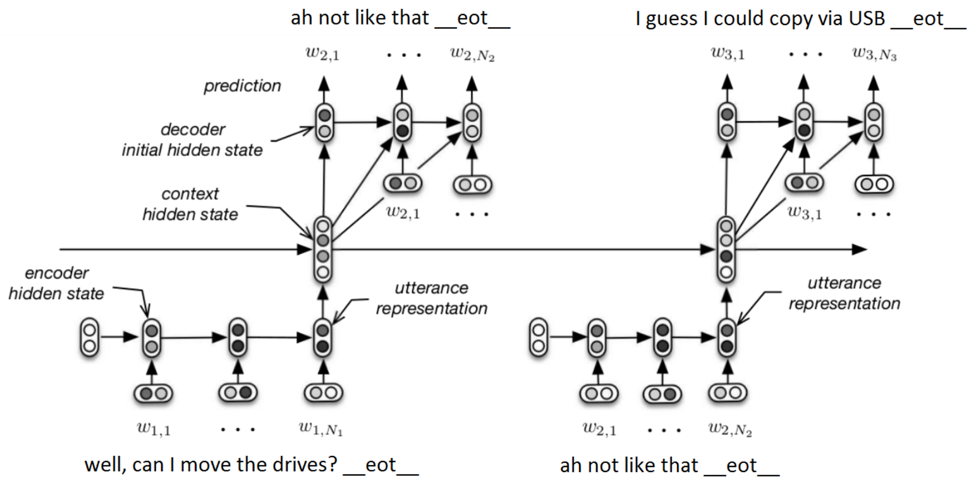



FIGURE 3.13: Diagram of the HRED model. Note that each utterance in the context is encoded with a separate 'utterance-level' encoder, which is then fed into a 'context-level' encoder.

on the context of the conversation for the Ubuntu Dialogue Corpus. It should be noted that the format of the dataset can easily be altered to support training in this manner: one can simply remove all *(context, response, flag)* triples with $flag = 0$, and be left with only the valid *(context,response)* pairs.

### 3.5.1 Generative Recurrent Neural Language Model

Our first model is a recurrent neural language model (RNN-LM), described in Section 2.3.2, which observes the dialogue word-by-word and updates its hidden state $h_t$ at time step (word index) $t$. Given a hidden state $h_t$ the model then outputs a probability distribution over all words in the vocabulary.

For dialogue response generation, the model is conditioned on the previous dialogue context and used to generate a response, i.e. the next utterance in the dialogue. Such a model could be used as a full dialogue system, as defined in Section 2.4.3, to carry out a conversation with a user.

### 3.5.2 Hierarchical Recurrent Encoder-Decoder

One problem with directly applying a standard RNN language model to modeling dialogues is that it does not take into account the turn-taking nature of conversations. It is well known that recurrent neural networks have trouble learning long-term dependencies (Bengio et al., 1994), a problem only partially alleviated with LSTMs. Thus, if a long context is fed into the encoder, it is possible for the model to put a large weight on only the most recent utterance. In order to investigate models that are able to retain state over long conversations, we implement the recently proposed hierarchical recurrent encoder-decoder (HRED) Sordoni et al. (2015a). While this model was initially proposed for context-sensitive query suggestion, it has been adapted for dialogue response generation on a dataset of movie subtitles (Serban et al., 2016).

The HRED model builds on the traditional encoder-decoder model (Cho et al., 2014), described in Section 2.3.3. The main addition is a second encoder, the *utterance-level encoder*, that takes as input the fixed-length vectors produced by the lower-level encoder, which we refer to as the *word-level encoder*. Instead of letting the decoder take as input the fixed-length vectors from the word-level encoder, the decoder takes as input the output of the utterance-level encoder. Intuitively, the utterance-level encoder summarizes the history of the conversations into a single vector, which is more sensitive to previous utterances in the conversation. This provides a more powerful architecture as it is now possible for the model to encode order-dependent patterns inherent in the turn-taking nature of dialogue. As before, the model is trained end-to-end to maximize the log-likelihood of the generated utterance. A diagram of the HRED model can be seen in Figure 3.13.

To summarize: the RNN-LM (Figure 2.2) uses a single RNN (or LSTM) to encode the entire history of the dialogue, which consists of all utterances in the context concatenated together. It then uses the same RNN (i.e. an RNN with the same parameters) to decode the prediction utterance. The Encoder-Decoder model (not shown here) augments this with a second RNN with different parameters for the decoder. All context utterances are still concatenated in the encoder, and thus it is difficult to model long-term dependencies for utterances that occur earlier in the dialogue. This problem is alleviated with the HRED model (Figure 3.13), which does not concatenate the context utterances: each is encoded with a separate utterance-level encoder, whose output is fed into an additional context-level encoder. The output of the context-level encoder depends on all the utterances in the context, and is fed into the decoder. Each RNN has separate parameters.

| | Generative Metrics | | |
|---|---|---|---|
| | Embedding Average | Greedy Matching | Vector Extrema |
| LSTM-LM | 0.561 | 0.425 | 0.380 |
| HRED | 0.617 | **0.452** | **0.408** |
| TF-IDF | 0.536 | 0.370 | 0.342 |
| Dual Encoder (LSTM) | **0.650** | 0.413 | 0.376 |

TABLE 3.5: Results for both the generative and retrieval models on the embedding average, greedy matching, and vector extrema scores. These scores provide an estimate of the topic consistency of the generated responses.

| | Retrieval Metrics | | | |
|---|---|---|---|---|
| | 1 in 2 R@1 | 1 in 10 R@1 | 1 in 10 R@2 | 1 in 10 R@5 |
| LSTM-LM | 58.9% | 19.6% | 33.1% | 61.4% |
| HRED | **61.8%** | **21.5%** | **35.8%** | **64.5%** |
| TF-IDF | 74.9% | 48.8% | 58.7% | 76.3% |
| Dual Encoder (RNN) | 77.7% | 37.9% | 56.1% | 83.6% |
| Dual Encoder (LSTM) | 86.9% | 55.2% | 72.1% | 92.4% |
| MEMN2N (Dodge et al., 2015) | — | 63.72% | — | — |
| RNN-CNN (Baudiš and Šedivỳ, 2016) | 91.1% | 67.2% | 80.9% | 95.6% |
| Ensemble (Kadlec et al., 2015) | **91.5%** | **68.3%** | **81.8%** | **95.7%** |
| r-LSTM (Xu et al., 2016) | 88.9% | 64.9% | 78.5% | 93.2% |

TABLE 3.6: Results for both the generative and retrieval models using various recall measures for binary (1 in 2) and 1 in 10 (1 in 10) next utterance classification %. We include state-of-the-art results from more recent papers.

### 3.5.3 Experimental Results

We now examine the performance of the generative models using the vector-based metrics defined in Section 3.4.3. The results for the RNN language model with LSTM units (LSTM-LM) and the HRED model can be seen in Figure 3.5. As expected, the HRED model outperforms the baseline LSTM model across all of the metrics. However, it is interesting to note that in a direct comparison with the Dual Encoder model, the HRED model has a higher score in 2 out of the 3 metrics considered. Thus, it is likely that the HRED model is generating responses that are more semantically similar to the ground-truth response, and is better at staying on topic.

We also present the results for these models on the NUC task. It is possible to apply the generative models to the NUC task, as all that is required is the ability to assign probabilities to sequences of utterances. Again, the HRED model predictably outperforms the LSTM-LM model on all metrics. This coincides with the results of (Serban et al., 2016) on a different dataset, using different metrics. We can also see that the generative models perform much worse than the models explicitly trained to retrieve utterances from a list. This is to be expected, as the retrieval models were trained explicitly on the NUC task, while the generative models were not. Because of the discrepancy in training objectives, we do not recommend the use of NUC for comparing generative models with retrieval models. However, we believe that NUC is very useful for comparing generative models with other generative models, and retrieval models with retrieval models; we justify this further in Section 3.6.3.

### 3.5.4 Examples of Generated Responses

In order to obtain a better understanding of the quality of responses from the generative model, we provide a table of examples from the LSTM-LM and HRED models in Table 3.7. We chose several representative samples that provide insight into the behaviour, and the limitations, of the generative models.[10]

First, we note that there are several situations where the generative models, particularly the HRED model, produce reasonable responses. For instance, in example 8), the models are able to generate a coherent and useful response concerning the installation of `grub`. The models achieve this despite having no knowledge of Ubuntu other than what it has observed in the training corpus.

The HRED model is also able to generate reasonable responses when it is playing the role of the non-expert who is seeking help. This is the case in examples 1), 3), and 6). However, the responses from the model are fairly homogeneous; it is simply stating that it does not know how to solve the problem. This is a commonly observed behaviour in the generated samples of the model. Given a context, it is often reasonable to simply state 'I don't know', or 'I'm not sure', and indeed these are likely responses in the training set given many different contexts. This is the problem of *generic responses*, and has previously been observed in end-to-end generative dialogue models (Vinyals and Le, 2015).

There are other instances where the lack of semantic understanding is clear. In example 2), the HRED makes a relevant statement about the command 'sudo apt-get', stating that the user should try to run this command. However, it fails to understand that the last user utterance ('I tried that, it comes the same

[10]The full set of model samples are available at: www.iulianserban.com/Files/UbuntuDialogueCorpus.zip

| Context | Response |
|---|---|
| **1)** i set up my hd such that i have to type a passphrase to access it at boot. how can i remove that passwrd, and just boot up normal. i did this at install, it works fine, just tired of having reboots where i need to be at terminal to type passwd in. help? $\rightarrow$ backup your data, and re-install without encryption “ might " be the easiest method | **HRED:** how would i do that? **LSTM:** how do i do that? |
| **2)** hi, everyone. I download the googleearth. deb, and want to install it with : sudo dpkg -i something. deb, It tells me : google-earth-stable depends on lsb-core (>= 3.2 ) ; however : Package lsb-core is not installed. So i just install the lsb-core, But it comes error : <url> what should i do? $\rightarrow$ the error suggests running “ sudo apt-get -f install " $\rightarrow$ i tried that, it comes the same error | **HRED:** try “ sudo apt-get -f install " **LSTM:** <unk> |
| **3)** I’ve a 2.0 TB hard drive. It no longer boots from the hard drive, but I can see it from Gpart and it has a “ boot " flag. How do I recover it? $\rightarrow$ the boot flag doesn’t mean anything in Linux why do you not have a backup? you can mount the partition in liveCD and access the data easily ( assuming the partition is healthy ) $\rightarrow$ I have one hard drive for the moment ( well, I don’t count the 250GB drive ) besides, I don’t have two SATA cables I will make a backup as soon as I get another nice sized hard drive and a SATA cable And how do I find out if the partition is healthy? $\rightarrow$ boot to ubuntu livecd and try and mount the partition | **HRED:** How do I do that? **LSTM:** on the live cd? |
| **4)** bottom line is usually if you got 64bit why not use it? ( unless there is something that doesn’t work on 64bit, which is amitedly not much ) $\rightarrow$ well, now Clip2net is not working... just seems Ubuntu runs more seamlessly on 32-bit and after reading this article, Im wondering if 64 is a bit hyped? <url> | **HRED:** I have no experience with 64bit, so I don’t know what you mean. **LSTM:** <unk>, <unk>, <unk>, <unk>, <unk> |
| **5)** How would I enable vim syntax highlighting on a default install of ubuntu 12.04? $\rightarrow$ have you insalled vim instead of the default vim-tiny? $\rightarrow$ I installed vim-runtime but still no highlighting | **HRED:** yes **LSTM:** yes |
| **6)** Its comming from my computer 15 people using same router and they have no issues, also it happened recently, before it was ok $\rightarrow$ yes but your router may have optional settings for dns which if not set to the same dns will override your network interface... it ’s an option to consider | **HRED:** how do I do that? **LSTM:** that is what I was looking for, thanks |
| **7)** it looks as if the network ’s DHCP server is dead : “ < warn > ( eth0) : DHCPv4 request timed out " $\rightarrow$ but it isn’t. windows works | **HRED:** that ’s what I thought, but I don’t know how to fix it **LSTM:**? |
| **8)** I’ve been having trouble with a new 3TB harddrive and a fresh Ubuntu install. I install ( as GPT ) and the install looks ok. But then get GRUB “ error : out of disk " after reboot. I have an October 2011 PC with updated BIOS and I _do_ see the 3TB. google searches haven’t been to helpful so far. has anyone seen GRUB trouble when installing on 3TB harddrives? $\rightarrow$ have you tried installing grub again and updating it? sudo grub-install sudo update-grub? $\rightarrow$ I guess I could try that. So I would have to install GRUB from an older Ubuntu install? | **HRED:** yes, you can install grub to the MBR of your hard drive **LSTM:** grub is installed on the mbr of the hard drive |

TABLE 3.7: Ubuntu model examples. The $\rightarrow$ token indicates a change of turn, and <unk> is the unknown token.

error'), refers to the user having already tried this exact command. It is difficult for these models to integrate information from multiple utterances (Li et al., 2016b), and to understand the concept of coreference.

Finally, we also observed that the LSTM language model in general produced many poor responses, consisting either of a single punctuation mark or a series of unknown tokens. This is seen in examples 2), 4), and 7). On the contrary, the HRED model rarely produced such responses; thus, there is some indication that by improving the model architecture, we will produce models that generate longer and more interesting responses.

## 3.6 Discussion

In this chapter we introduce the Ubuntu Dialogue Corpus v2, a large dataset for research in unstructured multi-turn dialogue systems. We describe the construction of the dataset and its properties. The availability of a dataset of this size opens up several interesting possibilities for research into dialogue systems based on rich neural-network architectures. We present results demonstrating use of this dataset to train end-to-end RNN-based models, and critically evaluate the errors they make. We find that, while these models hold promise for building non-task-oriented dialogue systems, they still make many obvious errors, and there is significant room for improvement.

### 3.6.1 Conversation disentanglement

Our approach to conversation disentanglement consists of a small set of rules. More sophisticated techniques have been proposed, such as training a maximum-entropy classifier to cluster utterances into separate dialogues (Elsner and Charniak, 2008). However, since we are not trying to replicate the *exact* conversation between two users, but only to retrieve *plausible* natural dialogues, the heuristic method presented in this paper may be sufficient. This seems supported through qualitative examination of the data, but could be investigated with a more formal evaluation.

### 3.6.2 Drawbacks of end-to-end dialogue systems

In this chapter, we place significant focus on end-to-end dialogue systems, and take time to analyze the errors made by retrieval models. However, encoder-decoder methods have become the primary method for end-to-end dialogue generation with neural networks (e.g. (Serban et al., 2016, 2017; Li et al., 2016a,b)), and they too have several drawbacks. Most notable is the problem of *generic responses* (Serban et al., 2016; Li et al., 2015). Essentially, when training a generative model for dialogue, the model tends to frequently produces responses such as 'I'm don't know' and 'I'm so sorry'. This is partly a byproduct of the mode-seeking nature of the maximum likelihood training objective — these utterances are likely given any context, and thus a model will obtain a high likelihood for producing them. However, a dialogue system producing solely such responses is largely useless. While some effort has been made to overcome this problem (Li et al., 2015), it remains largely unsolved.

Similarly, generative dialogue models often have trouble producing grammatically correct responses; they often repeat a word multiple times in a sentence, or misuse pronouns or adjectives (Li et al., 2016b). They can also lack topic consistency, jumping from one subject to the next with no discernible motivation. In short, the space of grammatical and sensible English sentences is small compared to the number of possible sentences using English words, and discovering this space is difficult.

### 3.6.3 Dialogue evaluation

**Non-task-oriented evaluation in Ubuntu** It may seem unconventional that, given the technical nature of the Ubuntu Dialogue Corpus and the fact that it involves interactions where the end goal is solving a user's problem, we are treating our models as *non-task-oriented*, meaning that we do not incorporate a supervised task completion or user satisfaction signal during training or evaluation.

The reasons for this are purely practical; in general, training large, end-to-end goal-driven models is very difficult as it requires the collection of a large amount of task completion data. Annotating data in this way on a large scale is extremely expensive, and is usually only feasible for technical support channels at large corporations, which are rarely released publicly. Indeed, the Ubuntu Dialogue Corpus has no such labelled task completion data, and thus cannot be analyzed in the task-oriented setting for the time being. Obtaining such signals automatically remains an open problem. On the other hand, training non-task-oriented dialogue systems such as chatbots only requires conversational data, which can be obtained and shared publicly on a large scale. We believe that significant progress in dialogue systems can be made in this manner, as there remains many unsolved problems as illustrated in Section 3.4.5.

However, in the non-task-oriented setting we consider here, evaluation is more difficult. This is particularly true for end-to-end systems, as there is no way to measure the accuracy of the state tracking module using tasks such as slot filling, since they are not modular systems. Indeed, there are several reasons for wanting to move away from the slot filling metrics that have become common for modular systems. In slot filling, the set of candidate outputs (*states*) is identified *a priori* through knowledge engineering, and is typically rather small in comparison to the set of responses considered by NUC. Further, it has been speculated that state-of-the-art state-tracking models (Henderson et al., 2014b; Williams, 2014) are achieving close to human-level performance. Thus it is desirable to move beyond this domain into more difficult problems. To do this, it is crucial to investigate ways to evaluate models in the non-task-oriented setting that do not require supervised test data for the internal modules of a system.

**Existing evaluation metrics** Researchers have previously proposed measuring word perplexity and word classification error rate, as these are widely applied in the language modeling and automatic speech recognition community (Serban et al., 2016; Vinyals and Le, 2015; Pietquin and Hastie, 2013). However, these metrics cannot be computed for retrieval models. Researchers have also proposed to use word overlap metrics from machine translation (Galley et al., 2015; Sordoni et al., 2015b). However, such metrics based on word overlaps suffer from severe sparsity issues, since it is unlikely that any sequence of words will be identical in both the generated and reference responses. We show in Section 4.1 that these metrics correlate poorly with human judgements when only a single ground-truth response is available. Furthermore, it has been argued that such metrics mainly focus on pronouns and punctuation marks when applied to non-task-oriented dialogue datasets (Serban et al., 2016). The word embedding metrics used here do not have a strong correlation with human judgement (as we show later in Section 4.1). However, they do have an additional interpretation of measuring the *semantic similarity* between the generated and reference responses (as argued in Section 3.4.3), which is why we favour them over word-overlap scores such as BLEU here. However, we reiterate that none of these metrics measures the *coherence* of the generated responses, and this remains an important direction for future work. We perform a more detailed analysis of existing dialogue evaluation metrics in Section 4.1.

**Next utterance classification** Another option for evaluating dialogue systems trained in an end-to-end manner is using an alternative task such as next utterance classification, as we do for retrieval models in this section. Here we provide some more justification for why NUC is a reasonable task; we conduct a human study with even more analysis in Section 4.2.

While this does not directly compare the generated response of the system to the ground-truth response, there are several reasons for preferring the recall metric:

1. It is a more difficult task than slot filling, and thus will require further development of more sophisticated dialogue systems in order to solve the task.
2. It does not suffer from the same problems as the word overlap metrics, as it does not have to directly compare the quality of a generated response to the ground-truth response, an inherently noisy process. Instead, it measures the model's capacity to pick out the correct response from a list of responses.
3. Performance using the recall metrics is easily interpretable, and can easily be compared to human performance. Indeed, we conduct this analysis in Section 4.2, and show that human performance on this task is above the performance for the Dual Encoder model on the Ubuntu Dialogue Corpus, as well as on movie and Twitter corpora. Thus, there is room for improvement for models on this task.

4. The task is *consistent* with the end goal of building dialogue systems that can converse naturally with humans. More precisely, models that are able to generate good responses should also be able to pick good responses from a list of candidates, as in NUC.

5. It is easy to alter the task difficulty in a controlled manner. We demonstrated this by moving from 1 to 9 false responses, and by varying the Recall@k parameter. In the future, instead of choosing false responses randomly, one could consider selecting false responses that are similar to the actual response (e.g. as measured by TF-IDF cosine similarity). A dialogue model that performs well on this more difficult task should also manage to capture a more fine-grained semantic meaning of sentences, as compared to a model that naively picks replies with the most words in common with the context such as TF-IDF. In fact, when the set of candidate responses for the model to choose from is close to the size of the dataset (e.g. all utterances ever recorded), then NUC becomes close to the response generation case.

Given the above points, we believe that evaluating models with the NUC task is very useful for the time being. However, we believe that caution should be used when comparing retrieval models to generative models using NUC, as the retrieval models are directly trained on the task of NUC, rendering it an unfair comparison.

### 3.6.4 Future Research Directions for End-to-End Systems

Given the analysis performed in Section 3.4.5, we postulate several interesting directions for future research on end-to-end dialogue systems, particularly on the Ubuntu Dialogue Corpus.

An important challenge in dialogue systems is the ability to understand the turn-taking structure of dialogue. This is a significant source of errors for the Dual Encoder model. Some progress in this direction has been made for end-to-end dialogue systems (Luan et al., 2016; Li et al., 2016a), using approaches derived from topic modelling or by explicitly modelling each user with a continuous-valued vector. However, this is still an open problem. This is related to the issue of end-to-end dialogue *personalization*, which involves building end-to-end dialogue systems that are tailored to a particular user and that evolve over time as the user's preferences change.

The largest source of errors from the analysis in Section 3.4.5 was in the failure to understand the semantic similarity between the context and response. This falls under the more general problem of *natural language understanding*, which arises in many NLP tasks. This may require adjustments in the architecture of end-to-end models to render them more suited to processing language. It is possible that insights can be derived from architectures developed on more targeted language understanding tasks, such as the CNN/ Daily Mail reading comprehension dataset (Hermann et al., 2015), where attention-based models have achieved strong performance. The recent success of very large neural models such as BERT (Devlin et al., 2018) and GPT-2 (Radford et al., 2019) using Transformer-based architectures (Vaswani et al., 2017) on multi-task NLP benchmarks such as Glue (Wang et al., 2018) indicate that simply scaling neural models could lead to further gains in natural language understanding.

In order to be able to correctly answer questions regarding Ubuntu and solve the user's problem, dialogue models will inevitably require some knowledge of the Ubuntu domain. This will most likely be achieved by using some source of *external knowledge*, in addition to the knowledge that is present in the dialogue of the Ubuntu Dialogue Corpus. Thus, an important direction for research is the investigation of methods that incorporate external knowledge sources with end-to-end dialogue systems. This applies more generally to any end-to-end system that is developed for the goal-oriented setting, and may require imposing additional structure on the output space of the model. There is promising work in this direction from Wen et al. (2016), however methods must be derived that are effective in a larger and more general setting than restaurant recommendation. More recent work (Ghazvininejad et al., 2018; Zhu et al., 2017) successfully augments neural sequence-to-sequence models with external knowledge in domains such as Foursquare and Twitter.

A common problem that has been observed when training generative end-to-end models that maximize the log-likelihood of the conversational response is that these models tend to produce generic responses at test time. This has been observed empirically (Vinyals and Le, 2015; Serban et al., 2016), and was also seen in some of the LSTM and HRED examples presented in Section 3.5.4. This has been investigated in (Li et al., 2015), where the authors construct an objective function based on mutual information that promotes diversity, however they achieve only modest improvements. This is a large impediment for building end-to-end systems that can have interesting and engaging interactions with users.

Finally, an important direction for future research is building large-scale datasets that allow the training of goal-oriented systems. The Ubuntu domain is particularly suited for training goal-oriented systems, however this is not yet possible on the Ubuntu Dialogue Corpus as there are no supervised task completion signals, as mentioned in Section 3.6.3. Building models that can approximate such signals is challenging, yet it may be necessary in order to develop systems that can solve users' problems in a meaningful way in a domain as complex as Ubuntu.

### 3.6.5 Reflecting on the Ubuntu Dialogue Corpus

Five years after its original publication (Lowe et al., 2015), the Ubuntu Dialogue Corpus has been used fairly extensively for research on dialogue systems (see e.g. (Wu et al., 2016; Dodge et al., 2015)). However, the dataset has been used less recently in favour of datasets such as PersonaChat (Zhang et al., 2018) and MultiWoz (Budzianowski et al., 2018). There are several reasons this could be the case. For one, while dialogues in the PersonaChat and MultiWoz datasets use external knowledge (such as the interlocutor's personality, or the desired restaurant to book), this information is made available at the beginning of each dialogue. Conversely, Ubuntu dialogues require external knowledge of Ubuntu (such as how to install a driver) which is not present in easily readable form. This makes the task of generating a response significantly harder in the Ubuntu domain. Due to the technical nature of conversations, it can also be difficult for humans without Ubuntu expertise to evaluate the quality of responses.

On top of this, as alluded to in Section 3.6.1, the disentanglement procedure used to generate the Ubuntu Dialogue Corpus is quite noisy, and leads to many conversations that are either incomplete, or contain extraneous information, or both. A recent study by Kummerfeld et al. (2018) found that 58% of dialogues in the Ubuntu Dialogue Corpus were missing a relevant utterance from the original chat logs. While many of these omissions occurred at the beginning or end of the dialogue (and thus, the resulting dialogue is still coherent, albeit lacking in context), this still indicates that the Ubuntu corpus isn't exactly reflective of real human-human conversations.

# Chapter 4

# Analyzing evaluation methods for dialogue systems

An important aspect of dialogue response generation systems, which are trained to produce a reasonable utterance given a conversational context, is how to evaluate the quality of the generated response. Typically, evaluation is done using human-generated supervised signals, such as a task completion test or a user satisfaction score (Walker et al., 1997; Möller et al., 2006), which are relevant when the dialogue is task-focused. As discussed in Section 2.4.1, we call models optimized for such objectives *task-oriented dialogue systems*, while those that do not are *non-task-oriented dialogue systems*.

This chapter consists of three separate works focused on assessing and improving automatic evaluation metrics for non-task-oriented systems, such as chatbots. These models are receiving increased attention, particularly using end-to-end training with neural networks (Serban et al., 2016; Sordoni et al., 2015b; Vinyals and Le, 2015). Automatic metrics are desirable as they avoid the need to collect supervised labels on a large scale, which can be prohibitively expensive. However, finding useful metrics to automatically evaluate the quality of non-task-oriented models is an open question. In Section 4.1, we examine various automatic metrics for non-task-oriented dialogue systems, including commonly-used ones such as BLEU, and find that they correlate poorly with human judgements. In Section 4.2, we conduct a human study of a retrieval-based metric we call next utterance classification (NUC), and find that human performance is significantly better than machine performance (as of 2016) on two different datasets, suggesting that NUC may be a useful metric to target for improving dialogue systems. Finally, in Section 4.3 we propose a method for *learning* a dialogue evaluation model based on a collected dataset of human judgements, and find that the model can achieve significant correlation with human judgements on the test set, although some evidence indicates it transfers poorly to other datasets.

## 4.1 A critique of word overlap-based evaluation metrics

### 4.1.1 Motivation

As previously mentioned, we want to usefully and automatically evaluate various dialogue systems without having to collect a dataset of human judgements for each model. Faced with similar challenges, other natural language tasks have successfully developed automatic evaluation metrics. For example, BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) are now standard for evaluating machine translation models, and ROUGE (Lin, 2004) is often used for automatic summarization. These metrics have recently been adopted by dialogue researchers (Ritter et al., 2011; Sordoni et al., 2015b; Li et al., 2015; Galley et al., 2015; Wen et al., 2015; Li et al., 2016a). However these metrics assume that valid responses have significant word overlap with the ground truth responses. This is a strong assumption for dialogue systems, where there is significant diversity in the space of valid responses to a given context. This is illustrated in Table 4.1, where two reasonable responses are proposed to the context, but these responses do not share any words in common and do not have the same semantic meaning.

In this chapter, we investigate the correlation between the scores from several automatic evaluation metrics and human judgements of dialogue response quality, for a variety of response generation models. We consider both statistical *word-overlap similarity metrics* such as BLEU, METEOR, and ROUGE, and *word embedding metrics* derived from word embedding models such as Word2Vec (Mikolov et al., 2013b). We find that all metrics show either weak or no correlation with human judgements, despite the fact that word overlap

| Context of Conversation |
| --- |
| Speaker A: Hey John, what do you want to do tonight? |
| Speaker B: Why don't we go see a movie? |
| **Ground-Truth Response** |
| Nah, I hate that stuff, let's do something active. |
| **Model Response** |
| Oh sure! Heard the film about Turing is out! |

Table 4.1: Example showing the intrinsic diversity of valid responses in a dialogue. The (reasonable) model response would receive a BLEU score of 0.

metrics have been used extensively in the literature for evaluating dialogue response models (see above, and (Nio et al., 2014b)). In particular, we show that these metrics have only a small positive correlation on the chitchat oriented Twitter dataset, and no correlation at all on the technical Ubuntu Dialogue Corpus. For the word embedding metrics, we show that this is true even though all metrics are able to significantly distinguish between baseline and state-of-the-art models across multiple datasets. We further highlight the shortcomings of these metrics using: a) a statistical analysis of our survey's results; b) a qualitative analysis of examples from our data; and c) an exploration of the sensitivity of the metrics.

Our results indicate that a shift must be made in the research community away from these metrics, and highlight the need for a new metric that correlates more strongly with human judgement.

### 4.1.2 Evaluation Metrics

Given a dialogue context and a proposed response, our goal is to automatically evaluate how appropriate the proposed response is to the conversation. We focus on metrics that compare it to the ground truth response of the conversation. In particular, we investigate two approaches: word based similarity metrics and word-embedding based similarity metrics.

#### 4.1.2.1 Word Overlap-based Metrics

We first consider metrics that evaluate the amount of *word-overlap* between the proposed response and the ground-truth response. We examine the BLEU and METEOR scores that have been used for machine translation, and the ROUGE score that has been used for automatic summarization. While these metrics have been shown to correlate with human judgements in their target domains (Papineni et al., 2002; Lin, 2004), they have not been thoroughly investigated for dialogue systems.[1]

We denote the ground truth response as $r$ (thus we assume that there is a single candidate ground truth response), and the proposed response as $\hat{r}$. The $j$'th token in the ground truth response $r$ is denoted by $w_j$, with $\hat{w}_j$ denoting the $j$'th token in the proposed response $\hat{r}$.

**BLEU** BLEU (Papineni et al., 2002) analyzes the co-occurrences of n-grams in the ground truth and the proposed responses. It first computes an n-gram precision for the whole dataset (we assume that there is a single candidate ground truth response per context):

$$P_n(r, \hat{r}) = \frac{\sum_k \min(h(k,r), h(k, \hat{r}_i))}{\sum_k h(k, r_i)}$$

where $k$ indexes all possible n-grams of length $n$ and $h(k,r)$ is the number of n-grams $k$ in $r$.[2] To avoid the drawbacks of using a precision score, namely that it favours shorter (candidate) sentences, the authors introduce a brevity penalty. BLEU-N, where $N$ is the maximum length of n-grams considered, is defined as:

$$\text{BLEU-N} := b(r, \hat{r}) \exp(\sum_{n=1}^{N} \beta_n \log P_n(r, \hat{r}))$$

$\beta_n$ is a weighting that is usually uniform, and $b(\cdot)$ is the brevity penalty. The most commonly used version of BLEU uses $N = 4$. Modern versions of BLEU also use sentence-level smoothing, as the geometric mean often results in scores of 0 if there is no 4-gram overlap (Chen and Cherry, 2014). Note that BLEU is usually calculated at the corpus-level, and was originally designed for use with multiple reference sentences.

[1]To the best of our knowledge, only BLEU has been evaluated in the dialogue system setting quantitatively by Galley et al. (2015) on the Twitter domain. However, they carried out their experiments in a very different setting with multiple ground truth responses, which are rarely available in practice, and without providing any qualitative analysis of their results.

[2]Note that the min in this equation is calculating the number of co-occurrences of n-gram $k$ between the ground truth response $r$ and the proposed response $\hat{r}$, as it computes the fewest appearances of $k$ in either response.

**METEOR** The METEOR metric (Banerjee and Lavie, 2005) was introduced to address several weaknesses in BLEU. It creates an explicit alignment between the candidate and target responses. The alignment is based on exact token matching, followed by WordNet synonyms, stemmed tokens, and then paraphrases. Given a set of alignments $m$, the METEOR score is the harmonic mean of precision $P_m$ and recall $R_m$ between the candidate and target sentence.

$$Pen = \gamma(\frac{ch}{m})^{\theta} \tag{4.1}$$

$$F_{mean} = \frac{P_m R_m}{\alpha P_m + (1-\alpha) R_m} \tag{4.2}$$

$$P_m = \frac{|m|}{\sum_k h_k(c_i)} \tag{4.3}$$

$$R_m = \frac{|m|}{\sum_k h_k(s_{ij})} \tag{4.4}$$

$$METEOR = (1 - Pen) F_{mean} \tag{4.5}$$

The penalty term $Pen$ is based on the 'chunkiness' of the resolved matches.

**ROUGE** ROUGE (Lin, 2004) is a set of evaluation metrics used for automatic summarization. We consider ROUGE-L, which is a F-measure based on the Longest Common Subsequence (LCS) between a candidate and target sentence. The LCS is a set of words which occur in two sentences in the same order; however, unlike n-grams the words do not have to be contiguous, i.e. there can be other words in between the words of the LCS.

#### 4.1.2.2 Embedding-based Metrics

An alternative to using word-overlap based metrics is to consider the meaning of each word as defined by a *word embedding*, which assigns a vector to each word. We use the greedy matching, embedding average, and vector extrema metrics that are defined in Section 3.4.3.

### 4.1.3 Dialogue Response Generation Models

In order to determine the correlation between automatic metrics and human judgements of response quality, we obtain responses from a diverse range of response generation models in the recent literature, including both retrieval and generative models. We've already described the distinction between generative and retrieval models in Section 2.4.2, and detailed the models that we use here in Chapter 3. Thus, in this section we simply give a high-level overview of the models we use.

| | **Ubuntu Dialogue Corpus** | | | **Twitter Corpus** | | |
|---|---|---|---|---|---|---|
| | Embedding Averaging | Greedy Matching | Vector Extrema | Embedding Averaging | Greedy Matching | Vector Extrema |
| R-TFIDF | 0.536 ± 0.003 | 0.370 ± 0.002 | 0.342 ± 0.002 | 0.483 ± 0.002 | 0.356 ± 0.001 | 0.340 ± 0.001 |
| C-TFIDF | 0.571 ± 0.003 | 0.373 ± 0.002 | 0.353 ± 0.002 | 0.531 ± 0.002 | 0.362 ± 0.001 | 0.353 ± 0.001 |
| DE | **0.650 ± 0.003** | 0.413 ± 0.002 | 0.376 ± 0.001 | **0.597 ± 0.002** | 0.384 ± 0.001 | 0.365 ± 0.001 |
| LSTM | 0.130 ± 0.003 | 0.097 ± 0.003 | 0.089 ± 0.002 | 0.593 ± 0.002 | **0.439 ± 0.002** | **0.420 ± 0.002** |
| HRED | 0.580 ± 0.003 | **0.418 ± 0.003** | **0.384 ± 0.002** | **0.599 ± 0.002** | **0.439 ± 0.002** | **0.422 ± 0.002** |

TABLE 4.2: Models evaluated using the vector-based evaluation metrics, with 95% confidence intervals.

We evaluate two different retrieval models here: TF-IDF, a simple statistic that captures how important a given word is to some document (described in Section 3.4.1), and the Dual Encoder, which uses two RNNs to give a score to each candidate response (described in Section 3.4.2). Specifically, we use two different TF-IDF models: R-TFIDF which calculates the TF-IDF score between the input context and the responses in the dataset, and C-TFIDF, which computes the most similar context in the dataset to the input context, and returns the corresponding response.

Retrieval models for dialogue systems are typically evaluated based on whether they can retrieve the correct response from a corpus of pre-defined responses, which includes the ground truth response to the conversation (Schatzmann et al., 2005). Such systems can be evaluated using recall or precision metrics. However, when deployed in a real setting these models will not have access to the correct response given an unseen conversation. Thus, in the results presented below we *remove* one occurrence of the ground-truth response from the corpus and ask the model to retrieve the most appropriate response from the remaining utterances. Note that this does not mean the correct response will not appear in the corpus at all; in particular, if there exists another context in the dataset with an identical ground-truth response, this will be available for selection by the model. We then evaluate each model by comparing the retrieved response to the ground truth response of the conversation. This closely imitates real-life deployment of these models, as it tests the ability of the model to generalize to unseen contexts.

We also consider two kinds of generative models: a simple RNN language model with LSTM units (described in Section 3.5.1) and the hierarchical recurrent encoder-decoder (HRED) model, which uses two separate LSTMs to model the context at the word and response level, and a third LSTM as the decoder (described in Section 3.5.2).

| | **Twitter** | | | | **Ubuntu** | | | |
|---|---|---|---|---|---|---|---|---|
| Metric | **Spearman** | p-value | **Pearson** | p-value | **Spearman** | p-value | **Pearson** | p-value |
| Greedy | 0.2119 | 0.034 | 0.1994 | 0.047 | 0.05276 | 0.6 | 0.02049 | 0.84 |
| Average | 0.2259 | 0.024 | 0.1971 | 0.049 | -0.1387 | 0.17 | -0.1631 | 0.10 |
| Extrema | 0.2103 | 0.036 | 0.1842 | 0.067 | 0.09243 | 0.36 | -0.002903 | 0.98 |
| METEOR | 0.1887 | 0.06 | 0.1927 | 0.055 | 0.06314 | 0.53 | 0.1419 | 0.16 |
| BLEU-1 | 0.1665 | 0.098 | 0.1288 | 0.2 | -0.02552 | 0.8 | 0.01929 | 0.85 |
| BLEU-2 | 0.3576 | $< 0.01$ | 0.3874 | $< 0.01$ | 0.03819 | 0.71 | 0.0586 | 0.56 |
| BLEU-3 | 0.3423 | $< 0.01$ | 0.1443 | 0.15 | 0.0878 | 0.38 | 0.1116 | 0.27 |
| BLEU-4 | 0.3417 | $< 0.01$ | 0.1392 | 0.17 | 0.1218 | 0.23 | 0.1132 | 0.26 |
| ROUGE | 0.1235 | 0.22 | 0.09714 | 0.34 | 0.05405 | 0.5933 | 0.06401 | 0.53 |
| Human | 0.9476 | $< 0.01$ | 1.0 | 0.0 | 0.9550 | $< 0.01$ | 1.0 | 0.0 |

Table 4.3: Correlation between each metric and human judgements for each response. Correlations shown in the human row result from randomly dividing human judges into two groups.

| | **Spearman** | p-value | **Pearson** | p-value |
|---|---|---|---|---|
| BLEU-1 | 0.1580 | 0.12 | 0.2074 | 0.038 |
| BLEU-2 | 0.2030 | 0.043 | 0.1300 | 0.20 |
| BLEU-3 | 0.2315 | 0.020 | 0.2435 | 0.015 |
| BLEU-4 | 0.2253 | 0.024 | 0.2640 | $< 0.01$ |

Table 4.4: Correlation between BLEU metric and human judgements after removing stopwords and punctuation for the Twitter dataset.

#### 4.1.3.1 Conclusions from an Incomplete Analysis

When evaluation metrics are not explicitly correlated to human judgement, it is possible to draw misleading conclusions by examining how the metrics rate different models. To illustrate this point, we compare the performance of selected models according to the embedding metrics on two different domains: the Ubuntu Dialogue Corpus (Lowe et al., 2015), which contains technical

| | Mean score | | |
|---|---|---|---|
| | $\Delta w <= 6$ (n=47) | $\Delta w >= 6$ (n=53) | p-value |
| BLEU-1 | 0.1724 | 0.1009 | $< 0.01$ |
| BLEU-2 | 0.0744 | 0.04176 | $< 0.01$ |
| Average | 0.6587 | 0.6246 | 0.25 |
| METEOR | 0.2386 | 0.2073 | $< 0.01$ |
| Human | 2.66 | 2.57 | 0.73 |

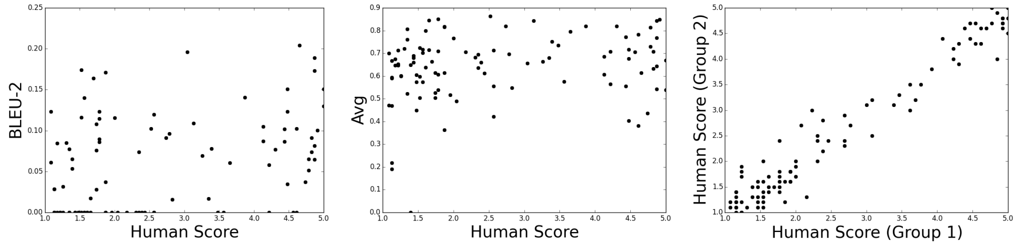
TABLE 4.5: Effect of differences in response length for the Twitter dataset, $\Delta w$ = absolute difference in #words between a ground truth response and proposed response

vocabulary and where conversations are often oriented towards solving a particular problem, and a non-technical Twitter corpus collected following the procedure of Ritter et al. (2010). We consider these two datasets since they cover contrasting dialogue domains, i.e. technical help vs casual chit-chat, and because they are amongst the largest publicly available corpora, making them good candidates for building data-driven dialogue systems.

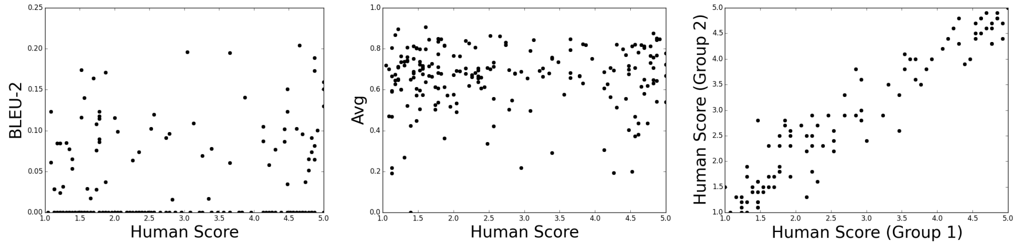
Results on the proposed embedding metrics are shown in Table 4.2. For the retrieval models, we observe that the DE model significantly outperforms both TFIDF baselines on all metrics across both datasets. Further, the HRED model significantly outperforms the basic LSTM generative model in both domains, and appears to be of similar strength as the DE model. Based on these results, one might be tempted to conclude that there is some information being captured by these metrics, that significantly differentiates models of different quality. However, as we show in the next subsection, the embedding-based metrics correlate only weakly with human judgements on the Twitter corpus, and not at all on the Ubuntu Dialogue Corpus. This demonstrates that metrics that have not been specifically correlated with human judgements on a new task should not be used to evaluate that task.

### 4.1.4 Human Correlation Analysis

**Data Collection** We conducted a human survey to determine the correlation between human judgements on the quality of responses, and the score assigned by each metric. We aimed to follow the procedure for the evaluation of BLEU (Papineni et al., 2002). 25 volunteers from the Computer Science department at the author's institution were given a context and one proposed




(A) Twitter




(B) Ubuntu

FIGURE 4.1: Scatter plots showing the correlation between metrics and human judgements on the Twitter corpus (a) and Ubuntu Dialogue Corpus (b). The plots represent BLEU-2 (left), embedding average (center), and correlation between two randomly selected halves of human respondents (right).

response, and were asked to judge the response quality on a scale of 1 to 5.[3]; a 1 indicates that the response is not appropriate or sensible given the context, and a 5 indicates that the response is very reasonable. Out of the 25 respondents, 23 had Cohen's kappa scores $\kappa > 0.2$ w.r.t. the other respondents, which is a standard measure for inter-rater agreement (Cohen, 1968). The 2 respondents with $\kappa < 0.2$, indicating slight agreement, were excluded from the analysis below. The median $\kappa$ score was approximately 0.55, roughly indicating moderate to strong annotator agreement.

Each volunteer was given 100 questions per dataset. These questions correspond to 20 unique contexts, with 5 different responses: one utterance randomly drawn from elsewhere in the test set, the response selected from each of the TF-IDF, DE, and HRED models, and a response written by a human annotator. These were chosen as they cover the range of qualities almost uniformly (see Figure 4.1).

[3]Studies asking humans to evaluate text often rate different aspects separately, such as 'adequacy', 'fluency' and 'informativeness' of the text (Hovy, 1999; Papineni et al., 2002) Our evaluation focuses on adequacy. We did not consider fluency because 4 out of the 5 proposed responses to each context were generated by a human. We did not consider informativeness because in the domains considered, it is not necessarily important (in Twitter), or else it seems to correlate highly with adequacy (in Ubuntu).

**Context of Conversation**
A: dearest! question. how many thousands of people can panaad occupy?
B: @user panaad has <number> k seat capacity while rizal has <number> k thats why they choose rizal i think .

**Ground Truth Response**
A: now i know about the siting capacity . thanks for the info @user great evening.

**Proposed Response**
A: @user makes sense. thanks!

**Context of Conversation**
A: never felt more sad than i am now
B: @user aww why ?
A: @user @user its a long story ! sure you wanna know it ? bahaha and thanks for caring btw <heart>

**Ground Truth Response**
A: @user i don ’t mind to hear it i ’ve got all day and youre welcome <number>

**Proposed Response**
A: @user i know , i ’m just so happy for you ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !

FIGURE 4.2: Examples where the metrics rated the response poorly and humans rated it highly (left), and the converse (right). Both responses are given near-zero score by BLEU-N for N> 1. While no metric will perform perfectly on all examples, we present these examples to provide intuition on how example-level errors become aggregated into poor correlation to human judgements at the corpus-level.

**Survey Results** We present correlation results between the human judgements and each metric in Table 4.3. We compute the Pearson correlation, which estimates linear correlation, and Spearman correlation, which estimates any monotonic correlation.

The first observation is that in both domains the BLEU-4 score, which has previously been used to evaluate unsupervised dialogue systems, shows very weak if any correlation with human judgement. In fact we found that the BLEU-3 and BLEU-4 scores were near-zero for a majority of response pairs; for BLEU-4, only four examples had a score $> 10^{-9}$. Despite this, they still correlate with human judgements on the Twitter Corpus at a rate similar to BLEU-2. This is because of the smoothing constant, which gives a tiny weight to unigrams and bigrams despite the absence of higher-order n-grams. BLEU-3 and BLEU-4 behave as a scaled, noisy version of BLEU-2; thus, if one is to evaluate dialogue responses with BLEU, we recommend the choice of $N = 2$ over $N = 3$ or 4. Note that using a test corpus larger than the size reported in this chapter may lead to stronger correlations for BLEU-3 and BLEU-4, due to a higher number of non-zero scores.

It is interesting to note that, while some of the embedding metrics and BLEU show small positive correlation in the non-technical Twitter domain, there is no metric that significantly correlates with humans on the Ubuntu Dialogue Corpus. This is likely because the correct Ubuntu responses contain specific technical words that are less likely to be produced by our models. Further, it is possible that responses in the Ubuntu Dialogue Corpus have intrinsically higher variability (or entropy) than Twitter when conditioned on the context, making the evaluation problem significantly more difficult.

Figure 4.1 illustrates the relationship between metrics and human judgements. We include only the best performing metric using word-overlaps, i.e. the BLEU-2 score (left), and the best performing metric using word embeddings, i.e. the vector average (center). These plots show how weak the correlation is: in both cases, they appear to be random noise. It seems as though the BLEU score obtains a positive correlation because of the large number of responses that are given a score of 0 (bottom left corner of the first plot). This is in stark contrast to the inter-rater agreement, which is plotted between two randomly sampled halves of the raters (right-most plots). We also calculated the BLEU scores after removing stopwords and punctuation from the responses. As shown in Table 4.4, this weakens the correlation with human judgements for BLEU-2 compared to the values in Table 4.3, and suggests that BLEU is sensitive to factors that do not change the semantics of the response.

Finally, we examined the effect of response length on the metrics, by considering changes in scores when the ground truth and proposed response had a large difference in word counts. Table 4.4 shows that BLEU and METEOR are particularly sensitive to this aspect, compared to the Embedding Average metric and human judgement.

**Qualitative Analysis** In order to determine specifically why the metrics fail, we examine qualitative samples where there is a disagreement between the metrics and human rating. Although these only show inconsistencies at the example-level, they provide some intuition as to why the metrics don't correlate with human judgements at the corpus-level. We present in Figure 4.2 two examples where all of the embedding-based metrics and BLEU-1 score the proposed response significantly differently than the humans.

The left of Figure 4.2 shows an example where the embedding-based metrics score the proposed response lowly, while humans rate it highly. It is clear from the context that the proposed response is reasonable – indeed both responses intend to express gratitude. However, the proposed response has a different wording than the ground truth response, and therefore the metrics are unable to separate the salient words from the rest. This suggests that the embedding-based metrics would benefit from a weighting of word saliency.

The right of the figure shows the reverse scenario: the embedding-based metrics score the proposed response highly, while humans do not. This is most likely due to the frequently occurring 'i' token, and the fact that 'happy' and 'welcome' may be close together in the embedding space. However, from a human perspective there is a significant semantic difference between the responses as they pertain to the context. Metrics that take into account the context may be required in order to differentiate these responses. Note that in both responses in Figure 4.2, there are no overlapping n-grams greater than unigrams between the ground truth and proposed responses; thus, all of BLEU-2,3,4 would assign a score near 0 to the response.

### 4.1.5 Discussion

We have shown that many metrics commonly used in the literature for evaluating unsupervised dialogue systems do not correlate strongly with human judgement. Here we elaborate on important issues arising from our analysis.

**Constrained tasks** Our analysis focuses on relatively unconstrained domains. Other work, which separates the dialogue system into a dialogue planner and a natural language generation component for applications in constrained domains, may find stronger correlations with the BLEU metric. For example, Wen et al. (2015) propose a model to map from dialogue acts to natural language sentences and use BLEU to evaluate the quality of the generated sentences. Since the mapping from dialogue acts to natural language sentences has lower diversity and is more similar to the machine translation task, it seems likely that BLEU will correlate better with human judgements. However, an empirical investigation is still necessary to justify this.

**Incorporating multiple responses** Our correlation results assume that only one ground truth response is available given each context. Indeed, this is the common setting in most of the recent literature on training end-to-end conversation models. There has been some work on using a larger set of automatically retrieved plausible responses when evaluating with BLEU (Galley et al., 2015). However, there is no standard method for doing this in the literature. Future work should examine how retrieving additional responses affects the correlation with word-overlap metrics.

**Searching for suitable metrics** While we provide evidence against existing metrics, we do not yet provide good alternatives for unsupervised evaluation. Despite the poor performance of the word embedding-based metrics in this survey, we believe that metrics based on distributed sentence representations hold the most promise for the future. This is because word-overlap metrics will simply require too many ground-truth responses to find a significant match for a reasonable response, due to the high diversity of dialogue responses. As a simple example, the skip-thought vectors of Kiros et al. (2015) could be considered. Since the embedding-based metrics in this chapter only consist of basic averages of vectors obtained through distributional semantics, they are insufficiently complex for modeling sentence-level compositionality in dialogue. Instead, these metrics can be interpreted as calculating the *topicality* of a proposed response (i.e. how on-topic the proposed response is, compared to the ground-truth).

All of the metrics considered in this chapter directly compare a proposed response to the ground-truth, without considering the context of the conversation. However, metrics that take into account the context could also be considered. Such metrics could come in the form of an *evaluation model* that is learned from data. This model could be either a discriminative model that attempts to distinguish between model and human responses, or a model that uses data collected from the human survey in order to provide human-like scores to proposed responses. Finally, we must consider the hypothesis that learning such models from data is no easier than solving the problem of dialogue response generation.

If this hypothesis is true, we must concede and always use human evaluations together with metrics that only roughly approximate human judgements.

## 4.2 A human study of next utterance classification

### 4.2.1 Motivation

In this chapter, we consider tasks and evaluation measures for 'non-task-oriented' dialogue systems. While there has been significant work on building end-to-end response generation systems (Vinyals and Le, 2015; Shang et al., 2015), we've shown above in Section 4.1 that many of the automatic evaluation metrics used for such systems correlate poorly or not at all with human judgement of the generated responses.

Retrieval-based systems are of interest because they admit a natural evaluation metric, namely the recall and precision measures. First introduced for evaluating user simulations by Schatzmann et al. (2005), such a framework has gained recent prominence for the evaluation of end-to-end dialogue systems (Lowe et al., 2015; Kadlec et al., 2015; Dodge et al., 2015). These models are trained on the *task* of selecting the correct response from a candidate list, which we call next utterance classification (NUC, detailed in Section 4.2.3), and are evaluated using the *metric* of recall. NUC is useful for several reasons: 1) the performance (i.e. loss or error) is easy to compute automatically, 2) it is simple to adjust the difficulty of the task, 3) the task is interpretable and amenable to comparison with human performance, 4) it is an easier task compared to generative dialogue modeling, which is difficult for end-to-end systems (Sordoni et al., 2015b; Serban et al., 2016), and 5) models trained with NUC can be converted to dialogue systems by retrieving from the full corpus (Liu et al., 2016). In this case, NUC additionally allows for making hard constraints on the allowable outputs of the system (to prevent offensive responses), and guarantees that the responses are fluent (because they were generated by humans). Thus, NUC can be thought of both as an *intermediate task* that can be used to evaluate the ability of systems to understand natural language conversations, similar to the bAbI tasks for language understanding (Weston et al., 2015), and as a *useful framework* for building chatbots. With the huge size of current dialogue datasets that contain millions of utterances (Lowe et al., 2015; Banchs, 2012; Ritter et al., 2010) and the increasing amount of natural language data, it is conceivable that retrieval-based systems will be able to have engaging conversations with humans.

However, despite the current work with NUC, there has been no verification of whether machine and human performance differ on this task. This cannot be assumed; it is possible that no significant gap exists between the two, as is the case with many current automatic response generation metrics (Liu et al., 2016). Further, it is important to benchmark human performance on new tasks such as NUC to determine when research has outgrown their use. In this chapter, we consider to what extent NUC is achievable by humans, whether human performance varies according to expertise, and whether there is room for machine performance to improve (or has reached human performance already) and we should move to more complex conversational tasks. We performed a user study on three different datasets: the SubTle Corpus of movie dialogues (Banchs, 2012), the Twitter Corpus (Ritter et al., 2010), and the Ubuntu Dialogue Corpus (Lowe et al., 2015). Since conversations in the Ubuntu Dialogue Corpus are highly technical, we recruit 'expert' humans who are adept with the Ubuntu terminology, whom we compare with a state-of-the-art machine learning agent on all datasets. We find that there is indeed a significant separation between machine and expert human performance, suggesting that NUC is a useful intermediate task for measuring progress.

### 4.2.2 Related Work

Evaluation methods for supervised systems have been well studied. They include the PARADISE framework (Walker et al., 1997), and MeMo (Möller et al., 2006), which include a measure of task completion. A more extensive overview of these metrics can be found in (Jokinen and McTear, 2009). We focus in this chapter on unsupervised dialogue systems, for which proper evaluation is an open problem.

Recent evaluation metrics for unsupervised dialogue systems include BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005), which compare the similarity between response generated by the model, and the actual response of the participant, conditioned on some context of the conversation. Word perplexity, which computes a function of the probability of re-generating


FIGURE 4.3: An example NUC question from the SubTle Corpus (Banchs, 2012).

examples from the training corpus, is also used. However, such metrics have been shown to correlate very weakly with human judgement of the produced responses (see Section 4.1). They also suffer from several other drawbacks, including low scores, lack of interpretability, and inability to account for the vast space of acceptable outputs in natural conversation.

### 4.2.3 Technical Background on NUC

Our long-term goal is the development and deployment of artificial conversational agents. Recent deep neural architectures offer perhaps the most promising framework for tackling this problem. However training such architectures typically requires large amounts of conversation data from the target domain, and a way to automatically assess prediction errors. Next utterance classification (NUC, see Figure 4.3) is a *task*, which is straightforward to evaluate, designed for training and validation of dialogue systems. They are evaluated using the *metric* of Recall@k, which we define in this section.

In NUC, a model or user, when presented with the context of a conversation and a (usually small) pre-defined list of responses, must select the most appropriate response from this list. This list *includes the actual next response* of the conversation, which is the desired prediction of the model. The other entries, which act as false positives, are sampled from elsewhere in the corpus. Note that no assumptions are made regarding the number of utterances in the context: these can be fixed or sampled from arbitrary distributions. Performance on this task is easy to assess by measuring the success rate of picking the correct next response; more specifically, we measure Recall@k (R@k), which is the percentage of correct responses (i.e. the actual response of the conversation) that are found in the top $k$ responses with the highest rankings according to the model. This task has gained some popularity recently for evaluating dialogue systems (Lowe et al., 2015; Kadlec et al., 2015).

There are several attractive properties of this approach, as detailed in the introduction: the performance is easy to compute automatically, the task is interpretable and amenable to comparison with human performance, and it is easier than generative dialogue modeling. A particularly nice property is that one can adjust the difficulty of NUC by simply changing the number of false responses (from one response to the full corpus), or by altering the selection criteria of false responses (from randomly sampled to intentionally confusing). Indeed, as the number of false responses grows to encompass all natural language responses, the task becomes identical to response generation.

One potential limitation of the NUC approach is that, since the other candidate answers are sampled from elsewhere in the corpus, these may also represent reasonable responses given the context. Part of the contribution of this work is determining the significance of this limitation.

### 4.2.4 Survey Methodology

#### 4.2.4.1 Corpora

We conducted our analysis on three corpora that have gained recent popularity for training dialogue systems. The SubTle Corpus (Banchs, 2012) consists of movie dialogues as extracted from subtitles, and includes turn-taking information indicating when each user has finished their turn. Unlike the larger OpenSubtitles[4] dataset, the SubTle Corpus includes turn-taking information indicating when each user has finished their turn. The Twitter Corpus (Ritter et al., 2010) contains a large number of conversations between users on the

[4] `http://www.opensubtitles.org`

| | |
|---|---|
| What is your gender? | |
| Male | 56.5% |
| Female | 44.5% |
| What is your age? | |
| 18-20 | 3.4% |
| 21-30 | 38.1% |
| 31-40 | 33.3% |
| 41-55 | 14.3% |
| 55+ | 10.2% |
| How would you rate your fluency in English? | |
| Beginner | 0% |
| Intermediate | 8.2% |
| Advanced | 6.8% |
| Fluent | 84.4% |
| What is your current level of education? | |
| High school or less | 21.1% |
| Bachelor's | 60.5% |
| Master's | 13.6% |
| Doctorate or higher | 3.4% |
| How would you rate your knowledge of Ubuntu? | |
| I've never used it | 70.7% |
| Basic | 21.8% |
| Intermediate | 5.4% |
| Expert | 2.7% |

TABLE 4.6: Data on the 145 AMT participants.

microblogging platform Twitter. Finally, the Ubuntu Dialogue Corpus contains conversations extracted from IRC chat logs (Lowe et al., 2015). [5] For more information on these datasets, we refer the reader to a recent survey on dialogue corpora (Serban et al., 2015). We focus our attention on these as they cover a range of popular domains, and are among the largest available dialogue datasets, making them good candidates for building data-driven dialogue systems. Note that while the Ubuntu Corpus is most relevant to supervised systems, the NUC task still applies in this domain. Models that take semantic information into account (i.e., to solve the user's problem) can still be validated with NUC.

A group of 145 paid participants were recruited through Amazon Mechanical Turk (AMT), a crowdsourcing platform for obtaining human participants for various studies. Demographic data including age, level of education, and fluency of English were collected from the AMT participants, and is shown in Table 4.6. An additional 8 volunteers were recruited from the student population in the computer science department at the author's institution.[6] This second

[5] `http://irclogs.ubuntu.com`

[6] None of these participants were directly involved with this research project.

group, referred to as "Lab experts", had significant exposure to technical terms prominent in the Ubuntu dataset; we hypothesized that this was an advantage in selecting responses for that corpus.

#### 4.2.4.2 Task description

Each participant was asked to answer either 30 or 40 questions (mean=31.9). To ensure a sufficient diversity of questions from each dataset, four versions of the survey with different questions were given to participants. For AMT respondents, the questions were approximately evenly distributed across the three datasets, while for the lab experts, half of the questions were related to Ubuntu and the remainder evenly split across Twitter and movies. Each question had 1 correct response, and 4 false responses drawn uniformly at random from elsewhere in the (same) corpus. An example question can be seen in Figure 4.3. Participants had a time limit of 40 minutes.

Conversations were extracted to form NUC conversation-response pairs as described in Sec. 4.2.3. The number of utterances in the context were sampled according to the procedure in (Lowe et al., 2015), with a maximum context length of 6 turns — this was done for both the human trials and artificial neural network (ANN) model. All conversations were pre-processed in order to anonymize the utterances. For the Twitter conversations, this was extended to replacing all user mentions (words beginning with @) throughout the utterance with a placeholder '@user' symbol, as these are often repeated in a conversation. Hashtags were not removed, as these are often used in the main body of tweets, and many tweets are illegible without them. Conversations were edited or pruned to remove offensive language according to ethical guidelines.

#### 4.2.4.3 ANN model

In order to compare human results with a strong artificial neural network (ANN) model, we use the dual encoder (DE) model from Lowe et al. (2015). This model uses recurrent neural networks (RNNs) with long-short term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) to encode the context $c$ of the conversation, and a candidate response $r$. More precisely, at each time step, a word $x_t$ is input into the LSTM, and its hidden state is updated. After all $T$

| | Number of Users | Movie Corpus R@1 | Movie Corpus R@2 | Twitter Corpus R@1 | Twitter Corpus R@2 | Ubuntu Corpus R@1 | Ubuntu Corpus R@2 |
|---|---|---|---|---|---|---|---|
| AMT non-experts | 135 | 65.9 ± 2.4% | 79.8 ± 2.1% | 74.1 ± 2.3% | 82.3 ± 2.0% | 52.9 ± 2.7% | 69.4 ± 2.5% |
| AMT experts | 10 | — | — | — | — | 52.0 ± 9.8% | 63.0 ± 9.5% |
| Lab experts | 8 | 69.7 ± 10% | 94.0 ± 5.2%* | 88.4 ± 7.0% | 98.4 ± 2.7%* | 83.8 ± 8.1% | 87.8 ± 7.2% |
| ANN model (Lowe et al., 2015a) | machine | 50.6% | 74.9% | 66.9% | 89.6% | 66.2% | 83.7% |

TABLE 4.7: Average results on each corpus. 'Number of Users' indicates the number of respondents for each category. 'AMT experts' and 'AMT non-experts' are combined for the Movie and Twitter corpora. 95% confidence intervals are calculated using the normal approximation, which assumes subjects answer each question independently of other examples and subjects. Starred (*) results indicate a poor approximation of the confidence interval due to high scores with small sample size, according to the rule of thumb by Brown et al. (2001).

words have been processed, the final hidden state $h_T$ can be considered a vector representation of the input sequence.

To determine the probability that a response $r$ is the actual next response to some context $c$, the model computes a weighted dot product between the vector representations $\mathbf{c}, \mathbf{r} \in \mathbb{R}^d$ of the context and response, respectively:

$$P(r \text{ is correct response}) = \sigma(\mathbf{c}^\top M \mathbf{r})$$

where $M$ is a matrix of learned parameters, and $\sigma$ is the sigmoid function. The model is trained to minimize the cross-entropy error of context-response pairs. For training negative examples are sampled randomly from elsewhere in the dataset.

The DE model is close to state-of-the-art for neural network dialogue models on the Ubuntu Dialogue Corpus; we obtained further results on the Movie and Twitter corpora in order to facilitate comparison with humans. For further model implementation details, see (Lowe et al., 2015).

### 4.2.5 Results

As we can see from Table 4.6, the AMT participants are mostly young adults, fluent in English with some undergraduate education. The split across genders is approximately equal, and the majority of respondents had never used Ubuntu before.

Table 4.7 shows the NUC results on each corpus. The human results are separated into AMT non-experts, consisting of paid respondents who have 'Beginner' or no knowledge of Ubuntu terminology; AMT experts, who claimed to have 'Intermediate' or 'Advanced' knowledge of Ubuntu; and Lab experts, who are the non-paid respondents with Ubuntu experience and university-level computer science training. We also presents results on the same task for a state-of-the-art artificial neural network (ANN) dialogue model (see (Lowe et al., 2015) for implementation details).

We first observe that subjects perform above chance level (20% for R@1) on all domains, thus the task is doable for humans. Second we observe difference in performances between the three domains. The Twitter dataset appears to have the best predictability, with a Recall@1 approximately 8% points higher than for the movie dialogues for AMT workers, and 18% higher for lab experts. Rather than attributing this to greater familiarity with Twitter than movies, it seems more likely that it is because movie utterances are often short, generic (e.g. contain few topic-related words), and lack proper context (e.g., video cues and the movie's story). Conversely, tweets are typically more specific, and successive tweets may have common hashtags.

As expected, untrained respondents scored lowest on the Ubuntu dataset, as it contains the most difficult language with often unfamiliar terminology. Further, since the domain is narrow, randomly drawn false responses could be more likely to resemble the actual next response, especially to someone unfamiliar with Ubuntu terminology. We also observe that the ANN model achieves similar performance to the paid human respondents from AMT. However, the model is still significantly behind the lab experts for Recall@1.

An interesting note is that there is very little difference between the paid AMT non-experts and AMT experts on Ubuntu. This suggests that the participants do not provide accurate self-rating of expertise, either intentionally or not. We also found that lab experts took on average approximately 50% more time to complete the survey than paid testers; this is reflected in the results, where the lab experts score 30% higher on the Ubuntu Corpus, and even 5-10% higher on the non-technical Movie and Twitter corpora. While we included attention check questions to ensure the quality of responses,[7] this reflects poorly on the

[7]Only the respondents who passed all attention checks were counted in the survey.

ability of crowdsourced workers to answer technical questions, even if they self-identify as being adept with the technology.

### 4.2.6 Discussion

Our results demonstrate that humans outperform current dialogue models on the task of next utterance classification, indicating that there is plenty of room for improvement for these models to better understand the nature of human dialogue. While our results suggest that NUC is a useful task, it is by no means sufficient; we strongly advocate for automatically evaluating dialogue systems with as many relevant metrics as possible. Further research should be conducted into finding metrics or tasks which accurately reflect human judgement for the evaluation of dialogue systems.

## 4.3 Learning to evaluate dialogue responses

### 4.3.1 Motivation

We've shown above that BLEU and other word-overlap metrics are biased and correlate poorly with human judgements of response quality (Liu et al., 2016). Despite this, there are few automatic alternatives available that correlate with human judgements.

To make progress towards the goal of developing useful automatic dialogue evaluation metrics, we make the simplifying assumption that a 'good' chatbot is one whose responses are scored highly on appropriateness by human evaluators. We believe this is sufficient for making progress as current dialogue systems often generate inappropriate responses. We also find empirically that asking evaluators for other metrics results in either low inter-annotator agreement, or the scores are highly correlated with appropriateness. Thus, we collect a dataset of appropriateness scores to various dialogue responses, and we use this dataset to train an *automatic dialogue evaluation model* (ADEM). The model is trained in a semi-supervised manner using a hierarchical recurrent neural network (RNN) to predict human scores. We show that ADEM scores correlate significantly with human judgement at both the utterance-level and system-level. We also show that ADEM can often generalize to evaluating new models, whose responses were unseen during training, making ADEM a strong first step towards effective automatic dialogue response evaluation.[8]

### 4.3.2 Data Collection

| | |
|---|---|
| # Examples | 4104 |
| # Contexts | 1026 |
| # Training examples | 2,872 |
| # Validation examples | 616 |
| # Test examples | 616 |
| $\kappa$ score (inter-annotator correlation) | 0.63 |

TABLE 4.8: Statistics of the dialogue response evaluation dataset. Each example is in the form *(context, model response, reference response, human score)*.

To train a model to predict human scores to dialogue responses, we first collect a dataset of human judgements (scores) of Twitter responses using the crowdsourcing platform Amazon Mechanical Turk (AMT).[9] The aim is to have accurate human scores for a variety of conversational responses — conditioned on dialogue contexts — which span the full range of response qualities. For example, the responses should include both relevant and irrelevant responses, both coherent and non-coherent responses and so on. To achieve this variety, we use candidate responses from several different models. Following Liu et al. (2016), we use the following 4 sources of candidate responses: (1) a response selected by a TF-IDF retrieval-based model, (2) a response selected by the Dual Encoder (DE) (Lowe et al., 2015), (3) a response generated using the hierarchical recurrent encoder-decoder (HRED) model (Serban et al., 2016), and (4) human-generated responses. It should be noted that the human-generated candidate responses are *not* the reference responses from a fixed corpus, but novel human responses that are different from the reference. In addition to increasing response variety, this is necessary because we want our evaluation model to learn to compare the reference responses to the candidate responses. Note that, in order to maximize the number of responses obtained with a

[8]Code and trained model parameters are available online: `https://github.com/mike-n-7/ADEM`.

[9]All data collection was conducted in accordance with the policies of the host institutions' ethics board.

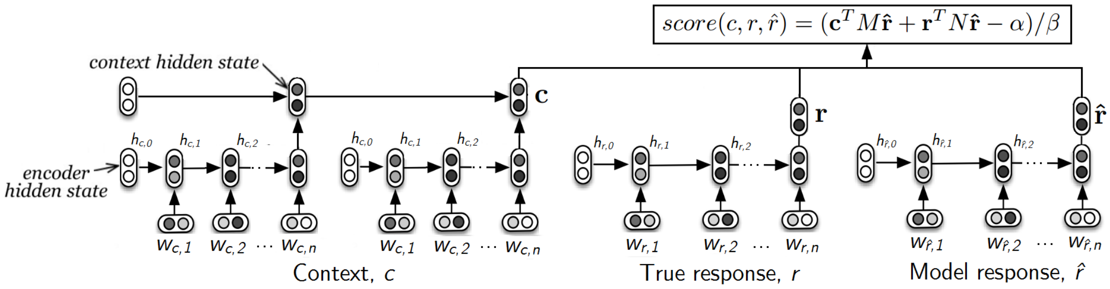



FIGURE 4.4: The ADEM model, which uses a hierarchical encoder to produce the context embedding **c**.

fixed budget, we only obtain one evaluation score per dialogue response in the dataset.

To train evaluation models on human judgements, it is crucial that we obtain scores of responses that lie near the distribution produced by advanced models. This is why we use the Twitter Corpus (Ritter et al., 2011), as such models are pre-trained and readily available. Further, the set of topics discussed is quite broad — as opposed to the very specific Ubuntu Dialogue Corpus (Lowe et al., 2015) — and therefore the model may also be suited to other chit-chat domains. Finally, since it does not require domain specific knowledge (e.g. technical knowledge), it should be easy for AMT workers to annotate.

### 4.3.3 An Automatic Dialogue Evaluation Model (ADEM)

To overcome the problems of evaluation with word-overlap metrics, we aim to construct a dialogue evaluation model that: (1) captures semantic similarity beyond word overlap statistics, and (2) exploits both the context and the reference response to calculate its score for the model response. We call this evaluation model ADEM.

ADEM learns distributed representations of the context, model response, and reference response using a hierarchical RNN encoder. Given the dialogue context $c$, reference response $r$, and model response $\hat{r}$, ADEM first encodes each of them into vectors ($\mathbf{c}$, $\hat{\mathbf{r}}$, and $\mathbf{r}$, respectively) using the RNN encoder. Then, ADEM computes the score using a dot-product between the vector representations of $c$, $r$, and $\hat{r}$ in a linearly transformed space:

$$score(c, r, \hat{r}) = (\mathbf{c}^T M \hat{\mathbf{r}} + \mathbf{r}^T N \hat{\mathbf{r}} - \alpha)/\beta \tag{4.6}$$

where $M, N \in \mathbb{R}^n$ are learned matrices initialized to the identity, and $\alpha, \beta$ are scalar constants used to initialize the model's predictions in the range $[1, 5]$. The model is shown in Figure 4.4.

The matrices $M$ and $N$ can be interpreted as linear projections that map the model response $\hat{\mathbf{r}}$ into the space of contexts and reference responses, respectively. The model gives high scores to responses that have similar vector representations to the context and reference response after this projection. The model is end-to-end differentiable; all the parameters can be learned by backpropagation. In our implementation, the parameters $\theta = \{M, N\}$ of the model are trained to minimize the squared error between the model predictions and the human score, with L2-regularization:

$$\mathcal{L} = \sum_{i=1:K} [score(c_i, r_i, \hat{r}_i) - human_i]^2 + \gamma ||\theta||_2 \tag{4.7}$$

where $\gamma$ is a scalar constant. The simplicity of our model leads to both accurate predictions and fast evaluation (see supp. material), which is important to allow rapid prototyping of dialogue systems.

The hierarchical RNN encoder in our model consists of two layers of RNNs (El Hihi and Bengio, 1995; Sordoni et al., 2015a). The lower-level RNN, the *utterance-level encoder*, takes as input words from the dialogue, and produces a vector output at the end of each utterance. The *context-level encoder* takes the representation of each utterance as input and outputs a vector representation of the context. This hierarchical structure is useful for incorporating information from early utterances in the context (Serban et al., 2016). Following previous work, we take the last hidden state of the context-level encoder as the vector representation of the input utterance or context. The parameters of the RNN encoder are pretrained and are not learned from the human scores.

An important point is that the ADEM procedure above *is not a dialogue retrieval model*: the fundamental difference is that ADEM has access to the reference response. Thus, ADEM can compare a model's response to a known good response, which is significantly easier than inferring response quality from solely the context.

**Pre-training with VHRED** We would like an evaluation model that can make accurate predictions from few labeled examples, since these examples are expensive to obtain. We therefore employ semi-supervised learning, and use a pre-training procedure to learn the parameters of the encoder. In particular, we train the encoder as part of a neural dialogue model; we attach a third *decoder RNN* that takes the output of the encoder as input, and train it to predict the next utterance of a dialogue conditioned on the context.

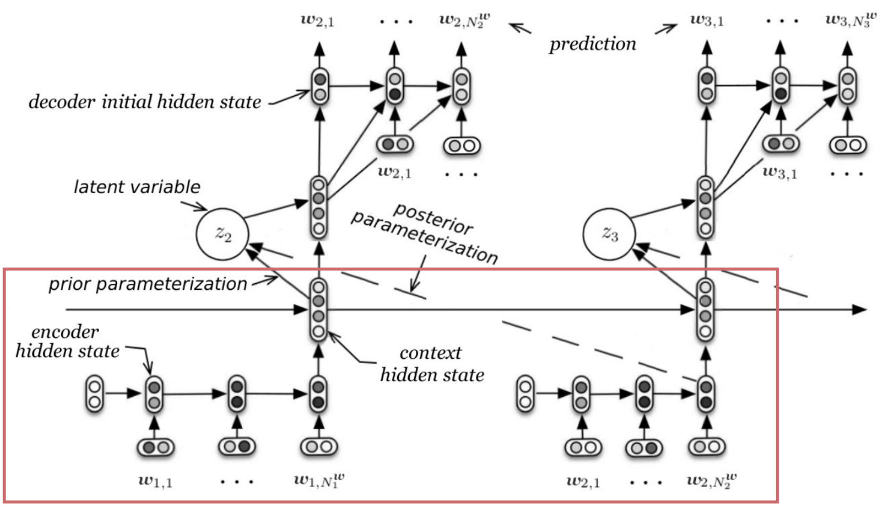
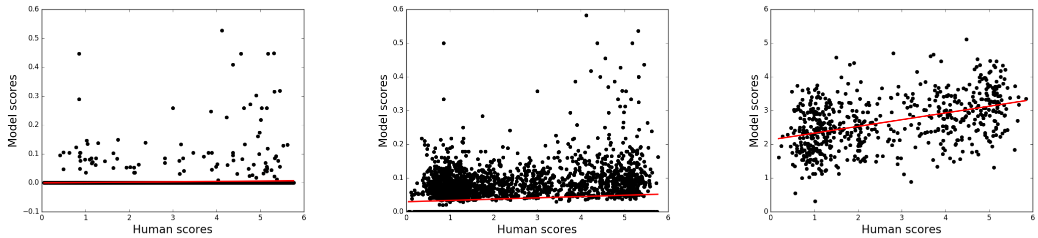



FIGURE 4.5: The VHRED model used for pre-training. The hierarchical structure of the RNN encoder is shown in the red box around the bottom half of the figure. After training using the VHRED procedure, the last hidden state of the context-level encoder is used as a vector representation of the input text.

The dialogue model we employ for pre-training is the latent variable hierarchical recurrent encoder-decoder (VHRED) model (Serban et al., 2017), shown in Figure 4.5. The VHRED model is an extension of the original hierarchical recurrent encoder-decoder (HRED) model (Serban et al., 2016) with a turn-level stochastic latent variable. The dialogue context is encoded into a vector using our hierarchical encoder, and the VHRED then samples a Gaussian variable that is used to condition the decoder. After training VHRED, we use the last hidden state of the context-level encoder, when $c$, $r$, and $\hat{r}$ are fed as input, as the vector representations for $\mathbf{c}, \mathbf{r}$, and $\hat{\mathbf{r}}$, respectively. We use representations from the VHRED model as it produces more diverse and coherent responses compared to HRED.




(A) BLEU-2 (B) ROUGE (C) ADEM

FIGURE 4.6: Scatter plot showing model against human scores, for BLEU-2 and ROUGE on the full dataset, and ADEM on the test set. We add Gaussian noise drawn from $\mathcal{N}(0, 0.3)$ to the integer human scores to better visualize the density of points, at the expense of appearing less correlated.

| | **Full dataset** | | **Test set** | |
|---|---|---|---|---|
| Metric | **Spearman** | **Pearson** | **Spearman** | **Pearson** |
| BLEU-2 | 0.039 (0.013) | 0.081 (<0.001) | 0.051 (0.254) | 0.120 (<0.001) |
| BLEU-4 | 0.051 (0.001) | 0.025 (0.113) | 0.063 (0.156) | 0.073 (0.103) |
| ROUGE | 0.062 (<0.001) | 0.114 (<0.001) | 0.096 (0.031) | 0.147 (<0.001) |
| METEOR | 0.021 (0.189) | 0.022 (0.165) | 0.013 (0.745) | 0.021 (0.601) |
| T2V | 0.140 (<0.001) | 0.141 (<0.001) | 0.140 (<0.001) | 0.141 (<0.001) |
| VHRED | -0.035 (0.062) | -0.030 (0.106) | -0.091 (0.023) | -0.010 (0.805) |
| | **Validation set** | | **Test set** | |
| C-ADEM | 0.338 (<0.001) | 0.355 (<0.001) | 0.366 (<0.001) | 0.363 (<0.001) |
| R-ADEM | 0.404 (<0.001) | 0.404 (<0.001) | 0.352 (<0.001) | 0.360 (<0.001) |
| ADEM (T2V) | 0.252 (<0.001) | 0.265 (<0.001) | 0.280 (<0.001) | 0.287 (<0.001) |
| ADEM | **0.410** (<0.001) | **0.418** (<0.001) | **0.428** (<0.001) | **0.436** (<0.001) |

TABLE 4.9: Correlation between metrics and human judgements, with p-values shown in brackets. 'ADEM (T2V)' indicates ADEM with tweet2vec embeddings (Dhingra et al., 2016), and 'VHRED' indicates the dot product of VHRED embeddings (i.e. ADEM at initialization). C- and R-ADEM represent the ADEM model trained to only compare the model response to the context or reference response, respectively. We compute the baseline metric scores (top) on the full dataset to provide a more accurate estimate of their scores (as they are not trained on a training set).

### 4.3.4 Experiments

#### 4.3.4.1 Experimental Procedure

In order to reduce the effective vocabulary size, we use byte pair encoding (BPE) (Gage, 1994; Sennrich et al., 2015), which splits each word into sub-words or characters. We also use layer normalization (Ba et al., 2016) for the hierarchical encoder, which we found worked better at the task of dialogue generation than the related recurrent batch normalization (Ioffe and Szegedy, 2015; Cooijmans et al., 2016). To train the VHRED model, we employed several of the same techniques found in (Serban et al., 2017) and (Bowman et al., 2016): we drop words in the decoder with a fixed rate of 25%, and we anneal the KL-divergence term linearly from 0 to 1 over the first 60,000 batches. We use Adam as our optimizer (Kingma and Ba, 2014).

When training ADEM, we also employ a sub-sampling procedure based on the model response length. In particular, we divide the training examples into bins based on the number of words in a response and the score of that response. We then over-sample from bins across the same score to ensure that ADEM does not use response length to predict the score. This is because humans have a tendency to give a higher rating to shorter responses than to longer responses (Serban et al., 2017), as shorter responses are often more generic and thus are more likely to be suitable to the context. Indeed, the test set Pearson correlation between response length and human score is 0.27.

For training VHRED, we use a context embedding size of 2000. However, we found the ADEM model learned more effectively when this embedding size was reduced. Thus, after training VHRED, we use principal component analysis (PCA) (Pearson, 1901) to reduce the dimensionality of the context, model response, and reference response embeddings to $n$. We found experimentally that $n = 50$ provided the best performance.

When training our models, we conduct early stopping on a separate validation set. For the evaluation dataset, we split the train/ validation/ test sets such that there is no context overlap (i.e. the contexts in the test set are unseen during training).

#### 4.3.4.2 Results

**Utterance-level correlations** We first present new utterance-level correlation results[10] for existing word-overlap metrics, in addition to results with embedding baselines and ADEM, in Table 4.9. The baseline metrics are evaluated on the entire dataset of 4,104 responses to provide the most accurate estimate of

[10]We present both the Spearman correlation (computed on ranks, depicts monotonic relationships) and Pearson correlation (computed on true values, depicts linear relationships) scores.

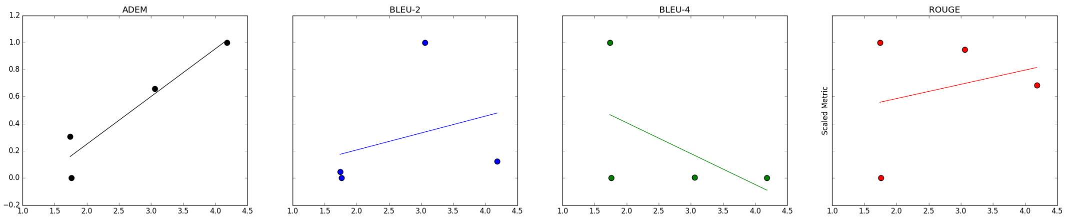



FIGURE 4.7: Scatterplots depicting the system-level correlation results for ADEM, BLEU-2, BLEU-4,and ROUGE on the test set. Each point represents the average scores for the responses from a dialogue model (TFIDF, DE, HRED, human). Human scores are shown on the horizontal axis, with normalized metric scores on the vertical axis. The ideal metric has a perfectly linear relationship.

the score. [11] We measure the correlation for ADEM on the validation and test sets, which constitute 616 responses each.

We can observe from the data in Table 4.9, that the correlations for the word-overlap metrics are even lower than estimated in previous studies (Liu et al., 2016; Galley et al., 2015). In particular, this is the case for BLEU-4, which has frequently been used for dialogue response evaluation (Ritter et al., 2011; Sordoni et al., 2015b; Li et al., 2015; Galley et al., 2015; Li et al., 2016a).

We can see from Table 4.9 that ADEM correlates far better with human judgement than the word-overlap baselines. This is further illustrated by the scatterplots in Figure 4.6. We also compare with ADEM using tweet2vec embeddings (Dhingra et al., 2016). In this case, instead of using the VHRED pre-training method described previously, we use off-the-shelf embeddings for $\mathbf{c}, \mathbf{r}$, and $\hat{\mathbf{r}}$, and fine-tune $M$ and $N$ on our dataset. These tweet2vec embeddings are computed at the character-level with a bidirectional GRU on a Twitter dataset for hashtag prediction (Dhingra et al., 2016). We find that they obtain reasonable but inferior performance compared to using VHRED embeddings.

**System-level correlations** We show the system-level correlations for various metrics in Table 4.10, and present it visually in Figure 4.7. Each point in the scatterplots represents a dialogue model; humans give low scores to TFIDF and DE responses, higher scores to HRED and the highest scores to other human responses. It is clear that existing word-overlap metrics are incapable of

[11] Note that our word-overlap correlation results in Table 4.9 are also lower than those presented in (Galley et al., 2015). This is because Galley et al. measure corpus-level correlation, i.e. correlation averaged across different subsets (of size 100) of the data, and pre-filter for high-quality reference responses.

capturing this relationship for even 4 models. This renders them completely deficient for dialogue evaluation. However, ADEM produces almost the same model ranking as humans, achieving a significant Pearson correlation of 0.954.[12] Thus, ADEM correlates well with humans both at the response and system level.

**Generalization to previously unseen models** When ADEM is used in practice, it will take as input responses from a new model that it has not seen during training. Thus, it is crucial that ADEM correlates with human judgements for new models. We test ADEM's generalization ability by performing a leave-one-out evaluation. For each dialogue model that was the source of response data for training ADEM (TF-IDF, Dual Encoder, HRED, humans), we conduct an experiment where we train on all model responses *except* those from the chosen model, and test *only* on the model that was unseen during training.

The results are given in Table 4.11. We observe that the ADEM model is able to generalize for all models except the Dual Encoder. This is particularly surprising for the HRED model; in this case, ADEM was trained only on responses that were written by humans (from retrieval models or human-generated), but is able to generalize to responses produced by a generative neural network model. When testing on the entire test set, the model achieves comparable correlations to the ADEM model that was trained on 25% less data selected at random.

**Qualitative Analysis** To illustrate some strengths and weaknesses of ADEM, we show human and ADEM scores for each of the responses to various contexts in Table 4.12. There are several instances where ADEM predicts accurately: in particular, ADEM is often very good at assigning low scores to poor responses. This seen in the first two contexts, where most of the responses given a score of 1 from humans are given scores less than 2 by ADEM. The single exception in response (4) for the second context seems somewhat appropriate and should perhaps have been scored higher by the human evaluator. There are also several instances where the model assigns high scores to suitable responses, as in the first two contexts.

One drawback we observed is that ADEM tends to be too conservative when predicting response scores. This is the case in the third context, where the

[12]For comparison, BLEU achieves a system-level correlation of 0.99 on 5 models in the translation domain (Papineni et al., 2002).

| Metric | Pearson |
|---|---|
| BLEU-1 | -0.079 (0.921) |
| BLEU-2 | 0.308 (0.692) |
| BLEU-3 | -0.537 (0.463) |
| BLEU-4 | -0.536 (0.464) |
| ROUGE | 0.268 (0.732) |
| ADEM | **0.954** (0.046) |

TABLE 4.10: System-level correlation, with the p-value in brackets.

| | **Test on full dataset** | | **Test on removed model responses** | |
|---|---|---|---|---|
| Data Removed | **Spearman** | **Pearson** | **Spearman** | **Pearson** |
| TF-IDF | 0.406 (<0.001) | 0.409 (<0.001) | 0.186 (0.021) | 0.196 (0.015) |
| Dual Encoder | 0.364 (<0.001) | 0.373 (<0.001) | 0.026 (0.749) | 0.027 (0.736) |
| HRED | 0.393 (<0.001) | 0.396 (<0.001) | 0.151 (0.060) | 0.148 (<0.070) |
| Human | 0.292 (<0.001) | 0.298 (<0.001) | 0.216 (<0.010) | 0.148 (<0.070) |
| **Average** | **0.364** | **0.369** | **0.145** | **0.130** |
| 25% at random | 0.378 (<0.001) | 0.384 (<0.001) | — | — |

TABLE 4.11: Correlation for ADEM when various model responses are removed from the training set. The left two columns show performance on the entire test set, and the right two columns show performance on responses only from the dialogue model not seen during training. The last row (25% at random) corresponds to the ADEM model trained on all model responses, but with the same amount of training data as the model above (i.e. 25% less data than the full training set).

model assigns low scores to most of the responses that a human rated highly. This behaviour is likely due to the squared error loss used to train ADEM; since the model receives a large penalty for incorrectly predicting an extreme value, it learns to predict scores closer to the average human score.

### 4.3.5 Related Work

Related to our approach is the literature on novel methods for the evaluation of machine translation systems, especially through the WMT evaluation task (Callison-Burch et al., 2011; Machácek and Bojar, 2014; Stanojevic et al., 2015). In particular, Albrecht and Hwa (2007); Gupta et al. (2015) have proposed to evaluate machine translation systems using Regression and Tree-LSTMs respectively. Their approach differs from ours as, in the dialogue domain, we must additionally condition our score on the context of the conversation, which is not necessary in translation.

| Context | Reference response | Model responses | Human score | ADEM score |
|---|---|---|---|---|
| photo to see my television debut go to - some. some on <url> - hehe <url> → it really was you? i thought ppl were recognizing someone who looked like you! were the oysters worth the wait? | yeah it was me . haha i 'd kinda forgotten about it it was filmed a while ago | 1) i'm not sure. i just don't know what to do with it.<br>2) you heard the horsepower productions remix of lee scratch perry's 'exercising' off his 'mighty upsetter' album?<br>3) you wont chug a fuzzy peach navel<br>4) they were! | 3<br>1<br>1<br>5 | 3.413<br>1.644<br>1.944<br>3.085 |
| just beat call of duty!! → want a cookie? → yes!! → come get it | im in kenmore at the moment | 1) i'm gonna get a new phone some moro<br>2) no way man.<br>3) wow i just got a free pizza coupon! get yours before theres no more! <url><br>4) i'm going to go to the mall. | 1<br>5<br>1<br>1 | 2.947<br>3.405<br>2.877<br>2.851 |
| am i out of twitter jail yet? testing → yeah. i posted bail → thanks. i am a right chatter tweetbox on sundays. same happened last sunday lol | any news on meeting our user ? i go to the us on friday and i don 't want to miss anything arranged | 1) i'm not sure if i'm going to be able to get it.<br>2) good to see another mac user in the leadership ranks<br>3) awww poor baby hope u get to feeling better soon. maybe some many work days at piedmont<br>4) did you tweet too much? | 3<br>4<br>2<br>5 | 2.651<br>2.775<br>2.173<br>3.185 |

TABLE 4.12: Examples of scores given by the ADEM model.

There has also been related work on estimating the quality of responses in chat-oriented dialogue systems. DeVault et al. (2011) train an automatic dialogue policy evaluation metric from 19 structured role-playing sessions, enriched with paraphrases and external referee annotations. Gandhe and Traum (2016) propose a semi-automatic evaluation metric for dialogue coherence, similar to BLEU and ROUGE, based on 'wizard of Oz' type data.[13] Xiang et al. (2014) propose a framework to predict utterance-level problematic situations in a dataset of Chinese dialogues using intent and sentiment factors. Finally, Higashinaka et al. (2014) train a classifier to distinguish user utterances from system-generated utterances using various dialogue features, such as dialogue acts, question types, and predicate-argument structures.

Several recent approaches use hand-crafted reward features to train dialogue models using reinforcement learning (RL). For example, Li et al. (2016b) use features related to ease of answering and information flow, and Yu et al. (2016) use metrics related to turn-level appropriateness and conversational depth. These metrics are based on hand-crafted features, which only capture a small set of relevant aspects; this inevitably leads to sub-optimal performance, and it is unclear whether such objectives are preferable over retrieval-based cross-entropy or word-level maximum log-likelihood objectives. Furthermore, many

[13] In 'wizard of Oz' scenarios, humans play the role of the dialogue system, usually unbeknown to the interlocutors.

of these metrics are computed at the conversation-level, and are not available for evaluating single dialogue responses. The metrics that can be computed at the response-level could be incorporated into our framework, for example by adding a term to equation 4.6 consisting of a dot product between these features and a vector of learned parameters.

There has been significant work on evaluation methods for task-oriented dialogue systems, which attempt to solve a user's task such as finding a restaurant. These methods include the PARADISE framework (Walker et al., 1997) and MeMo (Möller et al., 2006), which consider a task completion signal. PARADISE in particular is perhaps the first work on learning an automatic evaluation function for dialogue, accomplished through linear regression. However, PARADISE requires that one can measure task completion and task complexity, which are not available in our setting.

### 4.3.6 Discussion

#### 4.3.6.1 The problem of human appropriateness ratings

The evaluation model proposed in this chapter favours dialogue models that generate responses that are rated as highly appropriate by humans. It is likely that this property does not fully capture the desired end-goal of chatbot systems. For example, one issue with building models to approximate human judgements of response quality is the problem of generic responses. Since humans often provide high scores to generic responses due to their appropriateness for many given contexts (Shang et al., 2016), a model trained to predict these scores will exhibit the same behaviour. An important direction for future work is modifying ADEM such that it is not subject to this bias. This could be done, for example, by censoring ADEM's representations (Edwards and Storkey, 2016) such that they do not contain any information about length. Alternatively, one can combine this with an *adversarial evaluation model* (Kannan and Vinyals, 2017; Li et al., 2017) that assigns a score based on how easy it is to distinguish the dialogue model responses from human responses. In this case, a model that generates generic responses will easily be distinguishable and obtain a low score.

Another way to overcome this might be to give humans binary choice ratings (i.e. ask which of two responses is better), rather than asking them to give scalar scores on a Likert scale. Human evaluators could be explicitly instructed to prefer more specific responses to vaguer ones that could be used for any context. There is also evidence that binary choices elicit higher inter-annotator agreement in other domains (e.g. (Awad et al., 2014)), and this has also been adopted in dialogue evaluation (See et al., 2019).

#### 4.3.6.2 Evaluating full conversations

An important direction of future research is building models that can evaluate the capability of a dialogue system to have an engaging and meaningful interaction with a human. Compared to evaluating a single response, this evaluation is arguably closer to the end-goal of chatbots. However, such an evaluation is extremely challenging to do in a completely automatic way. We view the evaluation procedure presented in this chapter as a step towards this goal; current dialogue systems are incapable of generating responses that are rated as highly appropriate by humans, and we believe our evaluation model will be useful for measuring and facilitating progress in this direction.

#### 4.3.6.3 Reflecting on ADEM

Several years after the publishing of ADEM, the model hasn't seen much use in evaluating dialogue systems, even on Twitter. There are several reasons for this, which we detail in a written *retrospective* on the ADEM paper in (Lowe, 2019), which we summarize here. Essentially, despite the fact that ADEM generalizes on our Twitter test set and generalizes well to other models, it performs poorly on other datasets, such as a sanity check performed by an external researcher (Lowe, 2019). This anecdotal evidence is also supported by Sai et al. (2019), who find that ADEM is susceptible to adversarial examples, and other seemingly innocuous perturbations to the responses.

One thing that is not mentioned in the paper is that we performed a preliminary experiment testing our ADEM model on the Twitter and Ubuntu evaluation data from (Liu et al., 2016), and found that, on this data, ADEM didn't correlate with human judgements at all. While this was expected for Ubuntu, it was unexpected for Twitter —– we hypothesized that this was due to our data collection procedure being different, along with the fact that we used Turkers to evaluate responses in our ADEM paper and CS students to do the evaluations in (Liu et al., 2016). But this result should have been a red flag for us to probe more deeply into the model, and for full transparency, should have been mentioned in the paper.

The reason for the lack of generalization is likely two-fold: (1) the dataset really isn't that big, at only 1000 contexts and 4000 labeled responses; (2) there is a bias in the human responses towards being shorter. This is because we got the human responses from Mechanical Turkers, who satisfied the minimum criteria by generating reasonable responses that were as short as possible, as we did not incentivize longer responses with a bonus. In retrospect, it would have been beneficial to be more careful with data collection, including explicitly incentivizing the behaviour we wanted to occur.

We remain optimistic about the potential of learned models for evaluation. However, we believe that this will likely need to be done with much larger datasets than presented here, on the order of (at least) tens of thousands of responses. Further, this metric will likely need to be re-trained regularly, as dialogue systems practitioners build models that are more finely tuned to the evaluation model (and thus, will likely overfit). For this reason we also advocate for these learned models to be used in conjunction with learned 'discriminators', as described above.

# Part II

# Learning and evaluating emergent communication

# Chapter 5

# Centralized critics improve emergent communication and multi-agent learning

## 5.1 Motivation

In previous chapters, we've discussed methods of training and evaluating *dialogue systems* that are trained on large, static datasets. However, as discussed in Chapter 1, an important component of how humans use language is via *grounding* (Harnad, 1990). Humans use language to refer to things and accomplish tasks in the real world. In this part of the thesis, we investigate methods for training multiple agents to learn to communicate with each other in simulated environments. Since the *emergent communication* protocols learned by these agents are grounded in their environment, these settings are a useful stepping stone for understanding and building agents that speak grounded human language. Specifically in this chapter, we leverage the tools of *multi-agent reinforcement learning* to train agents to collaborate and compete in multiple tasks, including emerging a language from scratch (Sukhbaatar et al., 2016; Foerster et al., 2016; Mordatch and Abbeel, 2017). Successfully scaling RL to the multi-agent case is an important goal towards building artificially intelligent systems that can productively interact with humans and each other.

Unfortunately, traditional reinforcement learning approaches such as Q-Learning or policy gradient are poorly suited to multi-agent environments. One issue is that each agent's policy is changing as training progresses, and the environment becomes non-stationary from the perspective of any individual agent (in a way that is not explainable by changes in the agent's own policy). This presents learning stability challenges and prevents the straightforward use of past experience replay, which is crucial for stabilizing deep Q-learning. Policy gradient methods, on the other hand, usually exhibit very high variance when coordination of multiple agents is required. Alternatively, one can use model-based policy optimization which can learn optimal policies via back-propagation, but this requires a (differentiable) model of the world dynamics and assumptions about the interactions between agents. Applying these methods to competitive environments is also challenging from an optimization perspective, as evidenced by the notorious instability of adversarial training methods (Goodfellow et al., 2014).

In this chapter, we propose a general-purpose multi-agent learning algorithm that: (1) leads to learned policies that only use local information (i.e. their own observations) at execution time, (2) does not assume a differentiable model of the environment dynamics or any particular structure on the communication method between agents, and (3) is applicable not only to cooperative interaction but to competitive or mixed interaction involving both physical and communicative behavior. The ability to act in mixed cooperative-competitive environments may be critical for intelligent agents; while competitive training provides a natural curriculum for learning (Sukhbaatar et al., 2017), agents must also exhibit cooperative behavior (e.g. with humans) at execution time.

We adopt the framework of centralized training with decentralized execution, allowing the policies to use extra information to ease training, so long as this information is not used at test time. It is unnatural to do this with Q-learning without making additional assumptions about the structure of the environment, as the Q function generally cannot contain different information at training and test time. Thus, we propose a simple extension of actor-critic policy gradient methods where the critic is augmented with extra information about the policies of other agents, while the actor only has access to local information. After training is completed, only the local actors are used at execution phase, acting in a decentralized manner and equally applicable in cooperative and competitive settings.

Since the centralized critic function explicitly uses the decision-making policies of other agents, we additionally show that agents can learn approximate models of other agents online and effectively use them in their own policy learning procedure. We also introduce a method to improve the stability of multi-agent policies by training agents with an ensemble of policies, thus requiring robust interaction with a variety of collaborator and competitor policies. We empirically show the success of our approach compared to existing methods in cooperative as well as competitive scenarios, where agent populations are able to discover complex physical and communicative coordination strategies. Specifically, we find that our method improves learning speed dramatically in emerging a language to solve a simple multi-agent communication game.

## 5.2 Related Work

The simplest approach to learning in multi-agent settings is to use independently learning agents. This was attempted with Q-learning in (Tan, 1993), but does not perform well in practice (Matignon et al., 2012). As we will show, independently-learning policy gradient methods also perform poorly. One issue with independent Q-learning is that each agent's policy changes during training, resulting in a non-stationary environment and preventing the naïve application of experience replay. Previous work has attempted to address this by inputting other agent's policy parameters to the Q function (Tesauro, 2004), explicitly adding the iteration index to the replay buffer, or using importance sampling (Foerster et al., 2017b). Deep Q-learning approaches have previously been investigated in (Tampuu et al., 2017) to train competing Pong agents.

The nature of interaction between agents can either be cooperative, competitive, or both, and many algorithms are designed only for a particular nature of interaction. Most studied are cooperative settings, with strategies such as optimistic and hysteretic Q function updates (Lauer and Riedmiller, 2000; Matignon et al., 2007; Omidshafiei et al., 2017), which assume that the actions of other agents are made to improve collective reward. Another approach is to indirectly arrive at cooperation via sharing of policy parameters (Gupta et al., 2017), but this requires homogeneous agent capabilities. These algorithms are generally not applicable in competitive or mixed settings. See (Panait and Luke, 2005; Busoniu et al., 2008) for surveys of multi-agent learning approaches and applications.

Concurrently to our work, Foerster et al. (2017a) proposed a similar idea of using policy gradient methods with a centralized critic, and test their approach on a StarCraft micromanagement task. Their approach differs from ours in the following ways: (1) they learn a single centralized critic for all agents, whereas we learn a centralized critic for each agent, allowing for agents with differing reward functions including competitive scenarios, (2) we consider environments with explicit communication between agents, (3) they combine recurrent policies with feed-forward critics, whereas our experiments use feed-forward policies (although our methods are applicable to recurrent policies), (4) we learn continuous policies whereas they learn discrete policies.

Recent work has focused on learning grounded cooperative communication protocols between agents to solve various tasks (Sukhbaatar et al., 2016; Foerster et al., 2016; Mordatch and Abbeel, 2017). However, these methods are usually only applicable when the communication between agents is carried out over a dedicated, differentiable communication channel.

Our method requires explicitly modeling decision-making process of other agents. The importance of such modeling has been recognized by both reinforcement learning (Boutilier, 1996; Chalkiadakis and Boutilier, 2003) and cognitive science communities (Frank and Goodman, 2012). Hu and Wellman (1998) stressed the importance of being robust to the decision making process of other agents, as do others by building Bayesian models of decision making. We incorporate such robustness considerations by requiring that agents interact successfully with an ensemble of any possible policies of other agents, improving training stability and robustness of agents after training.

## 5.3 Background: High variance of policy gradient algorithms

Policy gradient methods are known to exhibit high variance gradient estimates (Ghavamzadeh and Engel, 2007; Gu et al., 2016). This is exacerbated in multi-agent settings; since an agent's reward usually depends on the actions of many agents, the reward conditioned only on the agent's own actions (when the actions of other agents are not considered in the agent's optimization process) exhibits much more variability, thereby increasing the variance of its gradients.

Below, we show a simple setting where the probability of taking a gradient step in the correct direction decreases exponentially with the number of agents, meaning that the gradients become progressively more useless as we add agents.

**Proposition 1.** *Consider $N$ agents with binary actions: $P(a_i = 1) = \theta_i$, where $R(a_1, \ldots, a_N) = \mathbf{1}_{a_1 = \cdots = a_N}$. We assume an uninformed scenario, in which agents are initialized to $\theta_i = 0.5\ \forall i$. Then, if we are estimating the gradient of the cost $J$ with policy gradient, we have:*

$$P(\langle \hat{\nabla} J, \nabla J \rangle > 0) \propto (0.5)^N$$

*where $\hat{\nabla} J$ is the policy gradient estimator from a single sample, and $\nabla J$ is the true gradient.*

*Proof.* We can write $P(a_i) = \theta_i^{a_i}(1-\theta_i)^{1-a_i}$, and $\log P(a_i) = a_i \log \theta_i + (1 - a_i)\log(1-\theta_i)$.

The policy gradient estimator (from a single sample) is:

$$\begin{aligned}
\frac{\hat{\partial}}{\partial \theta_i} J &= R(a_1, \ldots, a_N) \frac{\partial}{\partial \theta_i} \log P(a_1, \ldots, a_N) \\
&= R(a_1, \ldots, a_N) \frac{\partial}{\partial \theta_i} \sum_i a_i \log \theta_i + (1 - a_i) \log(1 - \theta_i) \\
&= R(a_1, \ldots, a_N) \frac{\partial}{\partial \theta_i} (a_i \log \theta_i + (1 - a_i) \log(1 - \theta_i)) \\
&= R(a_1, \ldots, a_N) \left( \frac{a_i}{\theta_i} - \frac{1 - a_i}{1 - \theta_i} \right)
\end{aligned} \tag{5.1}$$

For $\theta_i = 0.5$ we have:

$$\frac{\hat{\partial}}{\partial\theta_i}J = R(a_1, \ldots, a_N)\,(2a_i - 1)$$

And the expected reward can be calculated as:

$$\mathbb{E}(R) = \sum_{a_1,\ldots,a_N} R(a_1, \ldots, a_N)(0.5)^N$$

Consider the case where $R(a_1, \ldots, a_N) = \mathbf{1}_{a_1=\cdots=a_N=1}$. Then

$$\mathbb{E}(R) = (0.5)^N$$

and

$$\mathbb{E}(\frac{\hat{\partial}}{\partial\theta_i}J) = \frac{\partial}{\partial\theta_i}J = (0.5)^N$$

The variance of a single sample of the gradient is then:

$$\mathbb{V}(\frac{\hat{\partial}}{\partial\theta_i}J) = \mathbb{E}(\frac{\hat{\partial}}{\partial\theta_i}J^2) - \mathbb{E}(\frac{\hat{\partial}}{\partial\theta_i}J)^2 = (0.5)^N - (0.5)^{2N}$$

What is the probability of taking a step in the right direction? We can look at $P(\langle\hat{\nabla}J, \nabla J\rangle > 0)$. We have:

$$\langle\hat{\nabla}J, \nabla J\rangle = \sum_i \frac{\hat{\partial}}{\partial\theta_i}J \times (0.5)^N = (0.5)^N \sum_i \frac{\hat{\partial}}{\partial\theta_i}J,$$

so $P(\langle\hat{\nabla}J, \nabla J\rangle > 0) = (0.5)^N$. Thus, as the number of agents increases, the probability of taking a gradient step in the right direction decreases exponentially. □

While this is a somewhat artificial example, it serves to illustrate that there are simple environments that become progressively more difficult (in terms of the probability of taking a gradient step in a direction that increases reward) for policy gradient methods as the number of agents grows. This is particularly true in environments with sparse rewards, such as the one described above. Note that in this example, the policy gradient variance $\mathbb{V}(\frac{\hat{\partial}}{\partial\theta_i}J)$ actually decreases as N grows. However, the expectation of the policy gradient decreases as well, and

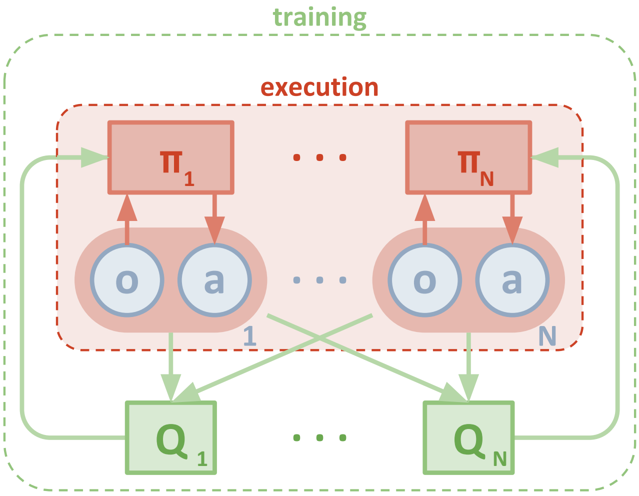



FIGURE 5.1: Overview of our multi-agent decentralized actor, centralized critic approach.

the signal to noise ratio $\mathbb{E}(\frac{\hat{\partial}}{\partial\theta_i}J)/(\mathbb{V}(\frac{\hat{\partial}}{\partial\theta_i}J))^{1/2}$ decreases with $N$, corresponding to the decreasing probability of a correct gradient direction.

## 5.4 Methods

We have argued in the previous section that naïve policy gradient methods perform poorly in simple multi-agent settings, and this is supported in our experiments in Section 5.5. Our goal in this section is to derive an algorithm that works well in such settings. However, we would like to operate under the following constraints: (1) the learned policies can only use local information (i.e. their own observations) at execution time, (2) we do not assume a differentiable model of the environment dynamics, unlike in Mordatch and Abbeel (2017), and (3) we do not assume any particular structure on the communication method between agents (that is, we don't assume a differentiable communication channel). Fulfilling the above desiderata would provide a general-purpose multi-agent learning algorithm that could be applied not just to cooperative games with explicit communication channels, but competitive games and games involving only physical interactions between agents.

### 5.4.1 Multi-Agent Actor Critic

Similarly to Foerster et al. (2016), we accomplish our goal by adopting the framework of centralized training with decentralized execution. Thus, we allow the policies to use extra information to ease training, so long as this information is not used at test time. It is unnatural to do this with Q-learning, as the Q function generally cannot contain different information at training and test time. Thus, we propose a simple extension of actor-critic policy gradient methods where the critic is augmented with extra information about the policies of other agents.

More concretely, consider a game with $N$ agents with policies parameterized by $\boldsymbol{\theta} = \{\theta_1, ..., \theta_N\}$, and let $\boldsymbol{\pi} = \{\boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_N\}$ be the set of all agent policies. Then we can write the gradient of the expected return for agent $i$, $J(\theta_i) = \mathbb{E}[R_i]$ as:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s \sim p^{\boldsymbol{\mu}}, a_i \sim \boldsymbol{\pi}_i}[\nabla_{\theta_i} \log \boldsymbol{\pi}_i(a_i|o_i) Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, ..., a_N)]. \tag{5.2}$$

Here $Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, ..., a_N)$ is a *centralized action-value function* that takes as input the actions of all agents, $a_1, \ldots, a_N$, in addition to some state information $\mathbf{x}$, and outputs the Q-value for agent $i$. In the simplest case, $\mathbf{x}$ could consist of the observations of all agents, $\mathbf{x} = (o_1, ..., o_N)$, however we could also include additional state information if available. Since each $Q_i^{\boldsymbol{\pi}}$ is learned separately, agents can have arbitrary reward structures, including conflicting rewards in a competitive setting.

We can extend the above idea to work with deterministic policies. If we now consider $N$ continuous policies $\boldsymbol{\mu}_{\theta_i}$ w.r.t. parameters $\theta_i$ (abbreviated as $\boldsymbol{\mu}_i$), the gradient can be written as:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x}, a \sim \mathcal{D}}[\nabla_{\theta_i} \boldsymbol{\mu}_i(a_i|o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, ..., a_N)|_{a_i = \boldsymbol{\mu}_i(o_i)}], \tag{5.3}$$

Here the experience replay buffer $\mathcal{D}$ contains the tuples $(\mathbf{x}, \mathbf{x}', a_1, \ldots, a_N, r_1, \ldots, r_N)$, recording experiences of all agents. The centralized action-value function $Q_i^{\boldsymbol{\mu}}$ is updated as:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x}, a, r, \mathbf{x}'}[(Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N) - y)^2], \quad y = r_i + \gamma \, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', a'_1, \ldots, a'_N)\big|_{a'_j = \boldsymbol{\mu}'_j(o_j)}, \tag{5.4}$$

where $\boldsymbol{\mu}' = \{\boldsymbol{\mu}_{\theta'_1}, ..., \boldsymbol{\mu}_{\theta'_N}\}$ is the set of target policies with delayed parameters $\theta'_i$. As shown in Section 5.5, we find the centralized critic with deterministic policies works very well in practice, and refer to it as *multi-agent deep deterministic policy gradient* (MADDPG). We provide the description of the full algorithm in the Appendix.

A primary motivation behind MADDPG is that, if we know the actions taken by all agents, the environment is stationary even as the policies change, since $P(s'|s, a_1, ..., a_N, \boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_N) = P(s'|s, a_1, ..., a_N) = P(s'|s, a_1, ..., a_N, \boldsymbol{\pi}'_1, ..., \boldsymbol{\pi}'_N)$ for any $\boldsymbol{\pi}_i \neq \boldsymbol{\pi}'_i$. This is not the case if we do not explicitly condition on the actions of other agents, as done for most decentralized RL methods.

Note that we require the policies of other agents to apply an update in Eq. 5.4. Knowing the observations and policies of other agents is not a particularly restrictive assumption; if our goal is to train agents to exhibit complex communicative behaviour in simulation, this information is often available to all agents. However, we can relax this assumption if necessary by learning the policies of other agents from observations — we describe a method of doing this in Section 5.4.2.

### 5.4.2 Inferring Policies of Other Agents

To remove the assumption of knowing other agents' policies, as required in Eq. 5.4, each agent $i$ can additionally maintain an approximation $\hat{\boldsymbol{\mu}}_{\phi_i^j}$ (where $\phi$ are the parameters of the approximation; henceforth $\hat{\boldsymbol{\mu}}_i^j$) to the true policy of agent $j$, $\boldsymbol{\mu}_j$. This approximate policy is learned by maximizing the log probability of agent $j$'s actions, with an entropy regularizer:

$$\mathcal{L}(\phi_i^j) = -\mathbb{E}_{o_j, a_j}\left[\log \hat{\boldsymbol{\mu}}_i^j(a_j|o_j) + \lambda H(\hat{\boldsymbol{\mu}}_i^j)\right], \tag{5.5}$$

where $H$ is the entropy of the policy distribution. With the approximate policies, $y$ in Eq. 5.4 can be replaced by an approximate value $\hat{y}$ calculated as follows:

$$\hat{y} = r_i + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', \hat{\boldsymbol{\mu}}_i'^1(o_1), \ldots, \boldsymbol{\mu}'_i(o_i), \ldots, \hat{\boldsymbol{\mu}}_i'^N(o_N)), \tag{5.6}$$

where $\hat{\boldsymbol{\mu}}_i'^j$ denotes the target network for the approximate policy $\hat{\boldsymbol{\mu}}_i^j$. Note that Eq. 5.5 can be optimized in a completely online fashion: before updating $Q_i^{\boldsymbol{\mu}}$, the centralized Q function, we take the latest samples of each agent $j$ from the replay buffer to perform a single gradient step to update $\phi_i^j$. Note also that, in the above equation, we input the action log probabilities of each agent directly into $Q$, rather than sampling.

### 5.4.3 Agents with Policy Ensembles

As previously mentioned, a recurring problem in multi-agent reinforcement learning is the environment non-stationarity due to the agents' changing policies. This is particularly true in competitive settings, where agents can derive a strong policy by overfitting to the behavior of their competitors. Such policies are undesirable as they are brittle and may fail when the competitors alter strategies.

To obtain multi-agent policies that are more robust to changes in the policy of competing agents, we propose to train a collection of $K$ different sub-policies. At each episode, we randomly select one particular sub-policy for each agent to execute. Suppose that policy $\boldsymbol{\mu}_i$ is an ensemble of $K$ different sub-policies with sub-policy $k$ denoted by $\boldsymbol{\mu}_{\theta_i^{(k)}}$ (denoted as $\boldsymbol{\mu}_i^{(k)}$). For agent $i$, we are then maximizing the ensemble objective: $J_e(\boldsymbol{\mu}_i) = \mathbb{E}_{k\sim\text{unif}(1,K), s\sim p^{\boldsymbol{\mu}}, a\sim\boldsymbol{\mu}_i^{(k)}}\left[R_i(s,a)\right].$

Since different sub-policies will be executed in different episodes, we maintain a replay buffer $\mathcal{D}_i^{(k)}$ for each sub-policy $\boldsymbol{\mu}_i^{(k)}$ of agent $i$. Accordingly, we can derive the gradient of the ensemble objective with respect to $\theta_i^{(k)}$ as follows:

$$\nabla_{\theta_i^{(k)}} J_e(\boldsymbol{\mu}_i) = \frac{1}{K}\mathbb{E}_{\mathbf{x},a\sim\mathcal{D}_i^{(k)}}\left[\nabla_{\theta_i^{(k)}}\boldsymbol{\mu}_i^{(k)}(a_i|o_i)\nabla_{a_i}Q^{\boldsymbol{\mu}_i}\left(\mathbf{x}, a_1, \ldots, a_N\right)\Big|_{a_i=\boldsymbol{\mu}_i^{(k)}(o_i)}\right]. \tag{5.7}$$

## 5.5 Experiments

### 5.5.1 Environments

To perform our experiments, we adopt the grounded communication environment proposed in (Mordatch and Abbeel, 2017)[1], which consists of $N$ agents

[1]Code for our Particle World environments can be found at: https://github.com/openai/multiagent-particle-envs. Code for implementing MADDPG can be found at: https://github.com/openai/maddpg

and $L$ landmarks inhabiting a two-dimensional world with continuous space and discrete time. Agents may take physical actions in the environment and communication actions that get broadcasted to other agents. Unlike Mordatch and Abbeel (2017), we do not assume that all agents have identical action and observation spaces, or act according to the same policy $\boldsymbol{\pi}$. We also consider games that are both cooperative (all agents must maximize a shared return) and competitive (agents have conflicting goals). Some environments require explicit communication between agents in order to achieve the best reward, while in other environments agents can only perform physical actions. We provide details for each environment below.

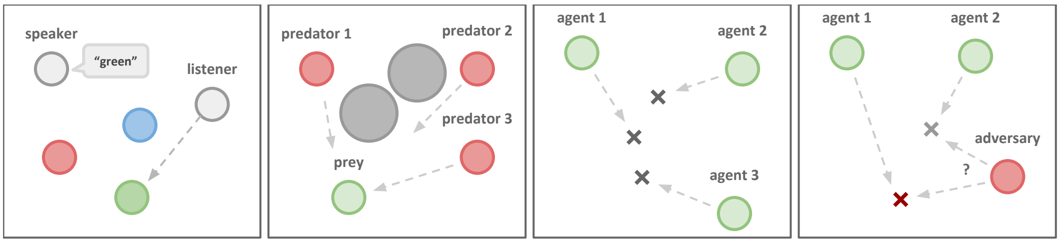



FIGURE 5.2: Illustrations of the experimental environment and some tasks we consider, including a) *Cooperative Communication* b) *Predator-Prey* c) *Cooperative Navigation* d) *Physical Deception*. See webpage for videos of all experimental results.

**Cooperative communication.** This task consists of two cooperative agents, a speaker and a listener, who are placed in an environment with three landmarks of differing colors. At each episode, the listener must navigate to a landmark of a particular color, and obtains reward based on its distance to the correct landmark. However, while the listener can observe the relative position and color of the landmarks, it does not know which landmark it must navigate to. Conversely, the speaker's observation consists of the correct landmark color, and it can produce a communication output at each time step which is observed by the listener. Thus, the speaker must learn to output the landmark colour based on the motions of the listener. Although this problem is relatively simple, as we show in Section 5.5.2 it poses a significant challenge to decentralized RL algorithms.

**Cooperative navigation.** In this environment, agents must cooperate through physical actions to reach a set of $L$ landmarks. Agents observe the relative positions of other agents and landmarks, and are collectively rewarded based on the proximity of any agent to each landmark. In other words, the agents have to 'cover' all of the landmarks. Further, the agents occupy significant physical space and are penalized when colliding with each other. Our agents learn to infer the landmark they must cover, and move there while avoiding other agents.

**Keep-away.** This scenario consists of $L$ landmarks including a target landmark, $N$ cooperating agents who know the target landmark and are rewarded based on their distance to the target, and $M$ *adversarial* agents who must prevent the cooperating agents from reaching the target. Adversaries accomplish this by physically pushing the agents away from the landmark, temporarily occupying it. While the adversaries are also rewarded based on their distance to the target landmark, they do not know the correct target; this must be inferred from the movements of the agents.

**Physical deception.** Here, $N$ agents cooperate to reach a single target landmark from a total of $N$ landmarks. They are rewarded based on the minimum distance of any agent to the target (so only one agent needs to reach the target landmark). However, a lone adversary also desires to reach the target landmark; the catch is that the adversary does not know which of the landmarks is the correct one. Thus the cooperating agents, who are penalized based on the adversary distance to the target, learn to spread out and cover all landmarks so as to deceive the adversary.

**Predator-prey.** In this variant of the classic predator-prey game, $N$ slower cooperating agents must chase the faster adversary around a randomly generated environment with $L$ large landmarks impeding the way. Each time the cooperative agents collide with an adversary, the agents are rewarded while the adversary is penalized. Agents observe the relative positions and velocities of the agents, and the positions of the landmarks.

**Covert communication.** This is an adversarial communication environment, where a speaker agent ('Alice') must communicate a message to a listener agent ('Bob'), who must reconstruct the message at the other end. However, an adversarial agent ('Eve') is also observing the channel, and wants to reconstruct the message — Alice and Bob are penalized based on Eve's reconstruction, and thus Alice must encode her message using a randomly generated *key*, known only to Alice and Bob. This is similar to the cryptography environment considered in Abadi and Andersen (2016).

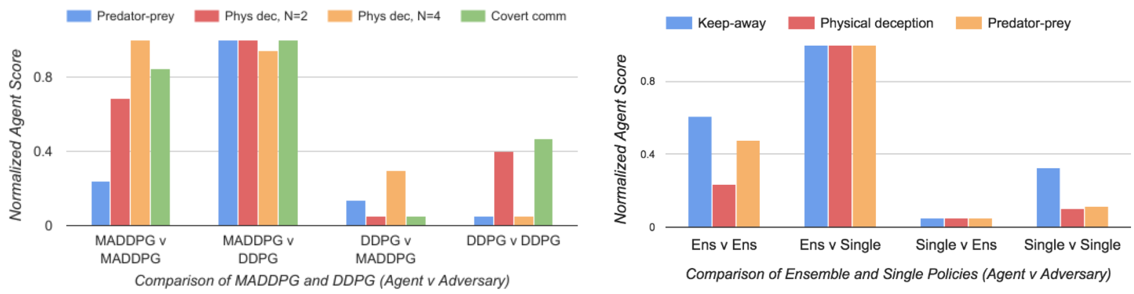



FIGURE 5.3: Comparison between MADDPG and DDPG (left), and between single policy MADDPG and ensemble MADDPG (right) on the competitive environments. Each bar cluster shows the 0-1 normalized score for a set of competing policies (agent v adversary), where a higher score is better for the agent. In all cases, MADDPG outperforms DDPG when directly pitted against it, and similarly for the ensemble against the single MADDPG policies. Full results are given in the Appendix.

### 5.5.2 Comparison to Decentralized Reinforcement Learning Methods

We implement our MADDPG algorithm and evaluate it on the environments presented in Section 5.5.1.[2] Unless otherwise specified, our policies are parameterized by a two-layer ReLU MLP with 64 units per layer. To support discrete communication messages, we use the Gumbel-Softmax estimator (Jang et al., 2016). To evaluate the quality of policies learned in competitive settings, we pitch MADDPG agents against DDPG agents, and compare the resulting success of the agents and adversaries in the environment. We train our models until convergence, and then evaluate them by averaging various metrics for 1000 further iterations. We summarize our results here.

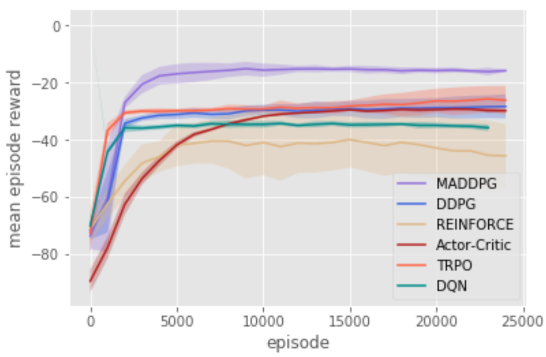
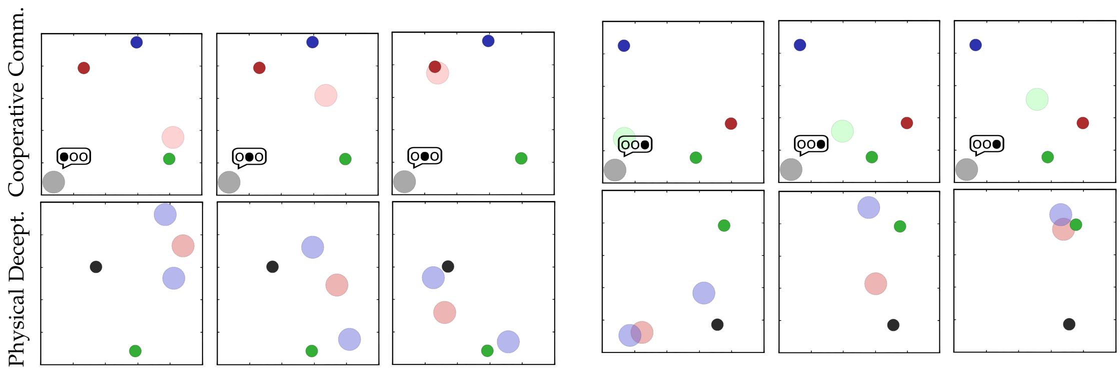



FIGURE 5.4: Agent reward on cooperative communication after 25000 episodes.

We first examine the cooperative communication scenario. Despite the simplicity of the task (the speaker only needs to learn to output its observation),

[2]Videos of our experimental results can be viewed at https://sites.google.com/site/multiagentac/




(A) MADDPG (B) DDPG

FIGURE 5.5: Comparison between MADDPG (left) and DDPG (right) on the cooperative communication (CC) and physical deception (PD) environments at $t = 0$, 5, and 25. Small dark circles indicate landmarks. In CC, the grey agent is the speaker, and the color of the listener indicates the target landmark. In PD, the blue agents are trying to deceive the red adversary, while covering the target landmark (in green). MADDPG learns the correct behavior in both cases: in CC the speaker learns to output the target landmark color to direct the listener, while in PD the agents learn to cover both landmarks to confuse the adversary. DDPG (and other RL algorithms) struggles in these settings: in CC the speaker always repeats the same utterance and the listener moves to the middle of the landmarks, and in PP one agent greedily pursues the green landmark (and is followed by the adversary) while the othe agent scatters. See video for full trajectories.

decentralized RL methods such as DQN, Actor-Critic, a first-order implementation of TRPO, and DDPG all fail to learn the correct behaviour (measured by whether the listener is within a short distance from the target landmark). In practice we observed that the listener learns to ignore the speaker and simply moves to the middle of all observed landmarks. We plot the learning curves over 25000 episodes for various approaches in Figure 5.4.

We hypothesize that a primary reason for the failure of decentralized RL methods in this (and other) multi-agent settings is the lack of a consistent gradient signal. For example, if the speaker utters the correct symbol while the listener moves in the wrong direction, the speaker is penalized. This problem is exacerbated as the number of time steps grows: we observed that decentralized policy gradient methods can learn when the objective of the listener is simply to reconstruct the observation of the speaker in a single time step, or if the initial positions of agents and landmarks are fixed and evenly distributed. This indicates that many of the multi-agent methods previously proposed for scenarios with short time horizons (e.g. (Lazaridou et al., 2016)) may not generalize to more complex tasks.


FIGURE 5.6: Policy learning success rate on cooperative communication after 25000 episodes.

Conversely, MADDPG agents can learn coordinated behaviour more easily via the centralized critic. In the cooperative communication environment, MADDPG is able to reliably learn the correct listener and speaker policies, and the listener is often (84.0% of the time) able to navigate to the target.

A similar situation arises for the physical deception task: when the cooperating agents are trained with MADDPG, they are able to successfully deceive the adversary by covering all of the landmarks around 94% of the time when $L = 2$ (Figure 5). Furthermore, the adversary success is quite low, especially when the adversary is trained with DDPG (16.4% when $L = 2$). This contrasts sharply with the behaviour learned by the cooperating DDPG agents, who are unable to deceive MADDPG adversaries in any scenario, and do not even deceive other DDPG agents when $L = 4$.

While the cooperative navigation and predator-prey tasks have a less stark divide between success and failure, in both cases the MADDPG agents outperform the DDPG agents. In cooperative navigation, MADDPG agents have a slightly smaller average distance to each landmark, but have almost half the average number of collisions per episode (when $N = 2$) compared to DDPG agents due to the ease of coordination. Similarly, MADDPG predators are far more successful at chasing DDPG prey (16.1 collisions/episode) than the converse (10.3 collisions/episode).

In the covert communication environment, we found that Bob trained with both MADDPG and DDPG out-performs Eve in terms of reconstructing Alice's message. However, Bob trained with MADDPG achieves a larger relative success rate compared with DDPG (52.4% to 25.1%). Further, only Alice trained with MADDPG can encode her message such that Eve achieves near-random reconstruction accuracy. The learning curve (a sample plot is shown in Appendix) shows that the oscillation due to the competitive nature of the environment often cannot be overcome with common decentralized RL methods. We emphasize that we do not use any of the tricks required for the cryptography

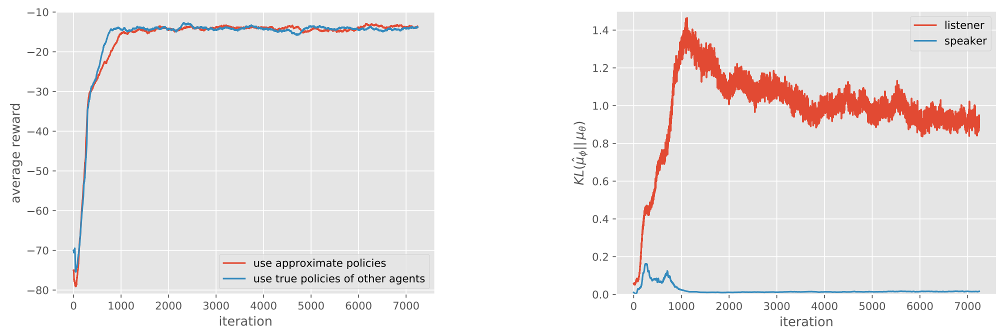



FIGURE 5.7: Effectiveness of learning by approximating policies of other agents in the cooperative communication scenario. *Left:* plot of the reward over number of iterations; MADDPG agents quickly learn to solve the task when approximating the policies of others. *Right:* KL divergence between the approximate policies and the true policies.

environment from Abadi and Andersen (2016), including modifying Eve's loss function, alternating agent and adversary training, and using a hybrid 'mix & transform' feed-forward and convolutional architecture.

### 5.5.3 Effect of Learning Polices of Other Agents

We evaluate the effectiveness of learning the policies of other agents in the cooperative communication environment, following the same hyperparameters as the previous experiments and setting $\lambda = 0.001$ in Eq. 5.5. The results are shown in Figure 5.7. We observe that despite not fitting the policies of other agents perfectly (in particular, the approximate listener policy learned by the speaker has a fairly large KL divergence to the true policy), learning with approximated policies is able to achieve the same success rate as using the true policy, without a significant slowdown in convergence.

### 5.5.4 Effect of Training with Policy Ensembles

We focus on the effectiveness of policy ensembles in competitive environments, including keep-away, cooperative navigation, and predator-prey. We choose $K = 3$ sub-policies for the keep-away and cooperative navigation environments, and $K = 2$ for predator-prey. To improve convergence speed, we enforce that the cooperative agents should have the same policies at each episode, and similarly for the adversaries. To evaluate the approach, we measure the performance of ensemble policies and single policies in the roles of both agent and adversary. The results are shown on the right side of Figure 5.3. We observe that agents with policy ensembles are stronger than those with a single policy. In particular, when pitting ensemble agents against single policy adversaries (second to left bar cluster), the ensemble agents outperform the adversaries by a large margin compared to when the roles are reversed (third to left bar cluster).

## 5.6 Discussion

We have proposed a multi-agent policy gradient algorithm where agents learn a centralized critic based on the observations and actions of all agents. Empirically, our method outperforms decentralized RL algorithms on a variety of cooperative and competitive multi-agent environments. We can further improve the performance of our method by training agents with an ensemble of policies, an approach we believe to be generally applicable to any multi-agent algorithm.

One downside to our approach is that the input space of $Q$ grows linearly (depending on what information is contained in $\mathbf{x}$) with the number of agents $N$. This could be remedied in practice by, for example, having a modular Q function that only considers agents in a certain neighborhood of a given agent. This downside has been addressed in subsequent work, e.g. (Chu and Ye, 2017).

# Chapter 6

# The pitfalls of measuring emergent communication

## 6.1 Motivation

In the previous chapter, we advance methods for multi-agent RL applied to emergent communication settings. Concurrently and subsequently to the publishing of this work (Lowe et al., 2017c), there has been a large number of papers written about emergent communication in various settings, from cooperative *referential games* (Das et al., 2017; Lazaridou et al., 2018; Kottur et al., 2017; Choi et al., 2018; Havrylov and Titov, 2017; Evtimova et al., 2017), to navigating a simulated world (Sukhbaatar et al., 2016; Jaques et al., 2018a; Foerster et al., 2018; Bogin et al., 2018), to games with pixel-based inputs (Lazaridou et al., 2018; Choi et al., 2018; Bogin et al., 2018), to settings incorporating human language (Lazaridou et al., 2016; Lee et al., 2017), and to even more complex games that require inferring the belief state of the opponent, like Hanabi (Foerster et al., 2018).

Despite these advances, there has been no formal study evaluating *how we measure* emergent communication. Most papers in this area show that adding a communication channel helps the agents achieve a higher reward, and attempt to understand the communication protocols qualitatively by, for example, plotting the distribution over messages in various states of the environment. This is a reasonable approach: if adding a communication channel increases the reward in an environment, then the agents are making some use of that channel. But it is useful to quantify more finely the degree of communication as it can provide more insights into agent behaviour. This may be essential for human monitoring; we want to understand why agents are making certain decisions, and understanding their language would make this significantly easier. Our aim is to develop a set of evaluation tools for detecting and measuring emergent communication that increases the robustness of emergent communication research on the way to developing agents that communicate with humans.

In this chapter, we take a step in this direction by analyzing some intuitive metrics and tests that have been used in the recent literature for measuring emergent communication. In particular, we show that some of these metrics, including speaker consistency (Jaques et al., 2018a) and simply examining the agents' policies qualitatively, can be misleading if not properly understood. We use a simple class of matrix games augmented with a bidirectional communication channel, which we call Matrix Communication Games (MCGs), to study the emergence of communication among deep RL agents. MCGs are simple and efficient for learning, and have the appealing property that the resulting communication policies are interpretable relative to more complex gridworld environments.

To aid in our analysis, we categorize emergent communication metrics into two broad classes: those that measure *positive signaling*, which indicates that an agent is sending messages that are related in some way with its observation or action; and those that measure *positive listening*, indicating that the messages are influencing the agents' behaviour in some way. Both positive signaling and positive listening are desirable; we want to develop communicative agents that can both speak about their world and intentions, and can interpret the messages of others and respond accordingly. While intuitively it might seem that positive listening is a prerequisite for positive signaling (otherwise, why would the agent signal in the first place?), we show that, surprisingly, this is not always true for deep RL agents; it is possible for positive signaling to emerge as a byproduct of the task and policy architecture.

## 6.2 Emergent communication metrics

### 6.2.1 A categorization of metrics

When analyzing metrics to measure a certain quantity, it is important to ask what that quantity actually represents. What does it mean for agents to be communicating with each other in a reinforcement learning setting? We take a pragmatic perspective, and identify two broad prerequisites for communication to occur: (1) one agent needs to produce a signal that is in some way correlated with its observation or intended action, and (2) another agent needs to update its belief or alter its behaviour after observing the signal.

We define *positive signaling* as behaviour that satisfies criterion (1), and *positive listening* as behaviour that satisfies criterion (2). To formalize these intuitions, we provide very broad definitions of these terms below.

**Definition 6.1** (Positive Signaling)**.** Let $\bar{m} = (m_0, m_1, ..., m_T)$ be the sequence of messages sent by an agent over the course of a trajectory of length $T$, and similarly for $\bar{o} = (o_0, o_1, ..., o_t)$ and $\bar{a} = (a_0, a_1, ..., a_T)$. An RL agent exhibits positive signaling if either $\bar{m} \not\perp \bar{o}$ or if $\exists t > t' s.t. m_{t'} \not\perp a_t$, i.e. if $\bar{m}$ is statistically dependent (indicated by $\not\perp$ ) in some way on $\bar{o}$, or if $m$ is statistically dependent on a future action $a$.

**Definition 6.2** (Positive Listening)**.** An RL agent exhibits positive listening if there exists a message generated by another agent $m \in \mathcal{A}_i^m$, for some $i \in \{1, ..., N\}$ such that $||\boldsymbol{\pi}(o, \mathbf{0}) - \boldsymbol{\pi}(o, m)||_\tau > 0$, where $\mathbf{0}$ is the 0 vector, and $|| \cdot ||_\tau$ is a distance in the space of expected trajectories followed by $\boldsymbol{\pi}$.

Evidently, these definitions are very loose, and most agents in an multi-agent environment with the capacity to communicate will satisfy them to some degree. However, we can speak of the degree or extent to which an agent exhibits positive signaling or positive listening behaviour, and measure this using metrics or tests. Thus, these terms are useful for categorizing different metrics of emergent communication.

### 6.2.2 What metrics are being used now?

We now conduct a brief review of the existing metrics being used in papers applying deep reinforcement learning to the problem of emergent communication, and categorize them as being indicative of positive signaling or positive listening. We focus on metrics that are the most prevalent in the recent deep RL literature to give an overview of how current research in this area is being conducted. From our review, we find that only one metric (instantaneous coordination) is explicitly designed to measure positive listening, and that it has several shortcomings. This motivates our definition of a new metric, the causal influence of communication, in Section 6.2.3.

**Reward and task completion** As previously mentioned, in all papers we surveyed, the authors either measure the task completion % of their proposed algorithm (for referential games (Lazaridou et al., 2016, 2018; Evtimova et al., 2017)), or show that adding a communication channel increases the agents' reward (for games where the agents interact with the world (Bogin et al., 2018; Choi et al., 2018; Sukhbaatar et al., 2016)). For referential games (Das et al., 2017; Lazaridou et al., 2018), this is an adequate measure of communicative performance because these games are *non-situated*. In the terminology of Wagner et al. (2003), a non-situated environment (or simulation) is one where agents' actions consist solely of sending and receiving signals. Non-embodied agents in these environments do not have non-communicative actions that affect other objects or each other. Thus, if task success increases in this setting, it is likely because the sender agent has developed a better communication protocol, or the listener agent has become better at understanding it.

In *situated* environments (Wagner et al., 2003), where agents have non-communicative actions that affect the environment and/or modify their internal state, an increase in reward is still a loose measure of both positive listening and positive signaling. If adding a communication channel leads to an increase in reward, then agents must be using that channel to transmit some kind of information that is affecting their behaviour. However, when comparing multiple algorithms on the same environment, an increase in reward may not necessarily indicate improved communication of one algorithm over another, as this increase could be simply due to the improved action policies. Thus, in this chapter we advocate for more fine-grained metrics for measuring the quality of emergent communication protocols. One of the reasons for our adoption of MCGs is that they are in some ways the simplest situated environment (agents have non-communicative interactions that affect each other's reward).

**Qualitative analysis of messages given states** An equally common practice is to analyze the communication policies qualitatively, to interpret what the agents are saying. This is done most commonly by visualizing which messages were sent with which inputs or observations (Sukhbaatar et al., 2016; Lazaridou et al., 2016; Havrylov and Titov, 2017; Jaques et al., 2018a; Bogin et al., 2018). For very simple environments, such as the riddle game (Foerster et al., 2016), the authors are able to make a full tree describing the agents' policies for all states in the game. However, for other games with larger state spaces, papers simply show some messages and inputs that the authors observed to have high co-occurrence. While this is a useful practice to understand the agents' behaviour, it is usually only testing positive signaling, as it does not analyze how the message changes the other agent's behaviour.

**Speaker consistency** Some papers quantify the degree of alignment between an agent's messages and its actions (Bogin et al., 2018; Jaques et al., 2018a). One such example is the *speaker consistency* (SC), proposed in Jaques et al. (2018a). The easiest way to understand the speaker consistency is as the mutual information between an agent's message and its future action:

$$SC = \sum_{a \in \mathcal{A}^e} \sum_{m \in \mathcal{A}^m} p(a, m) \log \frac{p(a, m)}{p(a)p(m)} \tag{6.1}$$

where the probabilities $p(a, m) = \frac{1}{N} \sum_{i=1}^{N} 1_{\{\text{act}=a, \text{comm}=m\}}$ are calculated empirically by averaging (message, action) co-occurrences over the $N$ episodes in each epoch.[1] The speaker consistency is a measure of positive signaling, as it indicates a statistical relationship between the messages and actions of an agent, and does not tell us anything about how these messages are interpreted.

On the surface, SC is a useful measure of communication because it tells us how much an agent's message reduces the uncertainty about its subsequent

[1] The SC metric in Jaques et al. (2018a) is a normalized version of the metric shown here.

action. Importantly, when an agent learns a deterministic policy independent of observations, the SC will be 0. Because of these appealing properties, we primarily focus on SC as our metric of comparison in our experiments.

**Context independence** Context independence (CI), introduced by Bogin et al. (2018), is designed to measure the degree of alignment between an agent's messages and task concepts (corresponding to a particular part of the input space). It is calculated as follows:

$$\forall c : \quad m^c = \arg\max_m p_{cm}(c|m)$$
$$CI(p_{cm}, p_{mc}) = \frac{1}{|\mathcal{C}|} \sum_c p_{mc}(m^c|c) \cdot p_{cm}(c|m^c)$$

where $\mathcal{C}$ is the space of all concepts, $p_{cm}(c|m)$ is the conditional probability of a concept given a message, and similarly for $p_{mc}(m|c)$. This quantity relies on having a well-defined notion of 'concept' (in Bogin et al. (2018), this corresponds e.g. to the number and colour of objects), and a way of estimating $p_{cm}$ and $p_{mc}$ (Bogin et al. (2018) use an IBM model 1 (Brown et al., 1993)).

Context independence captures the same intuition as speaker consistency: if a speaker is consistently using a specific word to refer to a specific concept, then communication has most likely emerged. Thus, it is also a measure of positive signaling. The difference is that CI emphasizes that a single symbol should represent a certain concept or input, whereas a high speaker consistency can be obtained using a set of symbols for each input, so long as this set of symbols is (roughly) disjoint for different inputs.

**Entropy of message distribution** Another common practice is to measure the perplexity or entropy ($H(\boldsymbol{\pi}^c)$) of the distribution over messages for different inputs (Havrylov and Titov, 2017; Choi et al., 2018; Evtimova et al., 2017). Different papers interpret the meaning of this quantity differently, but generally if the message distribution has low entropy for a given input, then the speaker is consistently using the same message to describe that input. This differs from the speaker consistency metric, as it does not measure whether or not the speaker is using a *different* message for each input. The entropy actually measures neither positive signaling (an agent that always outputs the same message, independent of observation and action, will have low entropy), nor positive listening (it does not take into account the other agent's response to the messages), which gives it questionable utility as a metric.

**Instantaneous coordination** Another metric considered in Jaques et al. (2018a) is instantaneous coordination (IC). IC uses the same formula as for SC (Eq. 6.1), except it calculates the mutual information between one agent's message and the *other* agent's action, again by averaging (message, action) co-occurrences over episodes. IC is a measure of positive listening; however, because of the way $p(a, m)$ is calculated, it only detects cases where a message from one agent changes the other agent's action regardless of context.

To illustrate why this is undesirable, we describe a simple matrix communication game (MCG) example (see Section 6.3.1) where IC would fail to detect positive listening. Let us draw the entries of the payoff matrix $R$ from a Gaussian at each timestep, and fix agent 1 to always truthfully signal the action it will take. In this case, the optimal policy for agent 2 is to select the best response to agent 1's action, which it knows exactly. Clearly this policy exhibits significant positive listening (it changes its action depending on the message from agent 1 and the input). However, if $R$ is drawn randomly, when averaged across inputs agent 2 will take action 1 and 2 equally. The IC will then be 0 in expectation, since it calculates $p(a, m)$ by averaging over episodes and does not condition on $R$.

**Algorithm 1:** One-step causal influence of communication.

**Data:** Agent policy $\boldsymbol{\pi}_1$, other agent policy $\boldsymbol{\pi}_2$, possible messages $\bar{m} = (m_0, ..., m_{M-1})$, number of test games $T$.
CIC $= 0$
**for** $i \in \{0, ..., T-1\}$ **do**
  Generate new state $S$, observations $O$.
  `// Intervene by changing message` $m_j$ `(replace` $p(m_j)$ `with` $1/|\bar{m}| = 1/M$`)`
  **for** $j \in \{0, ..., M-1\}$ **do**
    $p(m_j) \leftarrow \boldsymbol{\pi}_2(m_j|o_2), \quad p(a|m_j) \leftarrow \boldsymbol{\pi}_1(a|o_1, m_j)$
    $p(a, m_j) = p(a|m_j)/M$
    $p(a) = \sum_{m \in \mathcal{A}^m} p(a, m)$ CIC $+= 1/T \cdot \sum_{a \in \mathcal{A}^e} p(a, m_j) \log \frac{p(a, m_j)}{p(a)p(m_j)}$
  **end**
**end**

### 6.2.3 Causal influence of communication

We now propose a metric that more directly measures positive listening, which we call the causal influence of communication (CIC), following concurrent work in Jaques et al. (2018a). CIC measures the causal effect that one agent's message has on another agent's behaviour. In the simplest case, we can measure the effect that an agent's message has on the next action of the opponent. We call this one-step causal influence. The one-step CIC is calculated using the mutual information between an agent's message and the other agent's action, similarly as for IC (cf. Eq 6.1), however the probabilities $p(a, m) = \boldsymbol{\pi}_1(a|m)\boldsymbol{\pi}_2(m)$ represent the change in the agent's ($\boldsymbol{\pi}_1$) action probability distribution when intervening to change the message $m$ spoken by the other agent ($\boldsymbol{\pi}_2$). In other words, the probabilities are normalized over each game, rather than across games. We describe in detail how to calculate the one-step CIC in Algorithm 1, and we discuss how CIC might be generalized to the multi-step case in Section 6.5.1. When calculating the CIC, care must be taken that we condition on all the variables that can affect the other agent's action, to avoid back-door paths (Pearl et al., 2016). In our setting this is easy, as our MCGs are not iterated (the state and reward are independent of actions and states at previous timesteps).

## 6.3 Experimental setup

### 6.3.1 Matrix Communication Games

Our work is based on a simple class of games called *matrix games*, where each agent $i$'s reward $r_i^t$ at time step $t$ is determined via lookup into a fixed payoff matrix $R_i^t \in \mathbb{R}^{|\mathcal{A}_1^e| \times \cdots \times |\mathcal{A}_N^e|}$ for each agent, indexed by the agents' actions (i.e. $r_i^t = R_i^t(a_1^t, ... a_N^t)$). We study emergent communication in an augmented version of matrix games which we call *matrix communication games* (MCGs), where agents can send discrete, costless messages over a communication channel before acting. In other words, MCGs are matrix games where $|\mathcal{A}_i^m| > 0$ for some $i \in \{0, ..., N\}$. MCGs are interesting to study because they are the simplest form of game where agents can both communicate with each other, and act in the environment. MCGs can be easily adapted to various settings (cooperative vs. non-cooperative, fully vs. partially observable), are simple and efficient for learning, and the resulting communication policies are generally interpretable. Variants of matrix games with communication have been used to model emergent signaling in the fields of economics, game theory, and evolutionary biology (Farrell and Rabin, 1996; Wagner et al., 2003; Smith, 1991; Huttegger et al., 2010).

In our variant of the game, two agents play an MCG in rounds. A round begins with one agent communicating, and ends when both agents have acted and received a reward, at which point another payoff $R^{t+1} = (R_1^{t+1}, R_2^{t+1})$ is selected. The agents observe both payoff matrices and all messages sent that round. Communication occurs in turns;[2] at each round, one agent is randomly chosen to speak first, and sends a message (a one-hot vector of length $M = |\mathcal{A}^m|$) to the other agent. That agent observes the message from the first agent, and can in turn send its own message. After this exchange, both agents act simultaneously, and receive a reward. We consider the non-iterated case, where the actions and messages from previous rounds are not observed.[3] To simplify learning, we use this reward to update both the agent's action and communication policies for that round.

For the experiments in this chapter,[4] we consider the *general-sum* case, where agents are not strictly cooperative and may have competing objectives (i.e. $R_1 \neq R_2$). Note that this does not mean the agents are strictly competitive either; the agents have some *partial common interest* in the game which permits a degree of coordination. We vary the size of the payoff matrix from 2x2, to 4x4, to 8x8, and we provide a communication channel slightly larger than the number of actions ($M$ = 4, 6, and 10, respectively), so that the agents can in theory learn flexible communication policies.

### 6.3.2 Model and learning algorithm

We train our agents using an adaptation of the REINFORCE algorithm (Williams, 1992). We represent the agents' policies using a two-layer feed-forward network (with parameters $\theta^n$), with separate linear output layers for the action $a$ ($\theta^a$),

[2]The exact form of communication is not important for our results; we have observed similar behaviour when agents speak simultaneously, or if only one agent speaks.

[3]In Section 6.4.3 we show that the iterated case where agents (trained using the A2C algorithm (Mnih et al., 2016)) have a memory of previous interactions produces similar behaviour.

[4]Code available at `github.com/facebookresearch/measuring-emergent-comm`.

communication $c$ ($\theta^c$), and learned baseline $V$ ($\theta^v$). Let $\theta^{\mathbf{a}} = \{\theta^a, \theta^n\}$ be all the parameters of the action network giving rise to an action policy $\boldsymbol{\pi}_{\theta^{\mathbf{a}}}$, and similarly for $\theta^{\mathbf{c}}$ and $\theta^{\mathbf{v}}$ (with outputs $\boldsymbol{\pi}_{\theta^{\mathbf{c}}}$ and $V$). The objective for each agent can be written as:

$$J(\theta) = J_{pol}(\theta^{\mathbf{a}}) + \lambda_c J_{pol}(\theta^{\mathbf{c}}) + \lambda_{ent}(J_{ent}(\theta^{\mathbf{a}}) + J_{ent}(\theta^{\mathbf{c}})) + \lambda_v J_v(\theta^{\mathbf{v}}), \tag{6.2}$$

where $\theta = \{\theta^n, \theta^a, \theta^c, \theta^v\}$. This can be broken down as follows: $J_{pol}(\theta^{\mathbf{a}}) = \mathbb{E}_{\pi_{\theta^{\mathbf{a}}}}[-\log \pi_{\theta^{\mathbf{a}}}(a|o) \cdot (r - V(o))]$ is the normal REINFORCE update for the action probabilities, where $o \in \mathcal{O}$ is the observation and $V(o)$ is a learned value function used as a baseline to reduce the variance. $J_{pol}(\theta^{\mathbf{c}})$ takes the same form, except $V$ only uses information available to the agent at the time it sends its message, which may not include the message of the other agent (there is no separate value function for the communication output). This value function is updated using the squared loss $J_v(\theta^{\mathbf{v}}) = (r - V(o))^2$. The entropy bonus $J_{ent}(\theta)$ gives a small reward based on the entropy of the agent's policies, as is common in the RL literature to encourage exploration (Williams and Peng, 1991; Schulman et al., 2015; Christiano et al., 2017). Each $\lambda_i$ is a real-valued coefficient that we tune separately.

## 6.4 Results

We now train deep RL agents to play MCGs. We find that even though our agents show strong indicators of communicating according to speaker consistency and qualitative analysis, this does not mean that the communication is useful. In fact, we show that this 'communication' occurs even if the messages are scrambled (replaced by a random message) before being observed.

### 6.4.1 Positive signaling with random payoffs

**Fixed $R$ setting** It is known that humans are able to use communication to obtain a higher reward in general-sum matrix games, such as the Prisoner's Dilemma (Sally, 1995). To our knowledge, whether RL agents can learn to communicate for various MCGs remains an open question. So, we first conduct an experiment where we train two REINFORCE agents to play MCGs where the

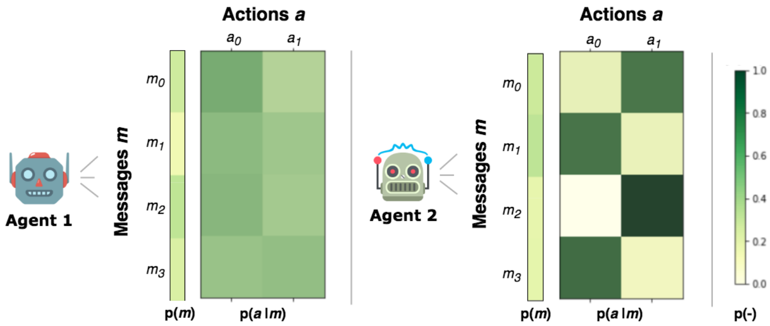



FIGURE 6.1: Visualization of example learned policies for two agents playing 2x2 MCGs ($M = 4$), averaged over 1000 test games. Agent 2 learns to clearly signal its next action.

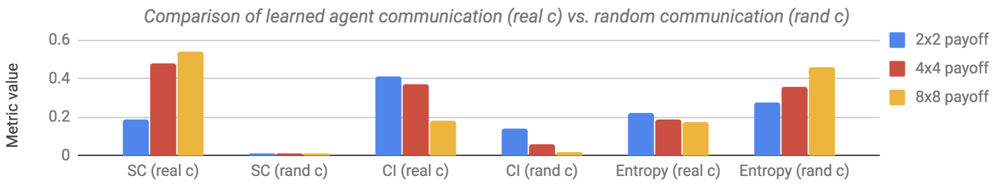



FIGURE 6.2: When comparing the speaker consistency (SC), context independence (CI), and entropy of the message policy (divided by 5) between agents who learned to signal (real $c$) and agents who message randomly (rand $c$), it appears as though the agents are communicating (note we expect a lower entropy for a communicating policy).

payoff $R^t$ is fixed for every timestep $t$. We vary the size of the payoff matrix and communication channel as described in Section 6.3.1. We find experimentally that, for every payoff matrix we tried, the agents don't learn to communicate. Instead, when there is partial common interest, agents collapse to executing a single action (even with a well-tuned $\lambda_{ent}$), and in zero-sum games they cycle between actions without communicating. Intuitively, this makes sense; the main utility in an agent learning to communicate in this setting is in reducing the other agent's uncertainty about their action. Evidently, when always playing the same payoff it is easier for these naïve agents to adapt to the actions of the opponent directly, rather than learning a communication protocol.

**Randomized $R$ setting** One way we can increase the uncertainty each agent has about the other's action (with the hope of producing emergent communication) is by *randomizing the payoffs* at each round. In our next experiment, we train two agents on an MCG where, at every round, each entry of the payoff matrix $R_i^t$ is drawn from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, with $\mu = 0$ and $\sigma^2 = 3$.

| Payoff Size | min(**CIC**) | Average CIC | % games with CIC < $1.02 \cdot \min(\mathbf{CIC})$ |
|---|---|---|---|
| 2x2 | 1.386 | $1.408 \pm 0.002$ | $89.3 \pm 0.6\%$ |
| 4x4 | 1.792 | $1.797 \pm 0.001$ | $97.9 \pm 0.4\%$ |
| 8x8 | 2.303 | $2.303 \pm 0.001$ | $99.9 \pm 0.1\%$ |

TABLE 6.1: Causal influence values for various matrix sizes, calculated over 1000 test games. In all cases, the average CIC is very close to the minimum CIC (when changing the message has no effect on the action distribution).

| Experiments | 2x2 payoff | 4x4 payoff | 8x8 payoff |
|---|---|---|---|
| Scrambled $c$ | $0.198 \pm 0.038$ | $0.487 \pm 0.051$ | $0.597 \pm 0.091$ |
| Separate $c$ net | $0.028 \pm 0.002$ | $0.124 \pm 0.011$ | $0.020 \pm 0.019$ |
| No $c$ training | $0.171 \pm 0.033$ | $0.428 \pm 0.025$ | $0.686 \pm 0.049$ |
| Default | $0.202 \pm 0.040$ | $0.510 \pm 0.094$ | $0.541 \pm 0.090$ |

TABLE 6.2: SC values for the randomized $R$ setting. 'Scrambled $c$' is when the messages are replaced by a random message before being observed, 'Separate $c$ net' is when the action and message networks have no shared parameters, and 'No $c$ training' is when $\lambda_c = 0$.

As shown in Figure 6.2, in this randomized setting speaker consistency emerges. The SC for the generated messages is significantly greater than for random messages, which rules out the possibility that the agents' messages are simply acting as a random public signal that the agents are using to condition their actions (as would be the case in a correlated equilibrium (Aumann, 1974)). The CI is also higher for the generated messages, and the entropy of the message distribution is lower, both indicating that communication has emerged. We also examine the policies qualitatively in Figure 6.1, and find that one agent clearly learns to signal its intended action. One plausible explanation for this phenomenon is that the agents have uncertainty about the intended action of the other agent, and receiving a high reward in these games requires agents to predict the intended action of the opponent. An agent can do this in two ways: first by using the opponent's payoff matrix (and its own) to determine what action they are likely to take to achieve a high reward, and second using the opponent's message to predict their most likely action. When the payoff matrix is changing at each time step, predicting the action of the opponent conditioned solely on the payoff matrix becomes significantly harder. Thus, the agents seem to learn to leverage the communication channel to provide additional information to the other agent, which is beneficial for both agents when the game has partial common interest.

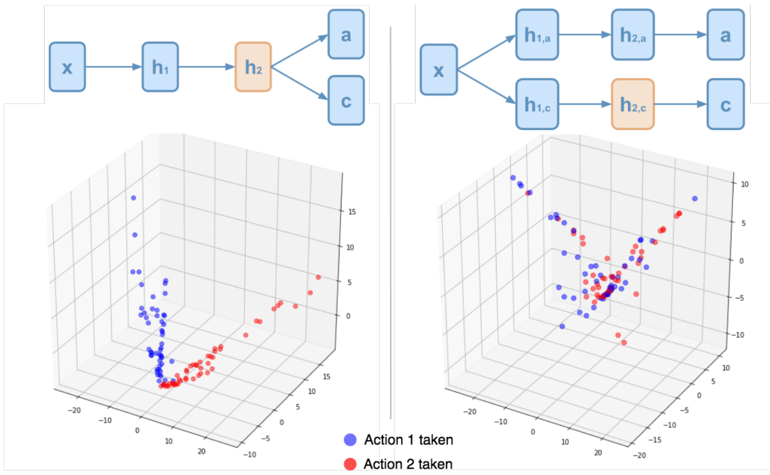



FIGURE 6.3: Activations of the last layer of the policy network for both the standard architecture (left), and when using a separate network for communication (right). Calculated on 100 random 2x2 games, and reduced to 3D using PCA.

### 6.4.2 Positive signaling $\not\Rightarrow$ positive listening

It turns out that, in the randomized $R$ setting, the communication has very little effect on the agents' behaviour. We show this in two ways. First, we examine the trained policies directly to see how altering the message in various games changes the resulting action selection, using the CIC metric. We calculate the CIC over 1000 test games, and show the results in Table 6.1. We find that, for the vast majority of games, the message sent by an agent has no effect on the action of the opponent. Thus, communication is not having a significant effect on the training process.

Second, we conduct an experiment where we train the agents in the randomized $R$ setting described above, except we *scramble* the messages received by both agents. That is, each agent outputs some message $c_i$, but we have both agents observe a different communication $c_i'$, which has no relation to $c_i$ and is drawn uniformly at random from the set of messages. Thus, there is no possibility for the agents' messages to impact the learning whatsoever. However, as shown in Table 6.2, the SC between the agents' action and sent message (*not* the random replacement message) is still positive and indistinguishable from the SC in the regular MCG set-up. This is convincing evidence that the correlation between actions and communications does not emerge because the message are useful, but rather as a byproduct of optimization.

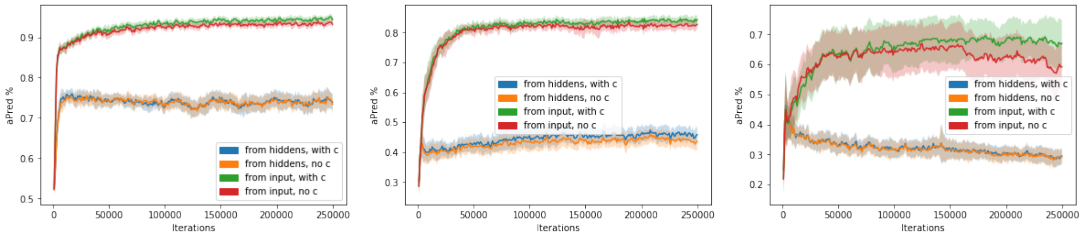



FIGURE 6.4: Accuracy in predicting the opponent's action using action classifier probes, for 2x2 payoffs (left), 4x4 payoffs (center), and 8x8 payoffs (right). 'no c' indicates that the other agent's communication was not used to predict their action, and 'from input' indicates that a separate network was trained to predict the action (rather than using a linear model on top of the last hidden layer of the policy network).

**Why is the SC positive?** If the emergent communication is not useful at all, why is the SC positive? To help us understand what the policies are learning, we train agents according to the randomized $R$ setting in Section 6.4.1, and we plot the activations of the last hidden layer (values in the policy network after the last activation function) for 100 inputs using principal component analysis (PCA) (Pearson, 1901) in Figure 6.3. This shows us our policy network's learned representations of the data (LeCun et al., 2015). When using shared features (left), the network learns to separate the inputs based on which action the agent takes. This makes sense: in order to take the appropriate action for a given input, the representations need to be linearly separable at the last layer of the network, since the action output layer has a linear decision boundary (Goodfellow et al., 2014). This separation of representations does not occur in the last layer of a separate communication network, which provides further evidence for this explanation.

This separation of representations makes it easy for a relationship to emerge between an agent's actions and messages, even if the parameters of the communication head are completely random; since the communication output layer also has a linear decision boundary, it is likely to separate based on intended actions to some degree. Indeed, we find that SC emerges using our architecture when the communication parameters are not trained (Table 6.2). Further, when we re-train the agents using completely separate networks for the actions and messages, we find that the SC completely disappears (see Table 6.2), showing that it was indeed our choice of architecture that resulted in the emergent signaling behaviour.

| Experiments | 2x2 payoff | 4x4 payoff | 8x8 payoff |
|---|---|---|---|
| Scrambled $c$ | $0.195 \pm 0.059$ | $0.127 \pm 0.065$ | $0.208 \pm 0.067$ |
| Separate $c$ net | $0.016\pm 0.019$ | $0.019 \pm 0.011$ | $0.000 \pm 0.000$ |
| Default | $0.169 \pm 0.015$ | $0.130 \pm 0.055$ | $0.269 \pm 0.076$ |

TABLE 6.3: Speaker consistency values for different experiments in the iterated MCG case with A2C agents (Mnih et al., 2016).

**Why aren't the agents using the messages?** We would imagine that, even if the correlation between actions and messages emerged accidentally, that this might still be useful for the agents in selecting their action. After all, isn't more information about the opponent strictly better? To answer this question, we use a set of *action classifier probes*, related to the linear classifier probes for understanding hidden layers of neural networks proposed in (Alain and Bengio, 2016). Specifically, in the randomized $R$ setting, we train a neural network 'probe' to predict the action of the opponent in the current round, based on either part of the input (using 2-layer MLP probes) or part of the last hidden layer of the policy network (using linear probes).

The results are shown in Figure 6.4. We observe that removing the opponent's message from the input does not significantly reduce the accuracy in predicting the opponent's action, both when using a probe directly from the input and a probe from the last layer of the network. This suggests that the information being provided by the messages is redundant when compared to the information provided by the payoff matrix itself.

### 6.4.3 Results in the iterated MCG setting

The results in this chapter are not limited to the non-iterated case. To show this, we run experiments on an iterated version of our environment, using the A2C algorithm (Mnih et al., 2016). We keep the policy architectures the same, except we give each agent a memory of the previous 5 rounds (the actions and the messages of both agents), which is concatenated to the input at each round. We increase the discount factor $\gamma$ to 0.9. Changing our REINFORCE algorithm (Eq. 6.2) to A2C requires changing the way $Q^{\pi}$ is estimated; instead of using the next reward, we use the $n$-step return (sum of $n$ next rewards), with $n = 5$ (Mnih et al., 2016). The results are shown in Table 6.3. We can see that the same general trend is present: there is positive SC in the randomized $R$ setting, even

| Metric | Positive signaling Sufficient? | Positive signaling Necessary? | Positive listening Sufficient? | Positive listening Necessary? | Remarks |
|---|---|---|---|---|---|
| SC | ✓ | ✗ | ✗* | ✗ | Not necessary for positive signaling, as there may be no relationship between the message and subsequent action (can communicate previous observations or actions). |
| CI | ✓ | ✗ | ✗* | ✗ | More restrictive than SC, punishes formation of synonyms. |
| $H(\boldsymbol{\pi}^c)$ | ✗ | ✗ | ✗* | ✗ | Useful for diagnosing policy behaviour, but not as a metric. |
| IC | ✗ | ✗ | ✓ | ✗* | Since it is not state-dependent, IC can miss many positive listening relationships. |
| MIN | ✗ | ✗ | ✗* | ✓ | If $\|\|W_m^1\|\| = 0$, then no positive listening is present. |
| $\Delta r$ | ✓ | ✗ | ✓ | ✗ | Should always be measured, strong indicator that communication is present. But less applicable when comparing the communicative behaviour of two policies |
| Qual. | ✓ | ✓ | ✗* | ✗ | Very useful for understanding agent behaviour. Can come in many forms. |
| CIC | ✗ | ✗* | ✓ | ✓ | Should always be used to determine effect communication has on agent behaviour. |

TABLE 6.4: Summary of the metrics analyzed in this chapter ($\Delta r$ = increase in reward when adding a communication channel, Qual. = qualitative analysis of messages, and MIN = message input norm, detailed in Section 6.5.2). Asterisks (*) mark relationships we have shown experimentally (Sections 6.4 and 6.5.2) or via counterexample (Section 6.2). See text for a more detailed explanation.

when scrambling the messages $c$; however, the SC disappears when a different network is used to produce the messages.

## 6.5 Discussion

### 6.5.1 How general is this analysis?

**Positive signaling without positive listening** An important question is whether the behaviours observed in this chapter are specific to training our policy architecture on MCGs, and whether any of these insights can be applied in other emergent communication settings. We conjecture that this could indeed happen whenever the agent's architecture uses shared feature learning layers between the action and communication outputs; policies will always learn representations that separate the inputs based on the desired action, and this may lead to spurious correlations between an agent's messages and actions. Since sharing features is quite common in RL (e.g. (Jaderberg et al., 2016)), it is possible that this becomes an occasional occurrence in emergent communication work.

However, our claim is not that this specific failure case will be frequently observed; rather, our goal is to highlight the importance of understanding what our metrics are measuring, and encourage emergent communication researchers to explore quantitative metrics for measuring the impact that the agents' communication is having on their behaviour (i.e. positive listening).

**Scaling causal influence** In this chapter, we focused on the one-step approximation to CIC, which only calculates the effect of an agent's message (which consists of one symbol) on the other agent's next action. While this is sufficient for the non-iterated MCG setting, as we move to more complex environments we will need to measure the effect of compositional messages on the long-term behaviour of all other agents in the environment. In this case, calculating CIC naïvely using Algorithm 1 will be computationally expensive. However, we can make this efficient via *sampling*; rather than iterating over all possible messages and all agents, the CIC could be calculated by iterating over a small set of messages sampled from the agent's communication policy, and evaluating the change in behaviour over finite time horizon for agents in some neighbourhood of the speaking agent. We leave a detailed exploration of this direction to future work.

### 6.5.2 Recommendations

Here we provide recommendations as to when the metrics presented in Section 6.2 might be used to either detect whether communication is emerging, or measure the difference in communication quality between algorithms. We also propose some other tests that could be used for this purpose. We summarize our insights in Table 6.4. In general, no single metric or test will tell the whole story, and we advise using several of them to illuminate the agents' behaviour.

**Detecting emergent communication** If our goal is to detect whether communication is emerging at all, showing that adding a communication channel to a given algorithm leads to improved reward is a sufficient indicator. However, it may not be necessary; agents may obtain a similar reward by coordinating via learned convention (Lerer and Peysakhovich, 2018), rather than communication. In other words, communication may act as an alternate pathway for optimization. Detecting useful communication is not as simple as testing if removing the communication channel at test time leads to a decrease in reward; neural networks are notoriously sensitive to their input distribution (Szegedy et al., 2013), and a change in this distribution (e.g. setting the messages to 0) may cause them to fail, even if the messages contain no useful information. We recommend instead using CIC and other causal metrics, discussed below.

The variant of SC explored here only measures the one-step relationship between an agent's message and subsequent action. In general, communication could influence the actions of agents further in the future than a single time step, and the language used by the agents may be compositional and temporally extended. This should be taken into account as we move towards more complex environments. In general, we acknowledge that speaker consistency is useful from the perspective of detecting positive signaling, but we reiterate that the observed relationships may be spurious.

There are other aspects of the environment that an agent could learn to signal about: an agent might send a message to get another agent to perform an action, to share an observation it has made, or to reveal the sequence of actions it has taken in the past. New metrics need to be developed to evaluate these possibilities. We recommend researchers evaluate the quantities that are relevant for their environment or task. The crucial point is that, if these quantities are measured by observing the agent's behaviour without causal intervention, detecting positive listening is difficult, as evidenced by our experiments in Section 6.4. If a relationship is observed between an agent's messages and some quantity in the environment, we recommend researchers *investigate the causal relationship* between these variables, by intervening to change in turn the environmental quantity and the agent's message, and observing the impact on the other quantity over a number of episodes. This has also been proposed in (Everitt et al., 2019), where the authors propose to use 'causal influence diagrams' to determine which aspects of the environment the agents have incentive to observe and control.

One way to tell conclusively that there is no positive listening is to look at the weight matrix of the first layer of the policy network ($W^1$), specifically the part that comes from the message inputs of the other agents ($W^1_m$). If the norm of this part of the weight matrix (the *message input norm*, MIN) is 0, then clearly no positive listening is present, as the messages from the other agents cannot affect a given agent's behaviour. However, just because $||W_m^1|| > 0$, does not mean positive listening is present; in our experiments on MCGs, we found that this norm was of similar magnitude to the norm of the weights from the payoff matrix.

**Measuring improvement in communication** How should we judge the quality of a learned communication protocol in a multi-agent environment? Of course, this depends on the environment and the objectives of the researcher. Often, researchers may want to show that their algorithm exhibits a new kind of communication (e.g. verbal agreements, compositional language, or deception). In these cases, it makes sense to use metrics targeted at measuring the phenomenon in question. If the goal is to develop compositional communication, as has been the case for several recent emergent communication papers (Mordatch and Abbeel, 2017; Havrylov and Titov, 2017; Bogin et al., 2018), then it is perhaps sufficient to evaluate using metrics designed to measure compositionality, such as context independence (Bogin et al., 2018). These metrics will have to be developed on a case-by-case basis, depending on the type of communication under investigation.

There may also be cases where we simply want to show that the learned communication protocol for a proposed algorithm has a larger effect on agent behaviour than for previous algorithms. Here, a variant of CIC should be used that measures the impact of communication on the long-term behaviour of the other agents. Another test one could run is measuring the difference in reward for each algorithm with and without communication. This should be done by training each algorithm from scratch with and without communication, rather than removing the communication channel at test time, to avoid the problems of distributional shift mentioned earlier in this section. Of course, since these metrics may be exploitable, it is important to benchmark against a range of metrics and tests to avoid overfitting.

# Chapter 7

# Bridging the gap from emergent communication to natural language

## 7.1 Motivation

A promising approach for training agents to interact with humans using natural language is to have a "human in the loop", meaning we collect problem-specific data from humans interacting directly with our agents for learning. However, human-in-the-loop data is expensive and time-consuming to obtain as it requires continuously collecting human data as the agent's policy improves, and recent work suggests that current machine learning methods (e.g. from deep reinforcement learning) are too data-inefficient to be trained in this way from scratch (Chevalier-Boisvert et al., 2018). Thus, an important open problem is: how can we make human-in-the-loop training as data efficient as possible?

To maximize data efficiency, it is important to fully leverage all available training signals. In this chapter, we study two categories of such training methods: imitating human data via supervised learning, and self-play to maximize reward in a multi-agent environment, both of which provide rich signals for endowing agents with language-using capabilities. However, these are potentially competing objectives, as maximizing environmental reward can lead to the resulting communication protocol drifting from natural language (Lewis et al., 2017; Lee et al., 2019). The crucial question, then, is how do we best combine self-play and supervised updates?

One could equally frame the motivation for this chapter through the lens of emergent communication. As discussed previously, one of the main goals of emergent communication research — and indeed, the goal of studying emergent communication in this thesis — is as a stepping stone to developing agents that can communicate with humans in natural language. However, since learned emergent protocols rarely bear much resemblance to natural language (Lewis et al., 2017; Lee et al., 2019), one of the biggest obstacles in emergent communication research is: how can we bridge the gap between emergent communication and natural language? Before the NeurIPS 2019 Emergent Communication Workshop,[1] this question had received surprisingly little attention from the emergent communication literature, where the question is generally left for future work (Mordatch and Abbeel, 2017; Cao et al., 2018). There was an unwritten consensus among many in the community that researchers would train agents in more and more complex environments to obtain increasingly complex languages, and eventually fine-tune agents on natural language data to get them to communicate with humans (Mordatch and Abbeel, 2017) (or instead, to learn a translation model between artificial and human languages as in Andreas et al. (2017)). Indeed, this was the opinion of the author of this thesis when writing the papers that constituted Chapters 5 and 6, and when starting to write the paper upon which this chapter is based on.

Our goal in this chapter is to investigate algorithms for combining supervised learning with self-play — which we call supervised self-play (S2P) algorithms — using two classic emergent communication tasks: a Lewis signaling game with symbolic inputs, and a more complicated image-based referential game with natural language descriptions. Our first finding is that supervised learning followed by self-play outperforms emergent communication with supervised fine-tuning in these environments, and we provide three reasons for why this is the case. We then empirically investigate several supervised-first S2P methods in our environments. Existing approaches in this area have used various ad-hoc schedules for alternating between the two kinds of updates (Lazaridou et al., 2016), but to our knowledge there has been no systematic study that has compared these approaches. Lastly, we propose the use of population-based methods for S2P, and find that it leads to improved performance in the more challenging image-based referential game. Our findings highlight the need for

[1] `https://sites.google.com/view/emecom2019`

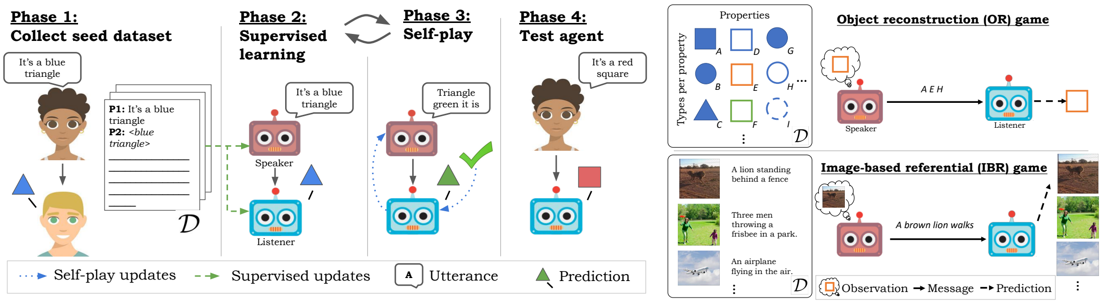
further work in combining supervised learning and self-play to develop more sample-efficient language learners.

## 7.2 Related work

In the past few years, there has been a renewed interest in the field of emergent communication (Sukhbaatar et al., 2016; Foerster et al., 2016; Lazaridou et al., 2016; Havrylov and Titov, 2017) culminating in 3 NeurIPS workshops. Empirical studies have showed that agents can autonomously evolve a communication protocol using discrete symbols when deployed in a multi-agent environment which helps them to play a cooperative or competitive game (Singh et al., 2018; Cao et al., 2018; Choi et al., 2018; Evtimova et al., 2017).

While the idea of promoting coordination among agents through communication sounds promising, recent experiments (Lowe et al., 2019; Chaabouni et al., 2019; Lazaridou et al., 2018; Kottur et al., 2017; Jaques et al., 2018b) have emphasized the difficulty in learning meaningful emergent communication protocols even with centralized training.

Apart from the above advances in emergent communication, there has been a long outstanding goal of learning intelligent conversational agents to be able to interact with humans. This involves training the artificial agents in a way so that they achieve high scores while solving the task and their language is interpretable by humans or close to natural language. Recent works also add another axis orthogonal to communication where the agent also takes a discrete action in an interactive environment (de Vries et al., 2018; Mul et al., 2019). Lewis et al. (2017) introduced a negotiation task which involves learning linguistic and reasoning skills. They train models imitating human utterances using supervised learning and found that the model generated human-like captions but were poor negotiators. So they perform self-play with these pretrained agents in an interleaved manner and found that the performance improved drastically while avoiding language drift. Lee et al. (2019) also propose using an auxiliary task for grounding the communication to counter language drift. They use visual grounding to learn the semantics of the language while still generating messages that are close to English.




(A) (B)

FIGURE 7.1: (a) Diagram of the supervised self-play (S2P) procedure (phases 1-3) and the testing procedure considered in this work (phase 4). (b) The environments considered in this chapter (Sec. 7.4).

A recent trend on using population based training for multi-agent communication is a promising avenue for research using inspirations from language evolution literature (Smith et al., 2003; Kirby, 2014; Raviv and Arnon, 2018). Cultural transmission is one such technique which focuses on the structure and compression of languages, since a language must be used and learned by all individuals of the culture in which it resides and at the same time be suitable for a variety of tasks. Graesser et al. (2019) shows the emergence of linguistic phenomena when a pool of agents contact each other giving rise to novel creole languages. (Li and Bowling, 2019; Cogswell et al., 2019; Tieleman et al., 2019) have also tried different ways of imposing cultural pressures on agents, by simulating a large population of them and pairing agents to solve a cooperative game with communication. They train the agent against a sampled *generation* of agents where the *generation* correspond to the different languages of the same agent at different times in the history.

Our work is inspired from these works where we aim to formalize the recent advancements in using self-play in dialog modeling, through the lens of emergent communication.

## 7.3 Methods

### 7.3.1 Problem definition

Our setting is similar to the set-up described in Section 2.6.2: our agents are embedded in a multi-agent environment with $N$ agents where they receive observations $o \in O$ (which are functions of a hidden state $S$) and perform actions $a \in A$ . Some actions $A_L \subset A$ involve sending a message $m \in A_L$ over a *discrete*, *costless* communication channel (i.e. a *cheap talk* channel (Farrell and Rabin, 1996)). The agents are rewarded with a reward $r \in R$ for their performance in the environment. We assume throughout that the environment is cooperative and thus the agents are trained to maximize the sum of rewards $R = \sum_{t=1:T} \sum_{i=1:N} r_{i,t}$ across both agents. This can be thought of as a cooperative partially-observable Markov game (Littman, 1994).

We define a target language $L^* \in \mathcal{L}$, usually corresponding to natural language, that we want our agents to learn (we further assume $L^*$ can be used to achieve high task reward). In this chapter, we consider a language $L \in \mathcal{L}$ to be simply a set of valid messages $A_L$ and a mapping between observations and messages in the environment, $L : O \times A_L \mapsto [0, 1]$. For example, in an English image-based referential game (Section 7.4) this corresponds to the mapping between images and image descriptions in English. We are given a dataset $\mathcal{D}$ consisting of $|\mathcal{D}|$ (observation, action) pairs, corresponding to $N_e$ 'experts' (for us, $N_e = 2$ playing the game using the target language $L^*$. Our goal is to train agents to achieve a high reward in the game while speaking language $L^*$ with an 'expert'. Specifically, we want our agents to *generalize* and to perform well on examples that are not contained in $\mathcal{D}$.

To summarize, we want agents that can perform well on a collaborative task with English-speaking humans, and we can train them using a supervised dataset $\mathcal{D}$ and via self-play.

### 7.3.2 Supervised Self-Play (S2P)

In recent years, there have been several approaches to language learning that have combined supervised or imitation learning with self-play. In this chapter, we propose an umbrella term for these algorithms called *supervised self-play* (S2P). S2P requires two things: (1) a multi-agent environment where at least one agent can send messages over a dedicated communication channel, along with a reward function that measures how well the agents are doing at some task; and (2) a supervised dataset $\mathcal{D}$ of experts acting and speaking language $L^*$ in the environment (such that they perform well on the task). Given these ingredients, we define S2P below (see Figure 7.2).

**Definition 7.1. Supervised self-play (S2P).** *Supervised self-play is a class of language learning algorithms that combines: (1) self-play updates in a multi-agent language environment, and (2) supervised updates on an expert dataset $\mathcal{D}$.*

S2P algorithms can differ in how they combine self-play and supervised learning updates on $\mathcal{D}$. When supervised learning is performed before self-play, we refer to the dataset $\mathcal{D}$ as the *seed data*.

Why might we want to train our agents via self-play? Won't their language diverge from $L^*$? One way to intuitively understand why S2P is beneficial is to think in terms of constraints. In our set-up, there are two known constraints on the target language $L^*$: (1) it is consistent with the samples from the supervised dataset $\mathcal{D}$, and (2) $L^*$ can be used to obtain a high reward in the environment. Thus, finding $L^*$ can be loosely viewed as a constrained optimization problem, and enforcing both constraints should clearly lead to better performance.

### 7.3.3 Algorithms for S2P

Here we describe several methods for S2P training. Our goal is not to exhaustively enumerate all possible optimization strategies, but rather provide a categorization of some well-known ways to combine self-play and supervised learning. To help describe these methods, we further split the seed dataset $\mathcal{D}$ into $\mathcal{D}_{train}$, which is used for training, and $\mathcal{D}_{val}$ which is used for early-stopping. We also visualize the schedules in Figure 7.2.

**Emergent communication with supervised fine-tuning (`sp2sup`):** We first perform self-play updates until the learning converges on the task performance. It is then followed by supervised updates on $\mathcal{D}_{train}$ until the listener performance converges on the dataset $\mathcal{D}_{val}$.

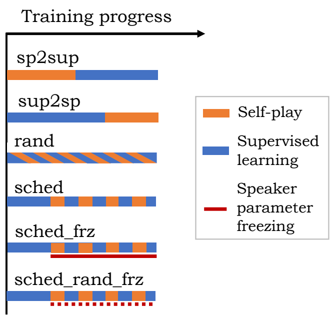



FIGURE 7.2: A visual representation of the different S2P methods.

**Supervised learning with self-play (`sup2sp`):** This is the complement of the above method which involves doing supervised updates until convergence on $\mathcal{D}_{val}$ followed by self-play updates until convergence on the task performance.

**Random updates (`rand`):** This is the method used in (Lazaridou et al., 2016). At each time step, we sample a bernoulli random variable $z \sim Bernoulli(q)$ where $q$ is fixed. If $z = 1$, we perform one supervised update, else we do one self-play update, and repeat until convergence on $\mathcal{D}_{val}$.

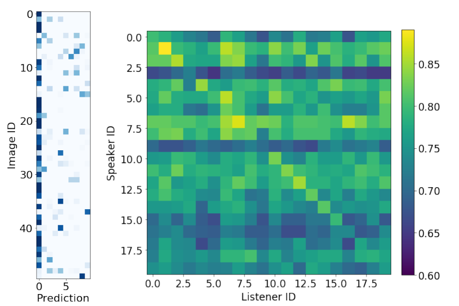
**Scheduled updates (`sched`):** We first pretrain the listener and the speaker until convergence on $\mathcal{D}_{val}$. Then we create a *schedule*, where we perform $l$ self-play updates followed by $m$ supervised updates, and repeat until convergence on the dataset.

**Scheduled updates with speaker freezing (`sched_frz`):** This method is based on the findings of Lewis et al. (2017), who do `sched` S2P while freezing the parameters of the speaker during self-play to reduce the amount of language drift. In our case, we freeze the parameters of the speaker after the initial supervised learning.

**Scheduled updates with random speaker freezing (`sched_rand_frz`):** Experimentally, we noticed that `sched_frz` didn't perform well in self-play. Thus, we introduce a variation, we sample a bernoulli random variable $z \sim Bernoulli(r)$ where $r$ is fixed. If $z = 1$, we freeze the parameters of the speaker during both self-play and supervised learning, else we allow updates to the speaker as well.

### 7.3.4 Population-based S2P (Pop-S2P)

As explained above, the goal of S2P is to produce agents that follow dataset $\mathcal{D}$ while maximizing reward in the environment. However, there are many such policies satisfying these criteria. This results in a large space of possible solutions, that increases as the environment grows more complex (but decreases with increasing $|\mathcal{D}|$). Experimentally, we find that this can result in diverse agent policies. We show this in Figure 7.3 by training 50 randomly initialized agents on the image-based referential game (defined in Sec. 7.4) the agents can often make diverse predictions




(A) (B)

FIGURE 7.3: Results from training 50 S2P agents on the IBR game with $|\mathcal{D}| = 10000$. (a) The agents have a range of predictions on many images. (b) When playing with each other, the agents exhibit uneven performance, indicating policy variability.

for a given image (7.3a) and achieve variable performance when playing with other populations with a slight preference towards their own partner (the diagonal in 7.3b).

Inspired by these findings, we propose to augment S2P by training a *population* of $N$ agents, and subsequently aggregating them back into a single agent (the 'student'). We call this population-based S2P (Pop-S2P). While there are many feasible ways of doing this, in this chapter we train the populations by simply randomizing the initial seed, and we aggregate the populations using a simple form of policy distillation (Rusu et al., 2015). Another simple technique to boost performance is via ensembling where we simply take the majority prediction at each time step.

## 7.4 Environments & implementation details

We consider environments based on classical problems in emergent communication. These environments are cooperative and involve the interaction between a speaker, who makes an observation and sends a message, and a listener, who observes the message and makes a prediction (see Figure 7.1b). Our goal is to train a listener such that it achieves high reward when playing with an expert speaking the target language $L^*$ on inputs unseen during training.[2]

**Environment 1: Object Reconstruction (OR)** Our first game is a Lewis signaling game (Lewis, 1969) and a simpler version of the Task & Talk game from Kottur et al. (2017), with a single turn and a much larger input space. The speaker agent observes an object with a certain set of properties, and must describe the object to the listener using a sequence of words. The listener then attempts to reconstruct the object. More specifically, the input space consists of $p$ *properties* (e.g. shape, color) of $t$ *types* each (e.g. triangle, square). The speaker observes a symbolic representation of the input, consisting of the concatenation of $p = 6$ one-hot vectors, each of length $t = 10$. The number of possible inputs scales as $t^p$. We define the vocabulary size (length of each one-hot vector sent

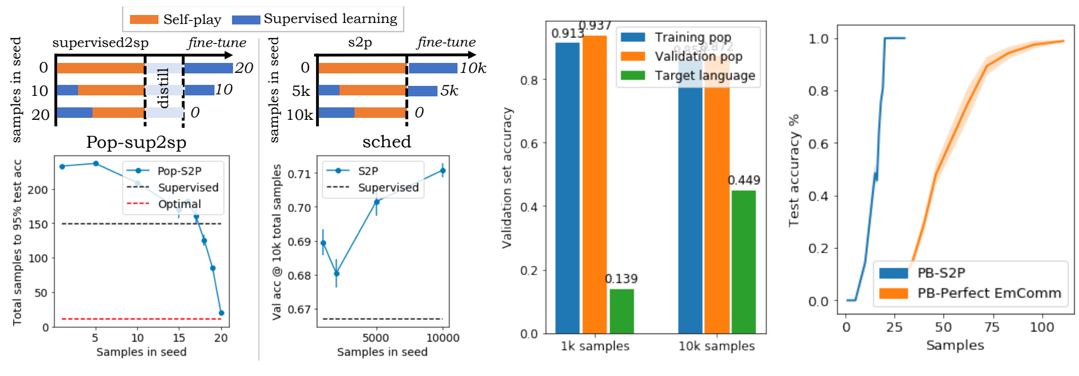
[2]Our approach could equally be used to train a speaker of language $L^*$; we leave this to future work.

from the speaker) as $|V| = 60$, and the number of words (fixed length message) sent to be $T$.

For our target language $L^*$ for this task, we programatically generate a perfectly compositional language, by assigning each object a unique word. In other words, to describe a blue shaded triangle, we create a language where the output description would be "blue, triangle, shaded", in some arbitrary order and without prepositions. By 'unique symbol', we mean that no two concepts are assigned the same word. The speaker and listener policies are parameterized using a 2-layer linear network (results were similar with added non-linearity and significantly worse with 1-layer linear networks) with 200 hidden units. During both supervised learning and self-play, the listener is trained to minimize the cross-entropy loss over property predictions.

**Environment 2: Image-Based Referential game with natural language (IBR)** Our second game is the communication task introduced in the Lee et al. (2017). The speaker observes a target image $d^*$, and must describe the image using a set of words. The listener observes the target image along with $D$ distractor images (for us, $D = 9$), and the message $y_{d^*}$ from the speaker, and is rewarded for correctly selecting the target image. For this game, the target language $L^*$ is English — we obtain English image descriptions using caption data from MS COCO and Flickr30k. We set the vocabulary size $|V| = 100$, and filter out any descriptions that contain more than 30% unknown tokens while keeping the maximum message length $T$ to 15.

Similarly to (Mordatch and Abbeel, 2017; Sukhbaatar et al., 2016), we train our agents end-to-end with backpropagation. Since the speaker sends discrete messages, we use the Straight-Through version of Gumbel-Softmax (Jang et al., 2016; Maddison et al., 2016) to allow gradient flow to the speaker during self-play. The speaker's predictions are trained on the ground truth English captions $m^*$ using the cross entropy loss. The listener is trained using the cross-entropy loss where the logits are the reciprocal of the mean squared error which was found to perform better in Lee et al. (2017). The mean squared error is taken over the listener's image representation $b_{lsn}$ of the distractor (or target) image




(A) (B) (C)

FIGURE 7.4: (a) Left: In the OR game, best performance (number of total samples required to achieve 95% generalization accuracy, lower is better) for S2P is achieved when all of the samples are in the seed. 0 on the x-axis corresponds to the emergent communication + fine-tuning baseline (here Optimal performance is the actual (minimum) number of samples required to solve this optimization problem). Right: This is also the case in the IBR game, where performance is measured by the generalization accuracy using 10k total training samples (higher is better). (c) Adding more samples to initial supervised learning in the IBR game improves agents' generalization to $L^*$. (d) Even when we learn the perfect distribution with emergent communication in the OR game, it still performs worse than S2P.

and the message representation given as input.

$$\mathcal{J}_{\text{spk-supervised}}(d^*) = -\sum_{t=1}^{T} \log p_{spk}(m_t|m_{<t}, d^*)$$

$$\mathcal{J}_{\text{lsn-supervised}}(m^*, d^*, D) = -\sum_{d=1}^{D+1} \log(\texttt{softmax}(1/p_{lsn}(m^*) - b_{\text{lsn}}(d))^2)$$

$$\mathcal{J}_{\text{self-play}}(d^*, D) = -\sum_{d=1}^{D+1} \log(\texttt{softmax}(1/p_{lsn}(y_{d^*}) - b_{\text{lsn}}(d))^2)$$

where $y_{d^*}$ is the concatenation of $T$ one-hot vectors $y^t_{d^*} = \texttt{ST-GumbelSoftmax}(p^t_{spk})$.

We use the same architecture as described in (Lee et al., 2017). The speaker and listener are parameterized by recurrent policies, both using an embedding layer of size 256 followed by an GRU (Cho et al., 2014) of size 512.

## 7.5 Experiment 1: Do supervised learning before self-play

A central question in our work is how to combine supervised and self-play updates for effective pre-training of conversational agents. In this section, we study this question by conducting experiments with two schedules: training with emergent communication followed by supervised learning (`sp2sup`), and training with supervised learning followed by self-play (`sched`). We also interpolate between these two regimes by performing the `sched` on $0 < n < |\mathcal{D}|$ samples, followed by supervised fine-tuning on the remaining $|\mathcal{D}| - n$ samples.

Our first finding is that it is best to use all of your samples for supervised learning before doing self-play. This can be seen in Figure 7.4: when all of the samples are used first for supervised learning, the number of total samples required to solve the OR game drastically, and in the IBR game the accuracy for a fixed number of samples is maximized (Figure 7.4a). While this may seem to be common sense, it in fact runs counter to the prevailing wisdom in much of the emergent communication literature, where languages are emerged from scratch with the ultimate goal of translating them to natural language.

To better understand why it is best to do supervised learning first, we now conduct a set of targeted experiments using the environments from Section 7.4. Results of our experiments suggest three main explanations:

**(1) Emerging a language is hard.** *For many environments, with emergent communication it's often hard to find an equilibrium where the agents meaningfully communicate.* The difficulty of 'emergent language discovery' has been well-known in emergent communication (Lowe et al., 2017c), so we will only briefly discuss it here. In short, to discover a useful communication protocol agents have to coordinate repeatedly over time, which is difficult when agents are randomly initialized, particularly in environments with sparse reward. Compounding the difficulty is that, if neither agent communicates and both agents act optimally given their lack of knowledge, they converge to a Nash equilibrium called the *babbling equilibrium*. This equilibrium must be overcome to learn a useful communication protocol. In S2P, the initial language supervision can help

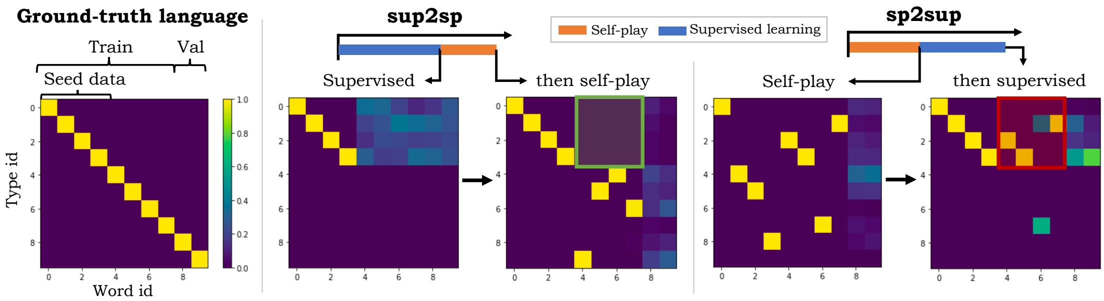



FIGURE 7.5: Results from the OR game with 1 property and 10 types. When the supervised updates are performed first (supervised data available for words 0-3), then the self-play updates make sensible predictions for the unknown words (words 4-7). When the self-play updates are done first, the subsequent supervised updates merely correct the predictions for words 1-4, without enforcing the constraint that each word should result in a separate type to solve the task.

overcome the discovery problem, as it provides an initial policy for how agents could usefully communicate (Lewis et al., 2017).

**(2) Emergent languages are different than natural language.** *Even if one does find an equilibrium where agents communicate and perform well on the task, the distribution of languages they find will usually be very different from natural language.* This is a problem because, if the languages obtained through self-play are sufficiently different from $L^*$, they will not be helpful for learning. This is seen for the OR game in Figure 7.4a, where 17 samples are required in the seed before S2P outperforms the supervised learning baseline. We speculate that this is due to the different pressures exerted during the emergence of artificial languages and human languages.

Thankfully, we can learn languages closer to $L^*$ by simply adding more samples to our initial supervised learning phase. We show this in Figure 7.4b, where we train populations of 50 agents on the IBR game and use Pop-S2P to produce a single distilled agent. With both 1K and 10K initial supervised samples, the distill agent generalizes to agents in the validation set of their population. However, the distilled agent trained with 10000 samples performs significantly better when playing with an expert agent speaking $L^*$, indicating that the training agents from that population speak languages closer to $L^*$.

**(3) Starting with self-play violates constraints.** *Even if you have 'perfect emergent communication' that learns a distribution over languages under which $L^*$ has high probability, current methods of supervised fine-tuning do not properly learn from this distribution.* What if we had all the correct learning pressures, such that we emerged a distribution over languages $\mathcal{L}$ with structure identical to $L^*$? Surprisingly, we find that S2P with all of the samples in the seed performs better than even this optimistic case. We conduct an experiment in the OR game where we programmatically define a distribution over compositional languages $\mathcal{L}_c$, of which our target language $L^*$ is a sample. Each language $L \in \mathcal{L}_c$ has the same structure, the only difference being the mapping between the word IDs and the corresponding type IDs, along with the order of properties in an utterance. As seen in Figure 7.4c, when we train a Pop-S2P agent on 50 of these compositional populations, we still need 3x more samples than S2P.

To understand why this happens, we conduct a case study in an even simpler setting: single-agent S2P in the OR game with $p = 1, t = 10, |V| = 10$. We find that agents trained via emergent communication consistently learn to solve this task. However, as shown in Figure 7.5, when subsequently trained via supervised learning on $\mathcal{D}$ to learn $L^*$, the learned language is no longer coherent (it maps different words to the same type) and doesn't solve the task. On the other hand, agents trained first with supervised learning are able to learn a language that both solves the task and is consistent with $\mathcal{D}$.

Intuitively, what's happening is that the samples in $\mathcal{D}$ are also valid for solving the task, since we assume agents speaking $L^*$ can solve the task. Thus, self-play after supervised learning simply 'fills in the gaps' for examples not in $\mathcal{D}$.[3] Emergent languages that start with self-play, on the other hand, contain input-output mappings that are inconsistent with $L^*$, which must be un-learned during subsequent supervised learning.

In theory, the above issue could be resolved using Pop-S2P; if the distilled agent could use the population of emergent languages to discover structural rules (e.g. discovering that the languages in the OR game in Figure 7.4c are compositional), it could use the samples from $\mathcal{D}$ to refine a posterior distribution over target languages that is consistent with these rules (e.g. learning the distribution of compositional languages consistent with $\mathcal{D}$). Current approaches to supervised

[3] In practice, we find that self-play updates can undo some of the learning of $\mathcal{D}$, which is why we apply an alternating schedule.

fine-tuning in language, though, do not do this (Lazaridou et al., 2016; Lewis et al., 2017). An interesting direction for future work is examining how to apply Bayesian techniques to S2P.

## 7.6 Experiment 2: Population-based approaches improve S2P

In this section, we aim to show that (1) S2P outperforms the supervised learning baseline, and (2) Pop-S2P outperforms S2P. We conduct our experiments in the more complex IBR game, since the agents must communicate in English, and measure performance by calculating the accuracy at different (fixed) numbers of samples. Our baseline is then the performance of a supervised learner on a fixed number of samples.

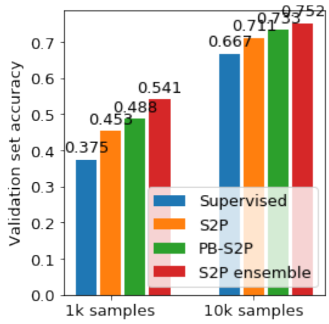
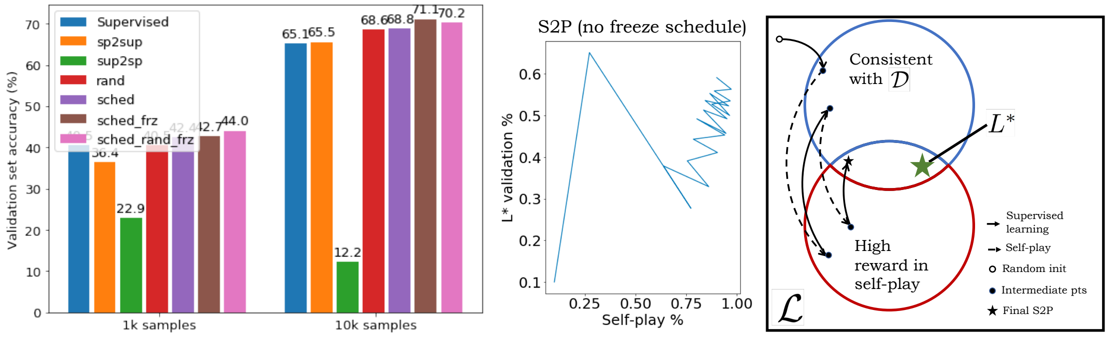



FIGURE 7.6: S2P (`sched`) outperforms the supervised baseline in the IBR game, and is in turn outperformed by Pop-S2P.

We show the results in Figure 7.6. We first note that, when both 1k and 10k samples are used for supervised learning, S2P (`sched`) outperforms the supervised learning baseline. We can also see that the population-based approach outperforms single agent S2P (`sched`) by a significant margin. We also compare our distillation method to an ensembling method that keeps all 50 populations at test time, and find that ensembling performs significantly better, although it is much less efficient. This suggests that there is room to push distilled Pop-S2P to even better performance.

## 7.7 Experiment 3: Examining S2P schedules

In this section, we aim to: (1) evaluate several S2P schedules empirically on the IBR game; and (2) attain a better understanding of S2P through quantitative experiments.




(A) (B)

FIGURE 7.7: (a) Comparing test performances of different S2P methods on the IBR game. For each method, we picked the model that gave the best performance on $\mathcal{D}_{val}$. (b) Left: 2D visualization of S2P (`sched`) performance over the course of training, in terms of performance on $L^*$ (vertical axis) and performance in self-play (horizontal axis). The zig-zag patterns indicates that most self-play updates result in a short-term decrease in target language performance. Right: visualization of the role of the supervised and self-play updates.

**Parameter freezing improves S2P** We show the results comparing different S2P schedules in Figure 7.7a. Interestingly, we find that in this more complex game, the supervised2sp method of S2P performs much worse than the other options. We also see that adding freezing slightly improves the performance on the target language (in the Appendix, we also show that it converges more quickly). We hypothesize that this is because it reduces the language drift that is experienced at each round of self-play updates (Lee et al., 2019).

**Self-play acts as a regularizer** What is the role of self-play in S2P? We can start to decipher this by taking a closer look at the `sched` S2P method. We plot the training performance of this method in Figure 7.7b. Interestingly, we notice from the zig-zag pattern that the validation performance usually goes *down* after every set of self-play updates. However, the overall validation performance goes up after the next round of supervised updates. This is also reflected in the poor performance of the `sup2sp` method in Figure 7.6.

This phenomenon can be explained by framing self-play as a form of regularization: alternating between supervised and self-play updates is a way to satisfy the parallel constraints of 'is consistent with the dataset $\mathcal{D}$' and 'performs well on the task'. We visualize this pictorially in Figure 7.7b: while a set of self-play updates results in poor performance on $\mathcal{D}$, eventually the learned language moves closer to satisfying both constraints.

## 7.8 Discussion

In this work, we investigated the research question of how to combine supervised and self-play updates, with a focus on training agents to learn language. However, this research question is not only important for language learning; it is also a important in equilibrium selection and learning social conventions (Lerer and Peysakhovich, 2019) in general games. For example, in robotics there may be a trade-off between performing a task well (moving an object to a certain place) and having your policy be interpretable by humans (so that they will not stumble over you). Examining how to combine supervised and self-play updates in these settings is an exciting direction for future work.

There are several axes of complexity not addressed in our environments and problem set-up. First, we consider only single-state environments, and agents don't have to make temporally extended decisions. Second, we do not consider pre-training on large text corpora that are separate from the desired task (Radford et al., 2019; Devlin et al., 2018). Third, we limit our exploration of self-play to the multi-agent setting, which is not the case in works such as instruction following (Andreas and Klein, 2015). Finally, we only consider the case where we learn a *listener* that can adapt to human language, whereas in general it is important to learn an agent that can also speak in human language. Introducing these elements may result in additional practical considerations for S2P learning, which we leave for future work. Our goal in this chapter is not to determine the best method of S2P in all of these settings, but rather to inspire others to use the framing of 'supervised self-play algorithms' to make progress on sample efficient human-in-the-loop language learning.

# Chapter 8

# Conclusion

## 8.1 Summary of contributions

This thesis makes several contributions to the area of human-machine communication. In Part I, we study how to learn and evaluate end-to-end, non-task-oriented dialogue systems. We introduce the Ubuntu Dialogue Corpus in Chapter 3, and use it to train several end-to-end dialogue systems. We also evaluate the quality of the responses generated by these systems qualitatively. In Chapter 4, we take several approaches to the problem of automatic dialogue evaluation. We first show in Section 4.1 that many existing evaluation methods based on word-overlap metrics such as BLEU correlate poorly with human judgements of response quality. In Section 4.2, we perform a human study of the next utterance classification (NUC) task and recall metrics, which indicates that they are useful for evaulating dialogue systems in the near term. In Section 4.3 we propose to *learn* a dialogue evaluation model (ADEM) from a dataset of human judgements on the Twitter dataset. While ADEM generalizes well to our test set, we also find that it seems to generalize poorly to other datasets, and thus has not been widely adopted for evaluating dialogue responses.

In Part II, we make contributions to the field of multi-agent RL and emergent communication. In Chapter 5, we propose a new multi-agent RL algorithm that outperforms decentralized RL alternatives on a range of tasks in our Particle World environments, including a cooperative communication game. In Chapter 6 we perform a detailed analysis of evaluation methods for emergent communication, and find that many existing metrics don't account for whether an agent's messages change the behaviour of other agents. Finally in Chapter 7 we make some first steps towards bridging the gap between work on emergent communication and natural language, by testing a variety of schedules combining self-play and supervised learning. Interestingly, we find that it is often not beneficial to emerge a language from scratch, and instead self-play should be incorporated later in training.

Upon reflection, the work that had the most impact was probably the datasets and environments: specifically the Ubuntu Dialogue Corpus and the Particle World environments, which have both been used many times in follow-up work. What helped these datasets become popular was timing, as they were introduced just before the respective fields started becoming more mainstream. For example, research on deep learning for dialogue systems was nascent in 2015, but increased greatly in popularity throughout 2016 thanks in part to work by (Vinyals and Le, 2015) which showed some promising samples generated by a model trained on a movie script dataset. Having a large, publicly available dataset (Ubuntu) with standardized metrics (recall) helped make working on this problem more tractable for researchers that were interested in the area. Similarly, research in multi-agent RL accelerated in 2018, and having a simple suite of environments to test ideas was helfpul. Of course, it is also likely that the introduction of these datasets had some degree of causal effect on the popularity of these fields.

## 8.2 Limitations

In this section, we discuss what we view as broad limitations of this thesis as a whole. In general, while this thesis opens many questions about evaluation, and the relationship between supervised learning of dialogue models and emergent communication, many of these questions do not have satisfactory answers and could benefit from extensive further study.

First, in Part I the question of how to properly evaluate dialogue systems automatically remains unsolved. In Section 4.1 we point out several flaws in existing word-overlap evaluation metrics. While we propose a new method for learning an automatic evaluation metric in Section 4.3, we find that ADEM doesn't transfer well to other datasets, as discussed in Section 4.3.6.3. While the recall metric is interesting for evaluating whether a model can correctly pick out the correct response from a list, it is inherently biased towards retrieval models that are trained via NUC. In the end, human evaluation of entire conversations with the model remains the gold standard. We discuss some promising directions for automatic dialogue evaluation in the next section.

One might also argue that the space of dialogue models considered in Chapter 3 is fairly limited, and doesn't consider many later advancements in dialogue system architectures. This is partly a reflection of the our shifting research interest over time; when this paper was first written in 2016, the number of established dialogue models was quite small. One limitation of our work on dialogue models in Chapter 3 is that the metrics we use to compare the generative to retrieval models either intrinsically favour retrieval models (recall), or are only a proxy metric for topicality rather than dialogue quality (embedding metrics). It would be interesting to perform a human study comparing retrieval and generative models.

Finally, in Part I our focus is solely on the non-task-oriented (or chit-chat) setting, which only comprises part of the overall interest in dialogue systems. Coming up with models and automatic metrics for task-oriented systems is an entirely separate challenge, which we do not address here.

Perhaps the greatest limitation of this thesis is the number of unexplored directions for combining and comparing work on supervised learning of dialogue systems with work on emergent communication and self-play in Part II. We perform a first exploration of this connection in Chapter 7, where we combine supervised learning and self-play in an image-based referential (IBR) game. However, there are many things we do not consider in this chapter. For instance, we frame the problem as minimizing the number of human-in-the-loop samples to achieve a certain performance threshold. But this does not address the more realistic scenario where we can pre-train language models on large amounts of text from the Internet (e.g. (Radford et al., 2019)). It is unclear, if we started from such a model, how much self-play would help for solving language tasks. Our work in Part II was also significantly motivated by the idea that *grounding* is an important concept for building machines that can communicate with humans and act in the world, however this is mostly an empirical question that our work in Part II sheds little light on. Further the IBR game we consider is somewhat removed from the goal of communicating with humans interactively, as in the dialogue systems in Part I. As such, the models we develop in Chapter 5 are not comparable to the models from Chapter 3.

## 8.3 Directions for future work

In the quest to build machines that can interact and do useful things for humans, there are many promising directions. The most compelling recent trend is the use of very large language models based on the Transformer architecture (Vaswani et al., 2017), which have achieved surprisingly strong performance on a variety of NLP tasks (Devlin et al., 2018; Radford et al., 2019; Yang et al., 2019; Dai et al., 2019; Raffel et al., 2019). Recent work has shown that model size, data size, and amount of compute matter significantly more than architectural details (such as width and depth) when training language models, in terms of test loss, and that the performance of these models follows a power law curve (Kaplan et al., 2020). Language modeling is useful for building language-using agents for at least two reasons: (1) it allows us to leverage the huge amount of text data that's available on the Internet; and (2) if a model performs well on the language modeling objective on the test set of such a large dataset, this means it has encoded a significant amount of world knowledge into its weights (for example, given the context "the Prime Minister of Canada is", the model needs to learn to give high probability to "Justin Trudeau"), and this can be used as a starting point for fine-tuning more specific models. Thus, it is possible that scaling language models further and fine-tuning them may have a huge impact on the quality of language-using systems.

Given the strength of large-scale language modeling, a crucial ingredient to investigate is the training data that it is fed. There are many large text datasets that are publicly available (e.g. (Chelba et al., 2013)), however it is unclear what is the best kind of data to train large language models. Is it best to simply have as much data as possible? Or, given a fixed compute budget, is important to emphasize diversity of the data sources? This is an empirical question that will depend on the intended downstream use-cases of the model.

Despite the strength of large-scale language models, they are certainly not the whole picture for developing agents that can use language to accomplish tasks for humans. Most importantly, we will likely want such agents to take actions in the real world. One of the most important open questions in building agents that communicate usefully with humans is: *how can we bridge the gap from large-scale language models to agents that use language to take actions in the world?*

Certainly, it seems like there are significant obstacles. Most starkly, a language model learns only what tokens correlate with other tokens in a dataset, and doesn't explicitly learn a causal model of the real world (Pearl, 2009). Having such a model would allow agents to reason about why certain decisions should be made instead of others, and might make the decision process more interpretable, which is currently not the case for large language models (Lipton, 2018). It is possible that language models can implicitly learn a model of the world that allows it to reason about relations between objects, however the exact properties of such a model has not been thoroughly investigated. Certainly, such a model would look very different from a human's model of the world, as it does not include perceptual data. Thus, it seems like what is necessary is a large model trained to take decisions via reinforcement learning, trained on multiple modalities (text, video, etc.). How one might train such a model is an important direction for future study.

Agents will also need to learn how to interact and deal with other learning agents like itself (and humans) in order to navigate the real world, something that is not captured in the pure language modeling objective. This is where multi-agent RL can come into play; agents could learn to solve tasks with other agents in simulated environments (perhaps via self-play), and transfer this knowledge to taking actions and cooperating with humans. As discussed in the previous section, it's still unclear how to combine self-play and supervised learning when using large-scale language models (both algorithmically, and from an environment point of view).

Finally, and perhaps most importantly, there are significant ethical and safety issues that come with the development of machines that can communicate with humans. The use of machine learning systems in real world applications can have averse effects on many groups. This is primarily because ML systems learn the biases that are inherent in the data they are trained on. When these systems are deployed without consideration for these biases, as has been the case for models predicting recidivism or face recognition, people can suffer (Metz, 2020). There are specific ethical issues surrounding the use of machines that communicate with humans (Henderson et al., 2018): how is the training data biased? To what extent are the models susceptible to adversarial examples? Can privacy be guaranteed? Tackling these problems will require cross-disciplinary efforts from researchers in computer science, sociology, policy, and other areas.

Ultimately, we want to build systems that improve human lives and well-being. Machines that can communicate with humans have great potential to change how we interact with the world; however there are also significant risks if the systems are not doing what we want them to do. An important component of this problem is AI safety, specifically the problem of AI alignment, which stated simply is "how do we align what AI systems do with what humans want to do?". This problem will become increasingly important as we build more competent systems that can perform more and more complex tasks. While there have been some interesting advances in recent years (Abbeel and Ng, 2004; Hadfield-Menell et al., 2016; Christiano et al., 2017), and some specifically in the domain of language (Ziegler et al., 2019), this remains one of the most important open problems in the field of AI.

# Bibliography


M. Abadi and D. G. Andersen. Learning to protect communications with adversarial neural cryptography. *arXiv preprint arXiv:1610.06918*, 2016.

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

R. Al-Rfou, M. Pickett, J. Snaider, Y.-h. Sung, B. Strope, and R. Kurzweil. Conversational contextual cues: The case of personalization and history for response ranking. *arXiv preprint arXiv:1606.00372*, 2016.

G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

J. Albrecht and R. Hwa. Regression for sentence-level MT evaluation with pseudo references. In *Proceedings of the Association for Computational Linguistics*, 2007.

D. Amodei and D. Hernandez. AI and compute. *https://blog. openai. com/aiand-compute*, 2018.

J. Andreas and D. Klein. Alignment-based compositional semantics for instruction following. *arXiv preprint arXiv:1508.06491*, 2015.

J. Andreas, A. Dragan, and D. Klein. Translating neuralese. *arXiv preprint arXiv:1704.06960*, 2017.

K. J. Åström. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

R. J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of mathematical Economics*, 1(1):67–96, 1974.

Z. Awad, A. S. Taghi, P. Sethukumar, P. Ziprin, A. Darzi, and N. S. Tolley. Binary versus 5-point likert scale in assessing otolaryngology trainees in endoscopic sinus surgery. *Otolaryngology—Head and Neck Surgery*, 151(1_suppl):P113–P113, 2014.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

R. E. Banchs. Movie-dic: A movie dialogue corpus for research and development. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, 2012.

S. Banerjee and A. Lavie. 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*, volume 29, pages 65–72, 2005.

P. Baudiš and J. Šedivỳ. Sentence pair scoring: Towards unified framework for text comprehension. *arXiv preprint arXiv:1603.06127*, 2016.

R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2): 157–166, 1994.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

B. Bogin, M. Geva, and J. Berant. Emergence of communication in an interactive world with consistent speakers. *arXiv preprint arXiv:1809.00549*, 2018.

A. Bordes and J. Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1511.06931*, 2016.

A. Bordes, J. Weston, and N. Usunier. Open question answering with weakly supervised embedding models. In *Proceedings of the Meeting on Machine Learning and Knowledge Discovery in Databases*, 2014.

C. Boutilier. Learning conventions in multiagent stochastic domains using likelihood estimates. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, 1996.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *COLING*, 2016.

E. Brodwin. A stanford researcher is pioneering a dramatic shift in how we treat depression–and you can try her new tool right now. *Business Insider*, 2018.

L. D. Brown, T. T. Cai, and A. DasGupta. Interval estimation for a binomial proportion. *Statistical science*, pages 101–117, 2001.

P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*, 2018.

L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews*, 38(2):156, 2008.

C. Callison-Burch, P. Koehn, C. Monz, and O. F. Zaidan. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2011.

K. Cao, A. Lazaridou, M. Lanctot, J. Z. Leibo, K. Tuyls, and S. Clark. Emergent communication through negotiation. *arXiv preprint arXiv:1804.03980*, 2018.

R. Chaabouni, E. Kharitonov, E. Dupoux, and M. Baroni. Anti-efficient encoding in emergent communication. *arXiv preprint arXiv:1905.12561*, 2019.

G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: a bayesian approach. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM, 2003.

C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

B. Chen and C. Cherry. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, 2014.

B. Chen and C. Metz. Google's duplex uses ai to mimic humans (sometimes). *The New York Times*, 2019.

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

M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. Babyai: First steps towards grounded language learning with a human in the loop. *arXiv preprint arXiv:1810.08272*, 2018.

K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

E. Choi, A. Lazaridou, and N. de Freitas. Compositional obverter communication learning from raw visual input. *arXiv preprint arXiv:1804.02341*, 2018.

P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

X. Chu and H. Ye. Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:1710.00336*, 2017.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

M. Cogswell, J. Lu, S. Lee, D. Parikh, and D. Batra. Emergence of compositional language with deep generational transmission. *arXiv preprint arXiv:1904.09067*, 2019.

J. Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968.

K. M. Colby, F. D. Hilf, S. Weber, and H. C. Kraemer. Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. *Artificial Intelligence*, 3:199–221, 1972.

T. Cooijmans, N. Ballas, C. Laurent, and A. Courville. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*, 2016.

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra. Learning cooperative visual dialog agents with deep reinforcement learning. *arXiv preprint arXiv:1703.06585*, 2017.

H. de Vries, K. Shuster, D. Batra, D. Parikh, J. Weston, and D. Kiela. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*, 2018.

J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2009.

D. DeVault, A. Leuski, and K. Sagae. Toward learning and evaluation of dialogue policies with text examples. In *Proceedings of the SIGDIAL 2011 Conference*, pages 39–48. Association for Computational Linguistics, 2011.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. W. Cohen. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481*, 2016.

J. Dodge, A. Gane, X. Zhang, A. Bordes, S. Chopra, A. Miller, A. Szlam, and J. Weston. Evaluating prerequisite qualities for learning end-to-end dialog systems. In *Proceedings of the International Conference on Learning Representations*, 2015.

H. Edwards and A. Storkey. Censoring representations with an adversary. *Proceedings of the International Conference on Learning Representations*, 2016.

S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems*, volume 400, page 409. Citeseer, 1995.

J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

M. Elsner and E. Charniak. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2008.

T. Everitt, P. A. Ortega, E. Barnes, and S. Legg. Understanding agent incentives using causal influence diagrams, part i: Single action settings. *arXiv preprint arXiv:1902.09980*, 2019.

K. Evtimova, A. Drozdov, D. Kiela, and K. Cho. Emergent communication in a multi-modal, multi-step referential game. *arXiv preprint arXiv:1705.10369*, 2017.

J. Farrell and M. Rabin. Cheap talk. *Journal of Economic perspectives*, 10(3): 103–118, 1996.

M. Federico. Bayesian estimation methods for n-gram language model adaptation. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 1. IEEE, 1996.

J. R. Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems 29*. 2016.

J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*, 2017a.

J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR, 2017b.

J. Foerster, F. Song, E. Hughes, N. Burch, I. Dunning, S. Whiteson, M. Botvinick, and M. Bowling. Bayesian action decoding for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1811.01458*, 2018.

P. W. Foltz, W. Kintsch, and T. K. Landauer. The measurement of textual coherence with latent semantic analysis. *Discourse processes*, 25(2-3):285–307, 1998.

G. Forgues, J. Pineau, J. Larchevêque, and R. Tremblay. Bootstrapping dialog systems with word embeddings. In *Proceedings of the Workshop on Modern Machine Learning and NLP, NIPS*, 2014.

M. C. Frank and N. D. Goodman. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998, 2012.

P. Gage. A new algorithm for data compression. *The C Users Journal*, 12(2): 23–38, 1994.

M. Galley, C. Brockett, A. Sordoni, Y. Ji, M. Auli, C. Quirk, M. Mitchell, J. Gao, and B. Dolan. deltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. *arXiv preprint arXiv:1506.06863*, 2015.

S. Gandhe and D. Traum. A semi-automated evaluation metric for dialogue model coherence. In *Situated Dialog in Speech-Based Human-Computer Interaction*, pages 217–225. Springer, 2016.

J. Gauthier and I. Mordatch. A paradigm for situated and goal-driven language learning. *arXiv preprint arXiv:1610.03585*, 2016.

M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In *Advances in neural information processing systems*, 2007.

M. Ghazvininejad, C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W.-t. Yih, and M. Galley. A knowledge-grounded neural conversation model. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

H. Gintis, E. A. Smith, and S. Bowles. Costly signaling and cooperation. *Journal of theoretical biology*, 213(1):103–119, 2001.

A. M. Glenberg and M. P. Kaschak. Grounding language in action. *Psychonomic bulletin & review*, 9(3):558–565, 2002.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, 2014.

I. J. Goodfellow, Y. Bengio, and A. Courville. Deep learning. *An MIT Press book in preparation*, 2015.

L. Graesser, K. Cho, and D. Kiela. Emergent Linguistic Phenomena in Multi-Agent Communication Games. *arXiv:1901.08706 [cs]*, Jan. 2019. arXiv: 1901.08706.

A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6): 602–610, 2005.

K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.

S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

J. K. Gupta, M. Egorov, and M. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

R. Gupta, C. Orasan, and J. van Genabith. Reval: A simple and effective machine translation evaluation metric based on recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan. Cooperative inverse reinforcement learning. In *Advances in neural information processing systems*, 2016.

S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42 (1-3):335–346, 1990.

S. Havrylov and I. Titov. Emergence of language with multi-agent games: learning to communicate with sequences of symbols. In *Advances in Neural Information Processing Systems*, pages 2149–2159, 2017.

M. Henderson, B. Thomson, and S. Young. Deep neural network approach for the dialog state tracking challenge. In *Proceedigns of the Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2013.

M. Henderson, B. Thomson, and J. Williams. The second dialog state tracking challenge. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2014a.

M. Henderson, B. Thomson, and S. Young. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2014b.

P. Henderson, K. Sinha, N. Angelard-Gontier, N. R. Ke, G. Fried, R. Lowe, and J. Pineau. Ethical challenges in data-driven dialogue systems. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018.

K. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, 2015.

R. Higashinaka, T. Meguro, K. Imamura, H. Sugiyama, T. Makino, and Y. Matsuo. Evaluating coherence in open domain conversational systems. In *Proceedings of INTERSPEECH*, pages 130–134, 2014.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, page 91, 1991.

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

S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A field guide to dynamical recurrent neural networks. IEEE Press*, 2001.

E. H. Hovy. Toward finely differentiated evaluation metrics for machine translation. In *Proceedings of the EAGLES Workshop on Standards and Evaluation*, 1999.

J. Hu and M. P. Wellman. Online learning about other agents in a dynamic multiagent system. In *Proceedings of the Second International Conference on Autonomous Agents*. ACM, 1998.

S. M. Huttegger, B. Skyrms, R. Smead, and K. J. Zollman. Evolutionary dynamics of lewis signaling games: signaling systems vs. partial pooling. *Synthese*, 172 (1):177, 2010.

M. Inaba and K. Takahashi. Neural utterance ranking model for conversational dialogue systems. In *Proceedings of the Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2016.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

C. L. Isbell, M. Kearns, D. Kormann, S. Singh, and P. Stone. Cobot in lambdamoo: A social statistics agent. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, 2000.

A. G. Ivakhnenko and V. G. Lapa. Cybernetic predicting devices. Technical report, Purdue School of Electrical Engineering, 1966.

M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

S. Jafarpour, C. Burges, and A. Ritter. Filter, rank, and transfer the knowledge: Learning to chat. *Advances in Ranking*, 10, 2010.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. A. Ortega, D. Strouse, J. Z. Leibo, and N. de Freitas. Intrinsic social motivation via causal influence in multi-agent rl. *arXiv preprint arXiv:1810.08647*, 2018a.

N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. A. Ortega, D. Strouse, J. Z. Leibo, and N. De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. *arXiv preprint arXiv:1810.08647*, 2018b.

F. Jelinek. Self-organized language modeling for speech recognition. *Readings in speech recognition*, pages 450–506, 1990.

K. Jokinen and M. McTear. *Spoken Dialogue Systems*. Morgan Claypool, 2009.

M. I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Artificial neural networks: concept learning*, pages 112–127. 1990.

R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

R. Kadlec, M. Schmid, and J. Kleindienst. Improved deep learning baselines for Ubuntu corpus dialogs. In *Proceedings of the Workshop on Spoken Language Understanding, NIPS*, 2015.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

A. Kannan and O. Vinyals. Adversarial evaluation of dialogue models. *arXiv preprint arXiv:1701.08198*, 2017.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

S. Kim, L. F. D'Haro, R. E. Banchs, J. D. Williams, and M. Henderson. The fourth dialog state tracking challenge. In *Dialogues with Social Robots*. Springer, 2017.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations*, 2014.

S. Kirby. Iterated learning and the evolution of language. *Current Opinion in Neurobiology*, 2014.

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, 2015.

R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1. IEEE, 1995.

S. Kottur, J. M. Moura, S. Lee, and D. Batra. Natural language does not emerge'naturally'in multi-agent dialog. *arXiv preprint arXiv:1706.08502*, 2017.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

J. K. Kummerfeld, S. R. Gouravajhala, J. Peper, V. Athreya, C. Gunasekara, J. Ganhotra, S. S. Patel, L. Polymenakos, and W. S. Lasecki. A large-scale corpus for conversation disentanglement. *arXiv preprint arXiv:1810.11118*, 2018.

T. K. Landauer and S. T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.

M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

A. Lazaridou, A. Peysakhovich, and M. Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

A. Lazaridou, K. M. Hermann, K. Tuyls, and S. Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. *arXiv preprint arXiv:1804.03984*, 2018.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

C. Lee, S. Jung, S. Kim, and G. G. Lee. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication*, 51(5):466–484, 2009.

J. Lee, K. Cho, J. Weston, and D. Kiela. Emergent translation in multi-agent communication. *arXiv preprint arXiv:1710.06922*, 2017.

J. Lee, K. Cho, and D. Kiela. Countering language drift via visual grounding. *arXiv preprint arXiv:1909.04499*, 2019.

A. Lerer and A. Peysakhovich. Learning social conventions in markov games. *arXiv preprint arXiv:1806.10071*, 2018.

A. Lerer and A. Peysakhovich. Learning existing social conventions via observationally augmented self-play. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. ACM, 2019.

D. Lewis. *Convention: A philosophical study*. Harvard University Press, 1969.

M. Lewis, D. Yarats, Y. Dauphin, D. Parikh, and D. Batra. Deal or No Deal? End-to-End Learning of Negotiation Dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

F. Li and M. Bowling. Ease-of-Teaching and Language Structure from Emergent Communication. *arXiv:1906.02403 [cs]*, June 2019. arXiv: 1906.02403.

J. Li. Teaching machines to converse. *arXiv preprint arXiv:2001.11701*, 2020.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the Meeting* *of the North American Chapter of the Association for Computational Linguistics*, 2015.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A persona-based neural conversation model. In *Proceedings of the Association for Computational Linguistics*, 2016a.

J. Li, W. Monroe, A. Ritter, and D. Jurafsky. Deep reinforcement learning for dialogue generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016b.

J. Li, W. Monroe, and D. Jurafsky. Learning to decode for future success. *arXiv preprint arXiv:1701.06549*, 2017.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8, 2004.

Z. C. Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157, 1994.

C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

R. Lowe. A retrospective for “towards an automatic turing test - learning to evaluate dialogue responses". *ML Retrospectives*, 2019.

R. Lowe, N. Pow, I. Serban, and J. Pineau. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2015.

R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. On the evaluation of dialogue systems with next utterance classification. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2016.

R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau. Towards an automatic turing test: Learning to evaluate dialogue responses. *Proceedings of the Association for Computational Linguistics*, 2017a.

R. Lowe, N. Pow, I. V. Serban, L. Charlin, C.-W. Liu, and J. Pineau. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65, 2017b.

R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 2017c.

R. Lowe, J. Foerster, Y.-L. Boureau, J. Pineau, and Y. Dauphin. On the Pitfalls of Measuring Emergent Communication. *arXiv:1903.05168 [cs, stat]*, Mar. 2019. arXiv: 1903.05168.

R. Lowe, A. Gupta, J. Foerster, D. Kiela, and J. Pineau. On the interaction between supervision and self-play in emergent communication. *International Conference on Learning Representations*, 2020.

Y. Luan, Y. Ji, and M. Ostendorf. LSTM based conversation models. *arXiv preprint arXiv:1603.09457*, 2016.

M. Machácek and O. Bojar. Results of the wmt14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Citeseer, 2014.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

J. Markoff and P. Mozur. For sympathetic ear, more chinese turn to smartphone program. *New York Times*, 2015.

L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *Proceedings of the International Conference on Intelligent Robots and Systems*. IEEE, 2007.

L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(01), 2012.

C. Metz. An algorithm that grants freedom, or takes it away. *New York Times*, 2020.

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*, 2010.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013b.

T. Mikolov, A. Joulin, and M. Baroni. A roadmap towards machine intelligence. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 29–61. Springer, 2016.

J. Mitchell and M. Lapata. Vector-based models of semantic composition. In *Proceedings of the Association for Computational Linguistics*, pages 236–244, 2008.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

S. Möller, R. Englert, K. Engelbrecht, V. Hafner, A. Jameson, A. Oulasvirta, A. Raake, and N. Reithinger. Memo: towards automatic usability evaluation of spoken dialogue services by user error simulations. In *Proceedings of INTERSPEECH*, 2006.

I. Mordatch and P. Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

M. Mul, D. Bouchacourt, and E. Bruni. Mastering emergent language: learning to guide in simulated navigation. *arXiv preprint arXiv:1908.05135*, 2019.

G. L. Murray. Autonomy and language learning in a simulated environment. *System*, 27(3):295–308, 1999.

M. Nagata and T. Morimoto. First steps towards statistical modeling of dialogue to predict the speech act type of the next utterance. *Speech communication*, 15 (3-4):193–203, 1994.

L. Nio, S. Sakti, G. Neubig, T. Toda, M. Adriani, and S. Nakamura. Developing non-goal dialog system based on examples of drama television. In *Natural Interaction with Robots, Knowbots and Smartphones*. Springer, 2014a.

L. Nio, S. Sakti, G. Neubig, T. Toda, and S. Nakamura. Utilizing human-to-human conversation examples for a multi domain chat-oriented dialog system. *IEICE Transactions on Information and Systems*, 97(6):1497–1505, 2014b.

S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the International Conference on Machine Learning*. JMLR. org, 2017.

L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11, 2005.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2002.

R. C. Parkinson, K. M. Colby, and W. S. Faught. Conversational language comprehension using integrated pattern-matching and parsing. *Artificial Intelligence*, 9(2):111–134, 1977.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *Proceedings of the International Conference on Machine Learning*, 28:1310–1318, 2013.

J. Pearl. *Causality*. Cambridge university press, 2009.

J. Pearl, M. Glymour, and N. P. Jewell. *Causal inference in statistics: a primer*. John Wiley & Sons, 2016.

K. Pearson. Principal components analysis. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 6(2):559, 1901.

J. Pennington, R. Socher, and C. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

O. Pietquin and H. Hastie. A survey on metrics for the evaluation of user simulations. *The Knowledge Engineering Review*, 28(01):59–73, 2013.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

J. Ramos. Using TF-IDF to determine word relevance in document queries. In *Proceedings of the International Conference on Machine Learning*, 2003.

A. Raux, B. Langner, D. Bohus, A. W. Black, and M. Eskenazi. Let's go public! taking a spoken dialog system to the real world. In *Proceedings of INTER-SPEECH*. Citeseer, 2005.

L. Raviv and I. Arnon. Systematicity, but not compositionality: Examining the emergence of linguistic structure in children and adults using iterated learning. *Cognition*, 181, 2018.

A. Ritter, C. Cherry, and W. Dolan. Unsupervised modeling of twitter conversations. In *Proceedings of the Meeting of the North American Chapter of the Association for Computational Linguistics*, 2010.

A. Ritter, C. Cherry, and W. Dolan. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.

R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, 2000.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

V. Rus and M. Lintean. 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*, 2012.

A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

A. B. Sai, M. D. Gupta, M. M. Khapra, and M. Srinivasan. Re-evaluating ADEM: A deeper look at scoring dialogue responses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2019.

D. Sally. Conversation and cooperation in social dilemmas: A meta-analysis of experiments from 1958 to 1992. *Rationality and society*, 7(1):58–92, 1995.

A. Saxe, J. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

J. Schatzmann, K. Georgila, and S. Young. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2005.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

A. See, S. Roller, D. Kiela, and J. Weston. What makes a good conversation? how controllable attributes affect human judgments. *arXiv preprint arXiv:1902.08654*, 2019.

S. Seneff. Tina: A natural language system for spoken language applications. *Computational linguistics*, 18(1):61–86, 1992.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

I. V. Serban, R. Lowe, L. Charlin, and J. Pineau. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*, 2015.

I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

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

L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*, 2015.

L. Shang, T. Sakai, Z. Lu, H. Li, R. Higashinaka, and Y. Miyao. Overview of the ntcir-12 short text conversation task. *Proceedings of NTCIR-12*, 2016.

A. Singh, T. Jain, and S. Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018.

S. Singh, D. Litman, M. Kearns, and M. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

J. M. Siskind. Grounding language in perception. *Artificial Intelligence Review*, 8 (5-6):371–391, 1994.

J. M. Smith. Honest signalling: The philip sidney game. *Animal Behaviour*, 1991.

K. Smith, H. Brighton, and S. Kirby. Complex systems in language evolution: the cultural emergence of compositional structure. *Advances in Complex Systems*, 6(04), 2003.

Y. Song, R. Yan, X. Li, D. Zhao, and M. Zhang. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*, 2016.

A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue S., and J. Y. Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the ACM International on Conference on Information and Knowledge Management*, 2015a.

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J. Nie, J. Gao, and W. Dolan. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the Meeting of the North American Chapter of the Association for Computational Linguistics*, 2015b.

M. Stanojevic, A. Kamran, P. Koehn, and O. Bojar. Results of the wmt15 metrics shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 2015.

L. Steels. Grounding language through evolutionary language games. *Language Grounding in Robots*, pages 1–22, 2012.

S. Sukhbaatar, A. Szlam, and R. Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems 29*. 2016.

S. Sukhbaatar, I. Kostrikov, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2014.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 1993.

D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432, 2015.

C. Tao, L. Mou, D. Zhao, and R. Yan. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

G. Tesauro. Extending q-learning to general adaptive multi-agent systems. In *Advances in neural information processing systems*, pages 871–878, 2004.

L. Theis, A. v. d. Oord, and M. Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

P. S. Thomas and A. G. Barto. Conjugate markov decision processes. In *Proceedings of the 28th International Conference on Machine Learning*, pages 137–144, 2011.

O. Tieleman, A. Lazaridou, S. Mourad, C. Blundell, and D. Precup. Shaping representations through communication: community size effect in artificial learning systems. *arXiv preprint arXiv:1912.06208*, 2019.

D. Traum, K. Georgila, R. Artstein, and A. Leuski. Evaluating spoken dialogue processing for time-offset interaction. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2015.

A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

D. Uthus and D. Aha. Extending word highlighting in multiparticipant chat. In *Proceedings of the Florida Artificial Intelligence Research Society Conference*, 2013.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

O. Vinyals and Q. Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

K. Wagner, J. A. Reggia, J. Uriagereka, and G. S. Wilkinson. Progress in the simulation of emergent communication and language. *Adaptive Behavior*, 11 (1):37–69, 2003.

M. Walker and S. Whittaker. Mixed initiative in dialogue: An investigation into discourse segmentation. In *Proceedings of the Association for Computational Linguistics*, 1990.

M. Walker, D. Litman, C. Kamm, and A. Abella. PARADISE: A framework for evaluating spoken dialogue agents. In *Proceedings of the Meeting of the European Chapter of the Association for Computational Linguistics*, 1997.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

J. Weizenbaum. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1): 36–45, 1966.

M. Wen, T.and Gasic, N. Mrksic, L. Rojas-Barahona, P. Su, S. Ultes, D. Vandyke, and S. Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.

T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*, 2015.

P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

J. Weston, A. Bordes, S. Chopra, and T. Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 2015.

J. Williams, A. Raux, D. Ramachandran, and A. Black. The dialog state tracking challenge. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2013.

J. D. Williams. Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of the Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014.

J. D. Williams and S. Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

J. D. Williams, M. Henderson, A. Raux, B. Thomson, A. Black, and D. Ramachandran. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124, 2014.

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

R. J. Williams and J. Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Y. Wu, W. Wu, C. Xing, M. Zhou, and Z. Li. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *arXiv preprint arXiv:1612.01627*, 2016.

Y. Xiang, Y. Zhang, X. Zhou, X. Wang, and Y. Qin. Problematic situation analysis and automatic recognition for chinese online conversational system. In *Proceedings of the Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, 2014.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

Z. Xu, B. Liu, B. Wang, C. Sun, and X. Wang. Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling. *arXiv preprint arXiv:1605.05110*, 2016.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, 2019.

S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.

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

S. J. Young. Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 358(1769):1389–1402, 2000.

L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.

Z. Yu, Z. Xu, A. W. Black, and A. I. Rudnicky. Strategy and policy learning for non-task-oriented conversational systems. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 404, 2016.

A. Zahavi. Mate selection—a selection for a handicap. *Journal of Theoretical Biology*, 53(1):205–214, 1975.

S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*, 2018.

T. Zhao and M. Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of the Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2016.

W. Zhu, K. Mo, Y. Zhang, Z. Zhu, X. Peng, and Q. Yang. Flexible end-to-end dialogue system for knowledge grounded conversation. *arXiv preprint arXiv:1709.04264*, 2017.

D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.