Université de Montréal

REPRESENTATION LEARNING FOR DIALOGUE SYSTEMS

Par

Iulian Vlad Serban

Institut Québécois d'Intelligence Artificielle (Mila)
Département d'Informatique et de Recherche Opérationnelle (DIRO),
Faculté des Arts et des Sciences

Thèse présentée à la Faculté des arts et des sciences en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.) en informatique.

mai 2019

Université de Montréal

Faculté des études supérieures et postdoctorales

Ce mémoire intitulé

REPRESENTATION LEARNING FOR DIALOGUE SYSTEMS

Présenté par

Iulian Vlad Serban

a été évalué par un jury composé des personnes suivantes :

Alain Tapp
(président-rapporteur)

Aaron Courville
(directeur de recherche)

Yoshua Bengio
(co-directeur de recherche)

Pascal Vincent
(membre du jury)

Phil Blunsom
Department of Computer Science
Oxford University
(examinateur externe)

I Sommaire

Cette thèse présente une série de mesures prises pour étudier l'apprentissage de représentations (par exemple, l'apprentissage profond) afin de mettre en place des systèmes de dialogue et des agents de conversation virtuels. La thèse est divisée en deux parties générales.

La première partie de la thèse examine l'apprentissage des représentations pour les modèles de dialogue génératifs. Conditionnés sur une séquence de tours à partir d'un dialogue textuel, ces modèles ont la tâche de générer la prochaine réponse appropriée dans le dialogue. Cette partie de la thèse porte sur les modèles séquence-à-séquence, qui est une classe de réseaux de neurones profonds génératifs. Premièrement, nous proposons un modèle d'encodeur-décodeur récurrent hiérarchique ("Hierarchical Recurrent Encoder-Decoder"), qui est une extension du modèle séquence-à-séquence traditionnel incorporant la structure des tours de dialogue. Deuxièmement, nous proposons un modèle de réseau de neurones récurrents multi-résolution ("Multiresolution Recurrent Neural Network"), qui est un modèle empilé séquence-à-séquence avec une représentation stochastique intermédiaire (une "représentation grossière") capturant le contenu sémantique abstrait communiqué entre les locuteurs. Troisièmement, nous proposons le modèle d'encodeur-décodeur récurrent avec variables latentes ("Latent Variable Recurrent Encoder-Decoder"), qui suivent une distribution normale. Les variables latentes sont destinées à la modélisation de l'ambiguïté et l'incertitude qui apparaissent naturellement dans la communication humaine. Les trois modèles sont évalués et comparés sur deux tâches de génération de réponse de dialogue: une tâche de génération de réponses sur la plateforme Twitter et une tâche de génération de réponses de l'assistance technique ("Ubuntu technical response generation task").

La deuxième partie de la thèse étudie l'apprentissage de représentations pour un système de dialogue utilisant l'apprentissage par renforcement dans un contexte réel. Cette partie porte plus particulièrement sur le système "Milabot" construit par l'Institut québécois d'intelligence artificielle (Mila) pour le concours "Amazon Alexa Prize 2017". Le Milabot est un système capable de bavarder avec des humains sur des sujets populaires à la fois par la parole et par le texte. Le système consiste d'un ensemble de modèles de récupération et de génération en langage naturel, comprenant des modèles basés sur des références, des modèles de sac de mots et des variantes des modèles décrits ci-dessus. Cette partie de la thèse se concentre sur la tâche de sélection de réponse. À partir d'une séquence de tours de dialogues et d'un ensemble des réponses possibles, le système doit sélectionner une réponse appropriée à fournir à l'utilisateur. Une approche d'apprentissage par renforcement basée sur un modèle appelée "Bottleneck Simulator" est proposée pour sélectionner le candidat approprié pour la réponse. Le "Bottleneck Simulator" apprend un modèle approximatif de l'environnement en se basant sur les trajectoires de dialogue observées et le "crowdsourcing", tout en utilisant un état abstrait représentant la sémantique du discours.

Le modèle d'environnement est ensuite utilisé pour apprendre une stratégie d'apprentissage du renforcement par le biais de simulations. La stratégie apprise a été évaluée et comparée à des approches concurrentes via des tests A / B avec des utilisateurs réel, où elle démontre d'excellente performance.

Mots clés: apprentissage profond, apprentissage par renforcement, systèmes de dialogue, agents de conversation virtuels, les modèles de dialogue génératifs

II Summary

This thesis presents a series of steps taken towards investigating representation learning (e.g. deep learning) for building dialogue systems and conversational agents. The thesis is split into two general parts.

The first part of the thesis investigates representation learning for generative dialogue models. Conditioned on a sequence of turns from a text-based dialogue, these models are tasked with generating the next, appropriate response in the dialogue. This part of the thesis focuses on sequence-to-sequence models, a class of generative deep neural networks. First, we propose the Hierarchical Recurrent Encoder-Decoder model, which is an extension of the vanilla sequence-to sequence model incorporating the turn-taking structure of dialogues. Second, we propose the Multiresolution Recurrent Neural Network model, which is a stacked sequence-to-sequence model with an intermediate, stochastic representation (a "coarse representation") capturing the abstract semantic content communicated between the dialogue speakers. Third, we propose the Latent Variable Recurrent Encoder-Decoder model, which is a variant of the Hierarchical Recurrent Encoder-Decoder model with latent, stochastic normally-distributed variables. The latent, stochastic variables are intended for modelling the ambiguity and uncertainty occurring naturally in human language communication. The three models are evaluated and compared on two dialogue response generation tasks: a Twitter response generation task and the Ubuntu technical response generation task.

The second part of the thesis investigates representation learning for a real-world reinforcement learning dialogue system. Specifically, this part focuses on the Milabot system built by the Quebec Artificial Intelligence Institute (Mila) for the Amazon Alexa Prize 2017 competition. Milabot is a system capable of conversing with humans on popular small talk topics through both speech and text. The system consists of an ensemble of natural language retrieval and generation models, including template-based models, bag-of-words models, and variants of the models discussed in the first part of the thesis. This part of the thesis focuses on the response selection task. Given a sequence of turns from a dialogue and a set of candidate responses, the system must select an appropriate response to give the user. A model-based reinforcement learning approach, called the Bottleneck Simulator, is proposed for selecting the appropriate candidate response. The Bottleneck Simulator learns an approximate model of the environment based on observed dialogue trajectories and human crowdsourcing, while utilizing an abstract (bottleneck) state representing high-level discourse semantics. The learned environment model is then employed to learn a reinforcement learning policy through rollout simulations. The learned policy has been evaluated and compared to competing approaches through A/B testing with real-world users, where it was found to yield excellent performance.

Keywords: deep learning, reinforcement learning, dialogue systems, conversational agents, generative dialogue models

Contents

III List of Tables

IV List of Figures

V Notation

- $\{\cdot\}$ denotes a set of items.
- $(\cdot)$ denotes a sequence of items.
- $\{x_i\}_{i=1}^{I}$ (or simply $\{x_i\}_i$) denotes a set of items $x_1, x_2, \ldots, x_{I-1}, x_I$.
- $|V|$, where V is a set, is the cardinality of the set (for example, if V is a finite set, then $|V|$ is the number of elements in the set).
- $A \times B = \{(a, b) \mid a \in A, b \in B\}$, where A and B are sets of items, denotes the Cartesian product of A and B.
- $\mathbb{R}$ denotes the set of real-valued numbers.
- $\mathbb{R}^n$ denotes the set of real-valued numbers in n dimensions.
- $\mathbb{N}$ denotes the set of non-negative integer numbers.
- $\mathbb{N}^+$ denotes the set of positive integer numbers.
- $\mathbb{N}^-$ denotes the set of negative integer numbers.
- $a \in \mathbb{R}$ denotes a real-valued variable named a.
- $[a, b]$, where $a, b \in \mathbb{R}$ and $b > a$, denotes the closed set of real-valued numbers between a and b, including a and b.
- (a, b), where $a, b \in \mathbb{R}$ and $b > a$, denotes the open set of real-valued numbers between a and b, excluding a and b.
- $\mathbf{a} \in \mathbb{R}^n$ denotes a real-valued vector of n dimensions.
- $A \in \mathbb{R}^{n \times m}$ denotes a real-valued matrix of $n \times m$ dimensions.
- A^T and A^T both denote the transpose of the matrix A.
- $i = 1, \ldots, n$ means that i will take integer values $1, 2, 3, 4, 5$ and so on until and including integer n.
- An n-gram is a sequence of n consecutive words (or tokens).
- $w \in U$, where U is a sequence of tokens, denotes a token inside U.

- $\exp(x)$ and e^x denotes the exponential function of the value x.
- $\log(x)$ and $ln(x)$ denotes the natural logarithm function of the value x.
- $tanh(x)$ denotes the hyperbolic tangent taken of value x.
- $f'(x)$ denotes the derivative of the function f w.r.t. variable x.
- $\frac{\delta}{\delta x} f(x)$ denotes the derivative of the function f w.r.t. variable x.
- $\nabla_\theta f_\theta(x)$ denotes the derivative of the function f w.r.t. parameters θ. If θ is a vector, then it denotes the Jacobian matrix.
- $x \cdot y$, where x and y are vectors or matrices, denotes the element-wise product between x and y.
- x often denotes an input variable (e.g. a real-valued variable or an input sequence of string tokens).
- y often denotes an output variable (e.g. an output label, such as a user intention label).
- θ and $\hat{\theta}$ usually denote model parameters.
- ψ and $\hat{\psi}$ usually denote model parameters.
- $P_\theta(\cdot)$ usually denotes the probabilistic model parametrized by parameters θ.
- $x \sim P_\theta(x)$ denotes a sample of the random variable x following the probabilistic model parametrized by parameters θ.
- $\mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}, \Sigma)$ denotes the probability of variable $\mathbf{x}$ under a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix Σ.
- $x \sim \text{Uniform}(a, b)$ denotes that x is an integer random variable sampled at uniformly random from the set $\{a, a+1, \cdots, b-1, b\}$, with $a, b \in \mathbb{N}$ and $b > a$.
- $x \sim \text{Uniform}(A)$, where A is a finite discrete set, denotes that x is a random variable sampled at uniformly random from the set A.
- $\mathbb{E}_{x \sim P(x)}[f(x)] = \sum_x P(x) f(x) = \int P(x) f(x) dx$ is the expectation of the function $f(x)$ w.r.t. the random variable x following the distribution given by the probability or density function P.

- KL$[Q||P] = -\int Q(x) \log(Q(x)/P(x))dx$ is the Kullback-Leibler (KL) divergence between the two probability distributions $P(x)$ and $Q(x)$.
- $1_{(.)}$ denotes a Dirac-delta function, which equals one if the statement $(\cdot)$ is true, and otherwise equals zero.
- $\forall$ denotes the *for all* operator.
- The phrase *s. t.* is an abbreviation of the phrase *subject to*.

VI Acknowledgements

I had no idea of what I had commited myself to when I decided to study for a Ph.D. degree. I made this decision while I was still studying my master's degree at University College London (UCL). Back then, my future supervisor Yoshua Bengio had come to UCL to give a talk about some of the advances of deep learning. Although I did not understand much of the talk, I found the abstract ideas he presented facinating and I thought it was a pity that they were not covered in more depth in any of my courses at the time. After his talk, another student and I had a chance to chat with him about his research and about machine learning in general. As the discussion turned to the topic of future research and the many real-world applications of our field, I saw a spark in his eyes and I realized that this was someone I wanted to work with. Afterwards, I went home and discovered that I had, in fact, already read and cited several of his older papers in my bachelor's thesis. It was at this point that I decided to apply for studying a Ph.D. degree in his research lab.

I was accepted into the Ph.D. program, and the following fall I arrived in Montreal. This was at once both a very exciting and a frightening time. I had never been to Canada before, I did not know anyone there and I certainly wasn't prepared for the language barrier or the harsh weather to come. Fortunately, the professors and students in the lab were very welcoming and helpful. It turned out that, like myself, many of them had come from abroad. This was when I got to know Aaron Courville, my second supervisor. I quickly realized that he was one of the people I could communicate with most easily. Every time one of us explained a complex idea, it seemed as if the other one instantly understood it and started extending it. I also found out that we shared many of the same ideas about probabilistic graphical models and their uses in deep learning models. I was fortunate enough that Aaron had time available and could take me on as one of his students. Through discussions with Yoshua and Aaron, I decided to focus my research on the application of deep learning and reinforcement learning for building dialogue systems. Shortly after, I was introduced to Joelle Pineau, my future unofficial advisor and long-term collaborator. Joelle had previously worked on applying reinforcement learning for building dialogue systems, and she was keen to start a new research group focused on it. A few months later, we started a research group with other students investigating deep learning and reinforcement learning techniques for building dialogue systems. This research group was of great help, because its weekly meetings provided a common structure and framework for us to work together. That is how the journey of my Ph.D. began nearly 5 years ago, with the help and supervision of Yoshua, Aaron and Joelle. I'd like to thank each of them for their enduring help and support throughout the years. Without them, my Ph.D. degree would never have been possible.

When I started, I thought that a Ph.D. mainly involved reading papers, running experiments and then writing papers: a set of well-defined, repeatable steps aimed towards advancing scientific

knowledge bit by bit. Although this is certainly part of it, I later discovered that a Ph.D. is also a long journey of exploration, discovery and reflection. It is a meticulous process of asking questions and seeking answers, where both the questions and answers are always kept under scrutiny. This process involves learning the assumptions and general world view of the researchers in the field and questioning them, as well as learning the general methodology of the field (including how to set up experiments, use software libraries, write scientific papers, and so on). However, the process also demands patience, persistence, diligence and last, but not least, a will to go on solitary campaigns to promote new ideas, ask new questions and give new answers. For accompanying me along this long and difficult journey, I would like to thank all of my collaborators who I have worked with throughout the years, including students and university staff members at University of Montreal and McGill University. In particular, I would like to thank Alessandro Sordoni, Caglar Gulcehre and Laurent Charlin, who have been supportive and who have helped teach and mentor me throughout the years. I would also like to give particular thanks to students in the dialogue research group: Nicolas Angelard-Gontier, Peter Henderson, Ryan Lowe, Michael Noseworthy, Prasanna Parthasarathi, Nissan Pow and Koustuv Sinha. Without their feedback and brainstorming meetings, many of the ideas proposed in this thesis would never habe been possible. I would also like to thank our team members from the Amazon Alexa Prize 2017 competition: Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Rajeswar, Alexandre de Brebisson, Jose M. R. Sotelo, Dendi Suhubdy, Vincent Michalski and Alexandre Nguyen. Although we did not win the competition, I am very proud to have worked together with them. Even though our work together is not covered in this thesis, I also had fruitful collaborations with Sungjin Ahn, Sarath Chandar, Alberto Garcia-Duran, Caglar Gulcehre and Alexander G. Ororbia II over the past few years. I am thankful to each of them for their collaboration and time. I would also like to thank all other team members of the Quebec Artificial Intelligence Institute (Mila), including in particular Hugo Larochelle, Frederic Bastien, Pascal Lamblin and Arnaud Bergeron for their help and technical assistance. I would also like to thank my collaborators from IBM Research: Tim Klinger, Kartik Talamadupula, Gerald Tesauro and Bowen Zhou. I hope our paths will cross again in the future. Last, not but not least, I would like to thank the thesis jury members for reading my thesis and providing their comments: Alain Tapp, Pascal Vincent, Aaron Courville, Yoshua Bengio and Phil Blunsom.

Finally, and most of all, I would like to thank my love, my wife and partner in life, Ansona Onyi Ching and our son Octavian Ching Serban. Ansona has stood by my side throughout the years, since before I started studying my Ph.D. degree, and she has always helped me shoulder the inevitable ups and downs of the journey. Much of my inspiration and courage to move forward with bold decisions and ideas has come from her. Octavian has in turn given the two of us a life of happiness and purpose, which I am not sure could be matched by any amount of scientific achievement. None of this would have been possible without their love and support.

1 Introduction

1.1 Motivation

Over the past decades, computers have become a ubiquitous and essential part of modern society. As a part of this transformation, the way we interact with computers has changed tremendously. The computers in the middle of the 20th century could only be programmed manually by swapping in different punch cards. Later, computers were equipped with an extensive internal memory and could be programmed by interacting with a terminal using a keyboard. In the 80s, a new wave of computers, known as personal computers, started to appear with graphical user interfaces, which allowed users to more naturally interact with them using both mouse and keyboard devices. Since then, other types of computers have emerged, including mobile smartphones, tablet computers and GPS navigation devices, which can be interacted with using touch gestures (e.g. touch user interfaces), as well as virtual reality platforms and the XBox 360 (Kinect), which can be interacted with using head and body gestures. Since at least the 90s, automated telephone systems (called spoken dialogue systems) have also been developed, which could understand natural language speech, for example by AT&T Research Labs. However, these systems were often exclusively built for one particular task with extremely limited capabilities, compared to the general interfaces discussed earlier. The reader is likely familiar with all of these technologies, but highlighting the chronological development of these technologies and their transformations serve an important purpose. With each new transformation computer interfaces have become more intelligent and more natural to interact with.

Very recently, software companies (e.g. Apple, Microsoft, Google, Amazon and Nuance) have started to develop general natural language dialogue systems, called intelligent personal assistants. These personal assistants aim to bridge one of the ultimate communication gaps between humans and computers, by allowing humans to interact with computers directly using spoken natural language for carrying out a multitude of tasks. Unfortunately, understanding and generating natural language is a very difficult problem. Therefore, it is not surprising that these technologies are still in their very infancy. This thesis is motivated by these technological developments and the related outstanding challenges. In addition to intelligent personal assistants, dialogue systems have also been deployed as supportive virtual friends (Markoff and Mozur, 2015; Dillet, 2016), healthcare assistants (Furness, 2016; Brodwin, 2018) and tutoring assistants (Nye et al., 2014).

The purpose of this thesis is to make a contribution to the research fields of natural language processing and representation learning, with the specific aim of building general-purpose natural language dialogue systems. In particular, the thesis will focus on probabilistic generative models for building natural language dialogue systems using large text corpora.

1.2 Central Assumptions

As is the case with much of scientific research, the work in this thesis is built upon several key assumptions. These key assumptions constitute the foundations underlying and motivating the work presented in this thesis. Some of these assumptions are well-established in the field, while others might be more contestable. This section provides an overview and discussion of these assumptions.

The first key assumption of this thesis is that communication between humans and machines should be collaborative in nature and be beneficial to all parties. In any conversation, both the human interlocutors (human speakers) and the machine interlocutors (machine speakers) are agents in their own respect, each one with their own goals. The reason that any two interlocutors might have a conversation in the first place must be because they both believe that there is something to be gained through the conversation. In other words, each interlocutor believes that there exists an alignment between their own goals and the goals of the other party, and that by conducting a conversation they may both benefit from it. However, it is important to stress that their goals are not necessarily perfectly aligned. Let's consider the example of a dialogue system selling flight tickets. In addition to its primary goal of finding a suitable ticket for a human customer, the system may have a secondary goal to maximize profits by selling the most expensive ticket commensurate with the human customer's spending budget. This secondary goal would be in direct conflict with the human customer, if the human customer has a secondary goal of purchasing an inexpensive ticket.

The second key assumption is that, in general, the human and machine interlocutors only have access to partial information about the state of the world, about the other interlocutor's information and goals and even about their own goals. For example, the dialogue system selling flight tickets cannot know the goals of a human customer beforehand, such as their departure city, destination city or even their spending budget. On the other hand, the human customer does not know which flight tickets are available and at what prices. The human customer may not even know their destination city or their exact spending budget. This is something the human customer might decide on based on the options presented by the dialogue system (e.g. based on the available destination cities and the price differences between economy and business class shown by the dialogue system).

Although these two key assumptions may appear evident to the avid reader, they go against some of the assumptions implied by some of the literature on goal-driven dialogue systems. In particular, research on *voice control systems* (or *voice command systems*) has sometimes made the implicit assumption that a goal-driven dialogue system should serve as a direct substitute for keyboard input, which will convert the human interlocutor's speech to an appropriate query and submit

it to an application or a service API. For example, consider the case of a voice-controlled GPS-based navigation system. This system might only expect the human user to mention a destination (e.g. an address or a location name) and would then, based on the received destination, map out a route for the human user and display it in a graphical user interface. Strictly speaking, this is not a collaborative dialogue where both parties stand to benefit. Rather it is a one-way communication channel, where the system's main purpose is to convert the words spoken by the human user into an appropriate format (e.g. an address string represented in a formal language) for making a query to a subsequent application or service. This is an example of semantic parsing (Wilks and Fass, 1992; Kamath and Das, 2019)

This simple system further makes the assumption that the human user has access to all relevant information, including their own goal (e.g. the exact destination address and the format the system requires).

The previous two assumptions discussed were related to the form of communication between the human and machine interlocutors. The next set of assumptions is related to the building of dialogue systems. A key assumption here is that versatile dialogue systems, which both satisfy the previous assumptions and are capable of solving real-world problems through effective and natural interactions with humans, can only be built by incorporating data-driven approaches. Such dialogue systems must incorporate modules based on data-driven approaches, such as machine learning, in order to solve either all or a subset of the underlying engineering problems (for example, natural language understanding, natural language generation and general decision making). This assumption has been adopted widely by the dialogue system research community, as will be discussed below. However, it should be noted how this assumption stands in contrast to predominantly rule-based dialogue systems (such as the ELIZA system and the ALICE system discussed later). Nevertheless, this assumption seems reasonable given the complexity of many of the underlying engineering problems. Consider, for example, the natural language understanding problem of classifying the intention of spoken utterances. Given the magnitude of possible intentions, the diversity of ways in which each intention can be formulated, and finally the contextual, ambiguous and error-prone nature of natural language, it would seem extremely difficult to build a deterministic, rule-based system to map any utterance to its underlying intention.

This thesis focuses on building dialogue systems using deep learning (a branch of machine learning), which is particularly suitable for large-scale data-driven machine learning. As will be discussed later on, the field of deep learning has made tremendous advances and helped set new state-of-the-art performance records across a variety of natural language processing tasks over the past few years. Many of the advances of deep learning have helped with natural language representations (e.g. methods for representing words, phrases and sentences) and natural language generation (e.g. generating phrases and sentences conditioned on specific information), which constitute

sub-problems faced by most dialogue systems. This makes deep learning particularly relevant for research on building general-purpose natural language dialogue systems.

The final key assumption of this thesis is based on the premise that humans learn about the world and about how to communicate through natural language by observing and interacting with others. For example, a toddler might hear a word spoken by a parent and then learn to associate that word with a particular object in the world. As a more elaborate example, consider a student studying deep learning, who is in the process of implementing a machine learning model. She might search on the Internet for similar implementations and find a relevant discussion thread on a forum website (such as Reddit or Stack Overflow). Suppose that on this discussion thread, another person exposes a solution to a similar problem and receives feedback from others about missing aspects in the implementation. By reading through the discussion thread, our protagonist might learn about the subtasks involved in her own implementation. Using this information, she might decompose the task into subtasks, with which she is already familiar, and finalize her own implementation. Alternatively, she may seek additional help by asking a related question in the discussion thread. Although our premise is that humans learn a significant amount of knowledge about the world and about how to communicate by observing and interacting, the reader should note that the premise is *not* that *all knowledge* is learned or acquired through these mechanisms. A significant amount of learning is bound to also occur through other mechanisms (for example, observing others do a task and then imitating it without any two-way communication). The premise is only that a significant amount of information is being learned by observing and interacting with others, and that this is a valuable source of information in its own right.

By accepting this premise – that humans learn a significant amount about the world and about how to communicate by observing others and by interacting with others – we arrive at the final key assumption of this thesis. The assumption is that a machine can *also* learn a significant amount of information about the world and about how to communicate in natural language by observing and interacting with others. This last assumption is perhaps the most contestable of all the assumptions discussed so far. However, it may be mitigated if it is further assumed that the system has access to other information, such as knowledge bases and encyclopedias.

Unfortunately, it is difficult to deploy real-world machine learning systems and, often even more difficult, to entice human users to interact with such systems and to collect relevant interaction data. Therefore, in the first part of this thesis, we will restrict the last assumption even further. Specifically, we will assume that a significant amount of information about the world and about how to communicate can be learned by simply observing the interactions of others (e.g. interactions between human interlocutors). In other words, by giving a machine access to a corpus or a stream of data containing interactions between human interlocutors, the machine can learn a substantial amount of information about the real world and about how to communicate in natural language.

This most restrictive version of the last assumption poses a problem for the so-called grounding process of natural language (Harnad, 1990; Quine, 2013). Without going into further details, one part of this process is where a learner learns to associate linguistic expressions with their meanings, such as words and their intended referents. This is very difficult to accomplish without any additional information. Consider the thought experiment presented by Harnad (1990): *"Suppose you had to learn Chinese as a second language and the only source of information you had was a Chinese/Chinese dictionary. The trip through the dictionary would amount to a merry-go-round, passing endlessly from one meaningless symbol or symbol-string to another, never coming to a halt on what anything meant."*. This thought experiment is very similar to the most restrictive version of our last assumption, where the system has to learn the meaning of words, phrases, dialogue turns and entire interactions by only observing the conversations between third-party interlocutors. However, in our case, the system has access to more information than in Harnad (1990)'s thought experiment. The system observes the interactions between interlocutors and can identify and distinguish the different interlocutors. As a minimum, the observed phrases can be grounded by the interlocutor who spoke them. In addition, the system knows that the dialogues are collaborative in nature, and that the majority of dialogues are beneficial to each party and involve some form of information exchange. Given this additional knowledge about each conversation, the system may be able to ground more of the linguistic content. For example, phrases emitted by one interlocutor, but not by another interlocutor, might be grounded as a "goal statement" or as an "information exchange" since such phrases must be present in the dialogue and would often only be spoken by one interlocutor.[1] Naturally, the process of grounding natural language becomes easier if the system has access to other information (e.g. knowledge bases, encyclopedias) or if the system can interact with human users. This is the case for the second part of the thesis.

1.3 Thesis Structure

The thesis is structured as follows.

Chapter 2 covers background theory related to machine learning and dialogue systems. The chapter is split into three parts. The first part focuses on probabilistic generative models, which form the foundation and act as a unifying framework for much of the work presented in this thesis. Neural network models are also presented here. The second part introduces reinforcement learning, a set of techniques used extensively later in the thesis. The third part discusses dialogue systems in detail, including system components, methods for optimizing system components and methods for system evaluation.

Chapter 3 proposes three sequence-to-sequence models, a class of generative deep neural net-

[1] For the sake of this argument, we will assume that the associated meaning of a phrase could be probabilistic.

works, for building generative dialogue models. Given a sequence of turns from a text-based dialogue, these models aim to generate an appropriate next response in the dialogue. The three models proposed are the Hierarchical Recurrent Encoder-Decoder (HRED), Multiresolution Recurrent Neural Network (MrRNN) and Latent Variable Recurrent Encoder-Decoder (VHRED). For each model, the contribution of the author of this thesis, the motivation, the prior related work and the model architecture and corresponding learning algorithm are discussed. Following this, experiments are presented on two dialogue response generation tasks: a Twitter response generation task and a Ubuntu technical response generation task. The chapter concludes with a general discussion and directions for future research.

Chapter 4 investigates a framework for building dialogue systems, based on combining representation learning and reinforcement learning, in order to develop a non-goal-driven dialogue system capable of learning from real-world interactions with humans. The work presented here focuses on the Milabot system built by the Quebec Artificial Intelligence Institute (Mila) for the Amazon Alexa Prize 2017 competition. The chapter first discusses the contribution of the author of this thesis. The chapter then discusses the motivation of the new framework and compares it to the earlier task of building generative dialogue models. Following this, a review of prior related work is presented. Then, the chapter presents an overview of the Milabot system and its underlying ensemble system, which consists of models generating natural language system responses. The problem of selecting an appropriate system response is presented next and framed as a sequential decision making problem, motivated by reinforcement learning methods. Following this, several reinforcement learning algorithms and supervised learning algorithms are proposed in order to learn policies capable of selecting an appropriate system response. In particular, a model-based reinforcement learning algorithm, named the *Bottleneck Simulator*, is proposed. Then, the chapter presents experiments evaluating the proposed policies, conducted based on real-world users, crowdsourced human annotations and simulations. Finally, the chapter concludes with a broader discussion and directions for future research.

Chapter 5 concludes the thesis. The chapter provides a brief summary of the work carried out in the thesis, reviews the main conclusions and provides a bird's-eye view of the work from the perspective of probabilistic generative models.

2 Technical Background

2.1 Probabilistic Generative Models

This thesis focuses on the field known as machine learning, a sub-field of computer science, statistics and mathematics (Bishop, 2006; Goodfellow et al., 2016). This chapter will introduce the technical background required to understand the reminder of the thesis and also provide pointers for further reading.

Arthur Lee Samuel defined the machine learning field as follows: *"[A field] of study that gives computers the ability to learn without being explicitly programmed"* (Simon, 2013). In this sentence, Samuel highlights precisely the advantage of machine learning for solving natural language generation and understanding problems. It is humanly impossible to explicitly write down rules for understanding and generating every relevant sentence for every conceivable natural language processing task. Therefore, it is necessary to build a computer with the ability to learn without being explicitly programmed. This is often done by letting a computer program learn from examples.

In the following, we will assume that the reader is familiar with basic calculus and probability theory, including concepts such as integrals, linear algebra, random variables, probability distributions, expectations, probabilistic independence and so on. In case the reader is not familiar with this material, please refer to Friedman et al. (2001) and Bishop (2006) for a detailed introduction to all of these. As another reference, the reader may also refer to Goodfellow et al. (2016).

The first concept we introduce is the probabilistic directed graphical model. A probabilistic directed graphical model is a set of random variables $\mathbf{x} = \{x_m\}_{m=1}^{M}$ and an associated directed graph $G = \{\{x_m\}_{m=1}^{M}, \{e_i\}_{i=1}^{I}\}$, with vertices (nodes) x_m, for $m = 1, \ldots, M$, and edges e_i, for $i = 1, \ldots, I$. The nodes are random variables. Each edge $e \in G$ has a tail, which corresponds to its origin node and a head, which corresponds to the node it is pointing to (different from the origin node). We define $\text{Pa}(x_m)$ as the set of parents of the random variable x_m, where $x_j \in \text{Pa}(x_m)$ if there exists an edge with tail x_j and head x_m. The graph G must then satisfy the following factorization of the distribution over $\mathbf{x}$:

$$P(\mathbf{x}) = \prod_{m=1}^{M} p(x_m|\text{Pa}(x_m)). \tag{1}$$

This factorization is crucial for understanding the relationships between the random variables. Given a probabilistic directed graphical model, we are able to follow the generative process of the model as well as deduce independence statements about the underlying random variables (Bishop, 2006). To illustrate this, take the directed graphical model shown in Figure 1 as an example. This model has random variables x_1, x_2, x_3, which according to the edges can be factorized as follows:

$$P(x_1, x_2, x_3) = P(x_1)P(x_2)P(x_3|x_1, x_2)$$

Based on this factorization, we may deduce that $x_1 \perp\!\!\!\perp x_2$, i.e. that x_1 is unconditionally independent of x_2. We arrive at this result by integrating out x_3:

$$P(x_1.x_2) = \int P(x_1, x_2, x_3)dx_3 = \int P(x_1)P(x_2)P(x_3|x_1, x_2)dx_3$$
$$= P(x_1)P(x_2)\int P(x_3|x_1, x_2)dx_3 = P(x_1)P(x_2)$$

Importantly we always assume that the probabilistic directed graphical model be non-acyclic.

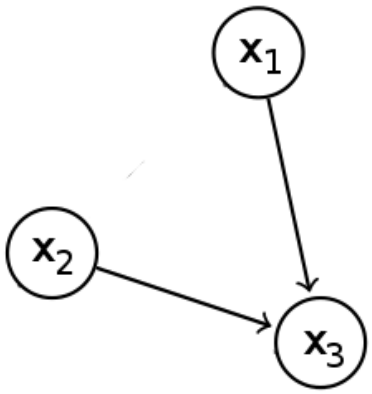

Figure 1: Example of a probabilistic directed graphical model.

In other words, there cannot exist any directed paths (sequence of connected edges) starting and ending at the same node.

2.1.1 n-Gram Models

An important class of probabilistic models are the n-gram models for discrete sequences, where $n \in \mathbb{N}$ and where $\mathbb{N}$ denotes the set of positive integers. Let $\mathbf{w} = (w_1, \ldots, w_M)$ be a sequence of M discrete symbols, where $w_m \in V$ for discrete set V. For example, the variables may be the words of a natural language dialogue or the words of a web document, represented by their indices. The n-gram model, with parameters θ, assumes the distribution over variables factorizes:

$$P_\theta(\mathbf{w}) = P_\theta(w_1, \ldots, w_M)$$
$$= P_\theta(w_1)P_\theta(w_2|w_1)\cdots P_\theta(w_{n-1}|w_1, \ldots, w_{n-2})\prod_{m=n}^{M} P_\theta(w_m|w_{m-n+1}, \ldots, w_{m-1})$$

The key approximation is that the probabilities over each variable can be computed using only the previous $n-1$ tokens:

$$P(w_m|w_1, \ldots, w_{m-1}) \approx P_\theta(w_m|w_{m-n+1}, \ldots, w_{m-1})$$
$$= \theta_{w_m, w_{m-n+1}, \ldots, w_{m-1}},$$

where $\theta_{v,w_{m-n+1},\ldots,w_{m-1}} \in [0,1]$ is the probability of observing token v given the $n-1$ previous tokens $w_{m-n+1},\ldots,w_{m-1}$, which must sum to one: $\sum_{v\in V}\theta_{v,w_{m-n+1},\ldots,w_{m-1}} = 1$. For the 2-gram model, also known as the bigram model, the factorization corresponds to the directed graphical model shown in Figure 2. This model is a probabilistic generative model, since it can assign a probability to any sequence of variables $w_1,\ldots,w_M$ and since it can generate any such sequence by sampling one variable at a time (first sampling w_1, then sampling w_2 conditioned on w_1 and so on). This model is used widely in natural language processing applications (Goodman, 2001).

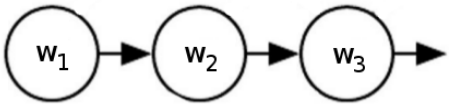

Figure 2: Probabilistic graphical model for bigram (2-gram) model.

Let $\{\mathbf{w}^i\}_{i=1}^I$ be a set of I example sequences, called the training dataset. We assume that the example sequences are independent and identically distributed. The model parameters θ may be estimated (learned) by maximizing the log-likelihood on the training set:

$$\theta = \arg\max_{\theta'} \sum_i \log P_{\theta'}(\mathbf{w}^i).$$

This is done by setting $\theta_{v,w_{m-n+1},\ldots,w_{m-1}}$ to be proportional to the number of times token v was observed after tokens $w_{m-n+1},\ldots,w_{m-1}$ in the training set. In practice, the parameters are often normally regularised or learned with Bayesian approaches (Goodman, 2001).

The approximation discussed earlier is problematic. As n grows, the model begins to suffer from what is known as the *curse of dimensionality* (Richard, 1961; Bishop, 2006). Since the variables are discrete, the number of possible combinations of n variables is $|V|^n$, which grows exponentially with n. This means that to estimate parameters θ, the model requires a number of data examples exponential in n. Therefore, in practice, n is usually a small number such as 3 or 4.

2.1.2 Recurrent Neural Networks

The second class of models we consider are known as recurrent neural networks (RNNs). We will focus on the well-established recurrent neural network language model (RNNLM) (Mikolov et al., 2010; Bengio et al., 2003). Other variants have been applied to diverse sequential tasks, including speech synthesis (Chung et al., 2015), handwriting generation (Graves, 2013) and music composition (Boulanger-Lewandowski et al., 2012; Eck and Schmidhuber, 2002). As before, let $w_1,\ldots,w_M$ be a sequence of discrete variables, such that $w_m \in V$ for a set V. We shall call V the vocabulary, and each discrete variable w_m a token. The RNNLM is a probabilistic generative

model, with parameters θ, which decomposes the probability over tokens:

$$P_\theta(w_1, \ldots, w_M) = \prod_{m=1}^{M} P_\theta(w_m | w_1, \ldots, w_{m-1}). \tag{2}$$

Unlike the n-gram models, the RNNLM does not make a hard assumption restricting the distribution over a token to only depend on the $n-1$ previous tokens. Instead, it parametrizes the conditional output distribution over tokens as:

$$P_\theta(w_{m+1} = v | w_1, \ldots, w_m) = \frac{\exp(g(h_m, v))}{\sum_{v' \in V} \exp(g(h_m, v'))}, \tag{3}$$

$$g(h_m, v) = O_v^{\mathrm{T}} h_m, \tag{4}$$

$$h_m = f(h_{m-1}, I_{w_m}), \tag{5}$$

where $h_m \in \mathbb{R}^{d_h}$, for $m = 1, \ldots, M$, are real-valued vectors called hidden states with dimensionality $d_h \in \mathbb{N}$. The function f is a non-linear smooth function called the hidden state update function. For each time step (each token) it combines the previous hidden state h_{m-1} with the current token input w_m to output the current hidden state h_m. The hidden state h_m acts as summary of all the tokens observed so far, which effectively makes it a sufficient statistic from a statistical point of view. The matrix $I \in \mathbb{R}^{d_e \times |V|}$ is the input word embedding matrix, where column j contains the embedding for word (token) index j and $d_e \in \mathbb{N}$ is called the word embedding dimensionality. Similarly, the matrix $O \in \mathbb{R}^{d_e \times |V|}$ is called the output word embedding matrix. By eq. (3) and eq. (4), the probability distribution over token w_{m+1} is parametrized as a *softmax* function over the dot products between the hidden state and the output word embeddings for each word in the vocabulary. Therefore, the more similar an output word embedding vector O_v is to the hidden state vector h_m (e.g. the smaller the angle between the two vectors) the higher the probability assigned to token v.

Unlike the n-gram models discussed earlier, the RNNLM does not parametrize a separate probability value for every possible combination of tokens. Instead it embeds words into real-valued vectors using the word embedding matrices, thereby allowing the rest of the model to use the same set of parameters for all words observed. This was the key innovation behind the so-called neural network language model proposed by Bengio et al. (2003), and it is used by the RNNLM and its extensions, which gained state-of-the-art performance on several machine learning tasks (Mikolov et al., 2010; Jozefowicz et al., 2016; Devlin et al., 2018). In addition to this, by using a RNN to compute the hidden state which parametrizes the output distribution, the RNNLM can potentially capture an unlimited amount of context (unlike both n-gram models and the earlier neural network language models). This is also the main motivation for the generative models we will discuss later. The graphical model is illustrated in Figure 3.[2]

[2]Although the classic RNNLM follow the probabilistic graphical model in Figure 3, later models such as the one

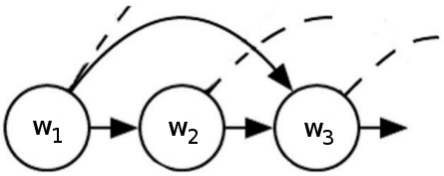

Figure 3: Probabilistic graphical model for a recurrent neural network language (RNNLM) model.

The model parameters are learned by maximum likelihood. However, unlike the n-gram models discussed earlier, there exists no closed-form solution. Therefore, the parameters are usually learned using stochastic gradient descent on the training set. Let $\{\mathbf{w}^i\}_{i=1}^I$ be the training dataset. An example sequence $\mathbf{w}^i$ is sampled at random, and the parameters are updated:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log P_\theta(w_1^i, \ldots, w_{M_i}^i),$$

where $\alpha > 0$ is the learning rate and M_i is the length of sequence i. In practice, any first-order optimization method can be used. For example, the method Adam developed by Kingma and Ba (2015) tends to work well and it is therefore used in all experiments presented in the first part of this thesis. For large models, in practice, the gradient w.r.t. each parameter can be computed efficiently using graphics processing units (GPUs) and parallel computing in combination with the *backpropagation algorithm*, a type of dynamic programming (Goodfellow et al., 2016).

There exists different parametrizations of the function f. One of the simplest and most popular parametrizations is the hyperbolic tangent one-layer neural network:

$$f(h_{m-1}, I_{w_m}) = \tanh(H^i I_{w_m} + H h_{m-1}), \tag{6}$$

where $H \in \mathbb{R}^{d_h \times d_h}$ and $H^i \in \mathbb{R}^{d_h \times d_e}$ are its parameters. Usually a constant, called the bias or intercept, is also added before applying the hyperbolic tangent transformation, but to keep the notation simple we will omit this. Another popular variant is the Gated Recurrent Unit (GRU) proposed by Cho et al. (2014):

$$r_m = \sigma(I_{w_m}^r + H_r h_{m-1}), \qquad \textit{(reset gate)} \tag{7}$$

$$u_m = \sigma(I_{w_m}^u + H_u h_{m-1}), \qquad \textit{(update gate)} \tag{8}$$

$$\bar{h}_m = \tanh(H_i I_{w_m} + H(r_m \cdot h_{m-1})), \qquad \textit{(candidate update)} \tag{9}$$

$$h_m = (1 - u_m) \cdot h_{m-1} + u_m \cdot \bar{h}_m, \qquad \textit{(update)}, \tag{10}$$

where $\cdot$ is the element-wise product and σ is the element-wise logistic function:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \tag{11}$$

proposed by Devlin et al. (2018) follow a different probabilistic graphical model.

and where $I, I^r, I^u \in \mathbb{R}^{d_h \times |V|}$, $H, H_r, H_u \in \mathbb{R}^{d_h \times d_h}$ and $H^i \in \mathbb{R}^{d_h \times d_e}$ are the parameters. The motivation for this parametrization is that the *reset gate* and *update gate* equations control whether or not the model reuses the previous hidden state when computing the current hidden state. If the previous state is useless (i.e. its value will not help determine future tokens in the sequence), r_m should be close to zero and the candidate update will be based mainly on the current input w_m. If the previous state is useful (i.e. h_{m-1} may help predict future tokens in the sequence) then r_m should not be zero. If the previous state h_{m-1} is useful, but the current input is useless, then the update gate should set u_m to zero ensuring that minimal information is stored from the current input. In fact, when u_m is close to zero the update is linear and this helps propagate the gradients in the training procedure. This parametrization appears to be superior to the hyperbolic tangent one-layer neural network across several machine learning problems (Greff et al., 2017). A third, also very popular, parametrization is the Long-Term Short-Term Unit (LSTM) (Hochreiter and Schmidhuber, 1997):

$$i_m = \sigma(I^r_{w_m} + H^{ih} h_{m-1} + H^{ic} c_{m-1}), \tag{12}$$

$$f_m = \sigma(I^f_{w_m} + H^{fh} h_{m-1} + H^{fc} c_{m-1}), \tag{13}$$

$$c_m = f_m c_{m-1} + i_m \text{tanh}(I^c_{w_m} + H^{ch} h_{m-1}), \tag{14}$$

$$o_m = \sigma(I^o_{w_m} + H^{oh} h_{m-1} + H^{oc} c_m), \tag{15}$$

$$h_m = o_m \text{tanh}(c_m), \tag{16}$$

where $h_m, c_m \in \mathbb{R}^{d_h}$, for $m = 1, \ldots, M$, are real-valued vectors, $I^r, I^f, I^c, I^o \in \mathbb{R}^{d_h \times |V|}$ and $H^{ih}, H^{ic}, H^{fh}, H^{fc}, H^{ch}, H^{oh}, H^{oc} \in \mathbb{R}^{d_h \times d_h}$ are the parameters. The variables c_m and h_m can be folded into a single vector by concatenation and rewritten as a hidden state update function. The motivation behind the LSTM parametrization is similar to that of the GRU parametrization. In practice, the LSTM unit appears to yield slightly more stable training compared to the GRU unit, although in terms of performance they appear to perform equally well (Greff et al., 2017). For more details, the reader may also refer to Lipton et al. (2015).

2.1.3 Latent Variable Models

Many probabilistic graphical models also contain latent (hidden) stochastic variables, i.e. stochastic variables which are not observed in the actual data. Two important classes of such models are the hidden Markov models (HMMs) and Kalman filters (also known as linear state space models). These models posit that there exists a sequence of latent stochastic variables, with precisely one latent stochastic variable for each observed token, which explains all the dependencies (e.g. correlations) between the observed tokens. Importantly, the latent stochastic variables obey the Markov property: each latent stochastic variable depends only on the previous latent stochastic variable.

This is similar to the bigram model discussed earlier. It is instructive to understanding the HMM and Kalman filter, as well as the role that latent variables may play in probabilistic graphical models. Therefore, we continue by giving a formal definition for these two models.

As before, let $w_1, \ldots, w_M$ be a sequence of discrete variables, such that $w_m \in V^w$ for $m = 1, \ldots, M$ for a vocabulary V^w. Let $s_1, \ldots, s_M$ be a sequence of discrete latent variables, such that $s_m \in V^s$ for $m = 1, \ldots, M$ for a discrete set V^s. The HMM, with parameters θ, factorizes the probability over variables as:

$$P_\theta(w_1, \ldots, w_M, s_1, \ldots, s_M) = P_\theta(s_1) \prod_{m=2}^{M} P_\theta(s_m|s_{m-1}) \prod_{m=1}^{M} P_\theta(w_m|s_m), \tag{17}$$

$$= \theta^0_{s_1} \prod_{m=2}^{M} \theta^s_{s_m, s_{m-1}} \prod_{m=1}^{M} \theta^w_{w_m, s_m}, \tag{18}$$

where $\theta^0 \in \mathbb{R}^{|V^s|}$, $\theta^s \in \mathbb{R}^{|V^s| \times |V^s|}$ and $\theta^w \in \mathbb{R}^{|V^w| \times |V^s|}$ are non-negative parameters, which define probability distributions. The corresponding graphical model is shown in Figure 4. It is straightforward to derive that the observed tokens are independent conditioned on the latent variables: $w_m \perp\!\!\!\perp w_{m'}|s_m, \ldots, s'_m$ for $m' \neq m$.

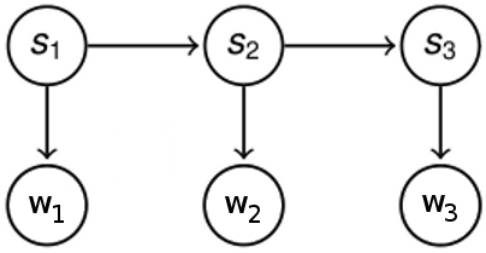

Figure 4: Probabilistic graphical model for hidden Markov model and Kalman filter model.

Next, we describe a variant of the Kalman filter, where the observed sequence is discrete. Let $w_1, \ldots, w_M$ be a sequence of discrete variables, such that $w_m \in V^w$ for $m = 1, \ldots, M$ for a vocabulary V^w. Let $\mathbf{s}_1, \ldots, \mathbf{s}_M$ be a sequence of continuous (real-world) latent variables, distributed according to a normal distribution, such that $\mathbf{s}_m \in \mathbb{R}^d$ for $m = 1, \ldots, M$ and $d \in \mathbb{N}$. The Kalman filter assumes that the latent stochastic variables define a trajectory in a continuous space, which describes the observations. The Kalman filter, with parameters θ, factorizes the probability over variables as:

$$P_\theta(w_1, \ldots, w_M, \mathbf{s}_1, \ldots, \mathbf{s}_M) = P_\theta(\mathbf{s}_1) \prod_{m=2}^{M} P_\theta(\mathbf{s}_m|\mathbf{s}_{m-1}) \prod_{m=1}^{M} P_\theta(w_m|\mathbf{s}_m), \tag{19}$$

$$= \mathcal{N}_{\mathbf{s}_1}(\boldsymbol{\theta}^0_\mu, \theta^s_\Sigma) \prod_{m=2}^{M} \mathcal{N}_{\mathbf{s}_m}(\theta^s_\mu \mathbf{s}_{m-1}, \theta^s_\Sigma) \prod_{m=1}^{M} \frac{\exp({\theta^w_{w_m}}^{\mathrm{T}} \mathbf{s}_m)}{\sum_{w'} \exp({\theta^w_{w'}}^{\mathrm{T}} \mathbf{s}_m)} \tag{20}$$

where $\mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}, \Sigma)$ is the probability of variable $\mathbf{x}$ under the multivariate normal distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$. The parameters defining the generation process over latent stochastic variables are $\boldsymbol{\theta}^0_\mu \in \mathbb{R}^d$ and $\theta^s_\mu, \theta^s_\Sigma \in \mathbb{R}^{d \times d}$. The parameter defining the generation process over observed variables is $\theta^w \in \mathbb{R}^{d \times |V^w|}$, used in a similar way to the RNN parameter in eq. (3). The graphical model is the same as the HMM model, shown in Figure 4.

For the HMM, for small vocabularies V^s and V^w the model parameters may be learned using the stochastic gradient descent procedure described earlier by simply summing out the latent variables. For the Kalman filter, as well as the HMM with large vocabularies, the exact gradient updates are generally intractable and instead other procedures must be used. Two such training procedures are the expectation-maximization algorithm (EM), and the variational learning procedure (Friedman et al., 2001; Bishop, 2006). We will describe the variational learning procedure, since it will be used later in the thesis. The variational learning procedure assumes that a posterior distribution $Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M)$ is estimated, with parameters ψ, which approximates $P_\theta(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M)$ by a multivariate normal distribution. It utilizes a lower-bound on the log-likelihood based on Jensen's inequality:

$$\log P_\theta(w_1, \ldots, w_M) = \log \sum_{\mathbf{s}_1, \ldots, \mathbf{s}_M} P_\theta(w_1, \ldots, w_M, \mathbf{s}_1, \ldots, \mathbf{s}_M) \tag{21}$$

$$= \log \sum_{\mathbf{s}_1, \ldots, \mathbf{s}_M} Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M) \frac{P_\theta(w_1, \ldots, w_M, \mathbf{s}_1, \ldots, \mathbf{s}_M)}{Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M)} \tag{22}$$

$$\geq \sum_{\mathbf{s}_1, \ldots, \mathbf{s}_M} Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M) \log \left(\frac{P_\theta(w_1, \ldots, w_M, \mathbf{s}_1, \ldots, \mathbf{s}_M)}{Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M)} \right) \tag{23}$$

$$= \sum_{\mathbf{s}_1, \ldots, \mathbf{s}_M} Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M) \log \left(\frac{P_\theta(\mathbf{s}_1, \ldots, \mathbf{s}_M) P_\theta(w_1, \ldots, w_M | \mathbf{s}_1, \ldots, \mathbf{s}_M)}{Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M)} \right) \tag{24}$$

$$= \sum_{\mathbf{s}_1, \ldots, \mathbf{s}_M} Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M) \log P_\theta(w_1, \ldots, w_M | \mathbf{s}_1, \ldots, \mathbf{s}_M)$$
$$- \sum_{\mathbf{s}_1, \ldots, \mathbf{s}_M} Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M) \log \left(\frac{Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M)}{P_\theta(\mathbf{s}_1, \ldots, \mathbf{s}_M)} \right) \tag{25}$$

$$= \mathbb{E}_{\mathbf{s}_1, \ldots, \mathbf{s}_M \sim Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M)} \left[\log P_\theta(w_1, \ldots, w_M | \mathbf{s}_1, \ldots, \mathbf{s}_M) \right]$$
$$- \mathrm{KL} \left[Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M) || P_\theta(\mathbf{s}_1, \ldots, \mathbf{s}_M) \right], \tag{26}$$

where $\mathbb{E}_{x \sim P}(f(x))$ is the expectation of function $f(x)$, with x distributed according to P, and $\mathrm{KL}[Q||P]$ is the Kullback-Leibler divergence between distribution Q and P. The distribution $Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M)$ depends on $w_1, \ldots, w_M$. Given a set of data examples, it is possible to maintain a Q distribution with separate parameters ψ over each example. A popular variant of this approach is known as mean-field variational Bayes (Beal, 2003). A more recent approach used in the neural network literature for continuous latent stochastic variables, is to have a neural

network parametrize the posterior, where all data examples share the same parameters (Kingma and Welling, 2014; Rezende et al., 2014). Here, the approximate posterior factorizes as:

$$Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M) = \prod_{m=1}^{M} Q_\psi(\mathbf{s}_m | w_1, \ldots, w_M) \tag{27}$$

$$= \prod_{m=1}^{M} \mathcal{N}_{\mathbf{s}_m}(\mu_m^\psi(w_1, \ldots, w_M), \theta_\Sigma^\psi(w_1, \ldots, w_M)), \tag{28}$$

where $\mu_m^\psi \in \mathbb{R}^d$ and $\theta_\Sigma^\psi \in \mathbb{R}^{d \times d}$ are functions of $w_1, \ldots, w_M$, defined by the approximate posterior parameters ψ, and where θ_Σ^ψ is a positive diagonal matrix. The functions μ_m^ψ and θ_Σ^ψ are typically parametrized as neural networks. The procedure now requires a re-parametrization in order to obtain samples $(\mathbf{s}_1, \ldots, \mathbf{s}_M) \sim Q_\psi(\mathbf{s}_1, \ldots, \mathbf{s}_M | w_1, \ldots, w_M)$. Let $\epsilon_m \sim \mathcal{N}(0, 1)$, for $m = 1, \ldots, M$ (i.e. a sample from the multivariate normal distribution with zero mean, identity covariance matrix and dimensionality d). It is then possible to rewrite $\mathbf{s}_m$ as:

$$\mathbf{s}_m = f_m(\epsilon_m, w_1, \ldots, w_M) = \mu_m^\psi(w_1, \ldots, w_M) + \sqrt{\text{diag}(\theta_\Sigma^\psi(w_1, \ldots, w_M))}\epsilon_m, \tag{29}$$

where $\sqrt{\text{diag}(\theta_\Sigma^\psi(w_1, \ldots, w_M))}$ is a diagonal matrix with diagonal elements equal to the square roots of the diagonal elements in $\theta_\Sigma^\psi(w_1, \ldots, w_M)$. This re-parametrization method allows the taking of gradients w.r.t. parameters ψ. Based on these gradients, the training procedure can use approximate stochastic gradient descent to learn the model parameters. As before, let $\{\mathbf{w}^i\}_i$ be the training dataset. An example i is sampled together with $\epsilon_1^i, \ldots \epsilon_M^i \sim \mathcal{N}(0, 1)$, and the parameters are updated by:

$$\begin{aligned}
\theta \leftarrow \theta &+ \alpha \nabla_\theta \log P_\theta(w_1^i, \ldots, w_M^i | \mathbf{s}_1^i, \ldots, \mathbf{s}_M^i) \\
&- \alpha \nabla_\theta \log \left(\frac{Q_\psi(\mathbf{s}_1^i, \ldots, \mathbf{s}_M^i | w_1^i, \ldots, w_M^i)}{P_\theta(\mathbf{s}_1^i, \ldots, \mathbf{s}_M^i)} \right) \\
\psi \leftarrow \psi &+ \alpha \nabla_\psi \log P_\theta(w_1^i, \ldots, w_M^i | \mathbf{s}_1^i, \ldots, \mathbf{s}_M^i) \\
&- \alpha \nabla_\psi \log \left(\frac{Q_\psi(\mathbf{s}_1^i, \ldots, \mathbf{s}_M^i | w_1^i, \ldots, w_M^i)}{P_\theta(\mathbf{s}_1^i, \ldots, \mathbf{s}_M^i)} \right),
\end{aligned}$$

where $\mathbf{s}_m^i = f_m(\epsilon_m^i, w_1^i, \ldots, w_M^i)$ for $m = 1, \ldots, M$. It is straightforward to compute the gradients w.r.t. θ, and since $\mathbf{s}_m^i$ have been re-parametrized in terms of ϵ_m^i, it is also straightforward to compute the gradients w.r.t. ψ. Furthermore, it is possible to compute the gradient of the exact Kullback-Leibler divergence, which corresponds to the negative term in both equations. For more details, see Kingma and Welling (2014) and Rezende et al. (2014). See also Jordan et al. (1999).

2.1.4 Learning Word, Phrase and Sentence Embeddings with Probabilistic Generative Models

Much of the work presented later in this thesis builds upon earlier work for learning distributed embeddings for linguistic units, such as words, phrases and sentences. In this section, we will introduce some of this work from the point of view of probabilistic graphical models.

The idea of learning distributed embeddings of linguistic units is that each linguistic unit can be mapped into a real-valued, distributed vector representing its semantic and syntactic components. For example, if two linguistic units are close to each other in this vector space, then it may be likely that they have similar semantic or syntactic components (e.g. topic information). An early and popular method for learning distributed word representations is Latent Semantic Analysis (LSA) (Deerwester et al., 1990; Landauer et al., 1998a,b). See Ferrone and Zanzotto (2017), Li and Yang (2018) and Camacho-Collados and Pilehvar (2018) for an overview of many different approaches.

One recent and widely used approach is the Skip-Gram model (Mikolov et al., 2013a,b). Let $\{\mathbf{w}^i\}_{i=1}^I$ be a set of I example sequences of word tokens, called the training dataset, and assume that each word token comes from the vocabulary V. In many real-world applications, these might be extracted from a large corpus of news articles or Wikipedia articles. The Skip-Gram model aims to learn representations of words, which predict their surrounding words (also called context words). This approach is motivated by the *distributional hypothesis*, which states that words which occur in the same contexts tend to have similar meanings (Harris, 1954). During training, the Skip-Gram model will sample a sequence at uniform random $\mathbf{w}^i$. Then, it will sample a word pair at uniform random from this sequence, $w_t, w_{t'} \in \mathbf{w}^i$, under the condition that the two words are within c distance of each other (i.e. $|t - t'| \leq c$). The parameter c is called the training context and is usually set somewhere in the range between 3 and 12. Following this, the Skip-Gram model predicts word $w_{t'}$ conditioned on word w_t by:

$$P_\theta(w_{t'}|w_t) = \frac{\exp(I_{w_{t'}}^T I_{w_t})}{\exp(\sum_{w \in V} I_w^T I_{w_t})}, \tag{30}$$

with word embedding parameters $\theta = I \in \mathbb{R}^{|V| \times d_e}$ and word embedding dimensionality $d_e \in \mathbb{N}$. These word embedding parameters represent the mapping from a word (e.g. a word index) to its corresponding real-valued, distributed vector representation. The simplest variant of the model updates its parameters by maximizing the log-likelihood for that particular sample with stochastic gradient descent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log P_\theta(w_{t'}|w_t). \tag{31}$$

The Skip-Gram model can be interpreted as a probabilistic graphical model when conditioned on a given word. Conditioned on one observed word (i.e. an observed random variable taking values in

Table 1: Examples of the closest tokens given by Skip-Gram model trained on 30 billion training words. This table was adapted from Mikolov et al. (2013b, p. 8).

Query Token	Redmond	Havel	ninjutsu	graffiti
Closest Tokens	Redmond Wash.	Vaclav Havel	ninja	spray paint
	Redmond Washington	president Vaclav Havel	martial arts	grafitti
	Microsoft	Velvet Revolution	swordsmanship	taggers

the set V), the probabilistic graphical model predicts the set of surrounding words independently (i.e. unobserved random variables taking values in the set V). This is illustrated in Figure 5.[3]

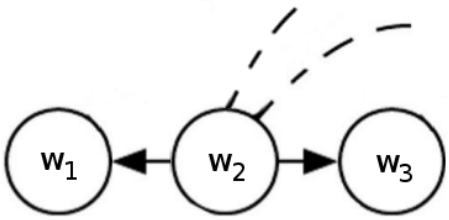

Figure 5: Example of Skip-Gram model as a probabilistic directed graphical model. Conditioned on word w_2 the model aims to predict the surrounding words: w_1, w_3, w_4 and so on. The dashed lines indicate arrows to words outside the diagram.

A simple extension of the Skip-Gram model, called the Skip-Phrase model, enables the learning of distributed representations for phrases such as *New York* and *Air Canada* (Mikolov et al., 2013b). In this case, frequently co-occuring tokens are mapped together to form a single token. For example, since *New* and *York* co-occur together frequently, the Skip-Phrase model might replace them with the combined token *New_York*. Example of the word and phrase embeddings learned by the Skip-Phrase model are shown in Table 1.

The Skip-Gram model belongs to a broader class of models known as Word2Vec word embedding models. These models have achieved success on many natural language processing tasks due to their performance and computational efficiency. For this reason, we will use them in some of the experiments in this thesis. Much work has been done in the area of learning distributed representations for words. A very related model based on co-occurence statistics is the Glove model (Pennington et al., 2014). Other relevant work are Gaussian word embeddings (Vilnis and McCal-

[3]The interpretation illustrated in Figure 5 only applies when the model is conditioned on one word. In general, since any neighbouring pair of words w_t and $w_{t'}$ are used to predict each other, it is not possible to rewrite the model into the form required by eq. (1). This means that is not possible to represent the full model as a probabilistic directed graphical model.

lum, 2015) and contextualized word embeddings (Peters et al., 2018; McCann et al., 2017). See also Li and Yang (2018) and Camacho-Collados and Pilehvar (2018).

The models discussed so far are capable of representing words and phrases. However, there exists also various methods and models for learning distributed sentence representations. Though it is beyond the scope of this thesis to discuss these in detail, it is instructive to discuss at least one model here.

The Skip-Thought Vectors model (Kiros et al., 2015) is one type of neural network, which learns to embed sentences into real-valued, distributed vectors. Analogous to the Skip-Gram model, this model aims to learn the sentence embeddings by predicting neighbouring sentences. Let $\{(\mathbf{w}_p^i, \mathbf{w}^i, \mathbf{w}_n^i)\}_{i=1}^I$ be a set of I example triples, called the training dataset. For each example i, let $\mathbf{w}_p^i$, $\mathbf{w}^i$ and $\mathbf{w}_n^i$ represent the sequence of word tokens in three consecutive sentences inside a document. As before, assume that each word token comes from the vocabulary V. Conditioned on a sentence $\mathbf{w}^i$, the Skip-Thought Vectors model predicts the previous sentence words ($\mathbf{w}_p^i$) and the next sentence words ($\mathbf{w}_n^i$) independently:

$$P_\theta(\mathbf{w}_p^i, \mathbf{w}_n^i | \mathbf{w}^i) = P_\theta(\mathbf{w}_p^i | \mathbf{w}^i) P_\theta(\mathbf{w}_n^i | \mathbf{w}^i), \tag{32}$$

where the probability distributions on the right-hand side are given by:

$$P_\theta(\mathbf{w}_p^i | \mathbf{w}^i) = P_\theta(w_{p,1}^i | \mathbf{w}^i) \prod_{m=2}^{M_{p,i}} P_\theta(w_{p,m}^i | \mathbf{w}^i, w_{p,1}^i, \ldots, w_{p,m-1}^i), \tag{33}$$

$$P_\theta(\mathbf{w}_n^i | \mathbf{w}^i) = P_\theta(w_{n,1}^i | \mathbf{w}^i) \prod_{m=2}^{M_{n,i}} P_\theta(w_{n,m}^i | \mathbf{w}^i, w_{p,1}^i, \ldots, w_{n,m-1}^i), \tag{34}$$

and where sentence $\mathbf{w}_p^i$ contains $M_{p,i}$ words, sentence $\mathbf{w}_n^i$ contains $M_{n,i}$ words, and where θ are the model parameters. The probability distributions given in eq. (33) and eq. (34) are parametrized as variants of the RNNLM with the GRU hidden state update function, but where the token word predictions are excluded for the conditioning sentence $\mathbf{w}^i$. The model is illustrated in Figure 6.

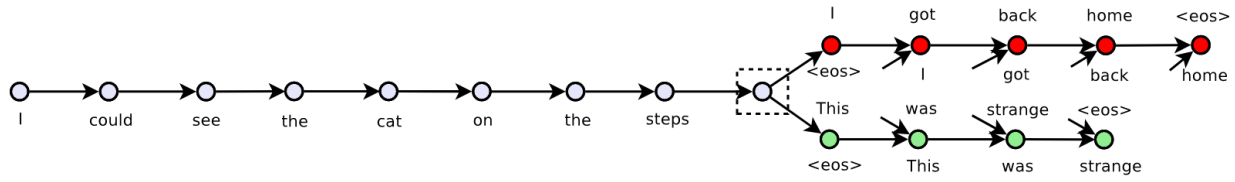

Figure 6: Illustration of the Skip-Thought Vectors model. Illustration taken from Kiros et al. (2015, p. 2)

Similar to the Skip-Gram model, the training dataset for the Skip-Thought Vector model might be extracted from a large corpus of news articles or Wikipedia articles. However, unlike the Skip-

Gram model, the structure of the training dataset is a triple of three sentences. When Kiros et al. (2015) proposed the model, it was trained on the Book Corpus dataset (Zhu et al., 2015).

To illustrate the utility of the Skip-Thought Vector model to learn sentence embeddings, Table 2 shows a list of example query sentences and their nearest neighbours from the Skip-Thought Vectors model trained on the Book Corpus dataset. Such sentence embeddings could potentially be a powerful tool for building data-driven dialogue systems.

One caveat, which is important to mention, is that it contestable how much learned sentence representations, such as those learned by the Skip-Thought Vectors model, capture higher-level sentence structure (such as word order and lexical dependencies). For example, Arora et al. (2017) demonstrate that across several natural language processing tasks the Skip-Thought Vector model, and other models, which presume to learn sentence embeddings capturing word order, can be outperformed by simpler bag-of-words models. Nevertheless, the field is constantly in movement and it is likely that new approaches, such as those proposed by Radford et al. (2018) and Devlin et al. (2018), may be capturing higher-level sentence structure.

Table 2: Examples of query sentences and their nearest sentences of the Skip-Thought Vectors model trained on the Book Corpus dataset (Zhu et al., 2015). This table was extracted from Kiros et al. (2015, p. 3).

Query and nearest sentence
he ran his hand inside his coat , double-checking that the unopened letter was still there .
he slipped his hand between his coat and his shirt , where the folded copies lay in a brown envelope .
im sure youll have a glamorous evening , she said , giving an exaggerated wink .
im really glad you came to the party tonight , he said , turning to her .
although she could tell he had n't been too invested in any of their other chitchat , he seemed genuinely curious about this .
although he had n't been following her career with a microscope , he 'd definitely taken notice of her appearances .
if he had a weapon , he could maybe take out their last imp , and then beat up errol and vanessa .
if he could ram them from behind , send them sailing over the far side of the levee , he had a chance of stopping them .
then , with a stroke of luck , they saw the pair head together towards the portaloos .
then , from out back of the house , they heard a horse scream probably in answer to a pair of sharp spurs digging deep into its flanks .
" i 'll take care of it , " goodman said , taking the phonebook .
" i 'll do that , " julia said , coming in .
he finished rolling up scrolls and , placing them to one side , began the more urgent task of finding ale and tankards .
he righted the table , set the candle on a piece of broken plate , and reached for his flint , steel , a

2.2 Reinforcement Learning

The second part of this thesis investigates building a real-world reinforcement learning dialogue system. In this section, we will provide a brief introduction to the main reinforcement learning theory required to understand that system.

Reinforcement learning is a machine learning framework dealing with sequential decision making. In this framework, a machine learning system is considered an agent, which takes a sequence of actions in an environment in order to maximize an objective function. For example, as we will discuss later, reinforcement learning may be applied to optimize the actions (e.g. responses) taken by a goal-driven dialogue system.

2.2.1 Markov Decision Process

One concept central to reinforcement learning is the Markov decision process (MDP). An MDP is a tuple $\langle S, A, P, R, \gamma \rangle$, where S is a set of states, A is a set of actions, P is a state transition probability function, $R(s, a) \in [0, r_{\text{max}}]$ is a reward function, with $r_{\text{max}} > 0$, and $\gamma \in (0, 1)$ is the discount factor (Sutton and Barto, 1998). Throughout the thesis, we will assume the formulation of the standard MDP with a finite time horizon. Time is assumed to be a discrete variable starting at time $t = 0$. Furthermore, for simplicity, the set of actions A is assumed to be discrete. For a given time t, the agent is in a state $s_t \in S$. In this state, the agent (e.g. the dialogue system) must take (select) an action $a_t \in A$. After this, the agent receives a reward $r_t = R(s_t, a_t)$ and transitions to a new state $s_{t+1} \in S$ with probability $P(s_{t+1}|s_t, a_t)$.[4] The sequence of states, actions and rewards $(s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$ is called an episode. We assume the terminal state is always reached within a finite number of transitions (steps) $T \in \mathbb{N}$. For simplicity, if the episode terminates at $T' < T$ steps, then we assume that the agent reaches the terminal state at $s_{T'} = s_{\text{terminal}}$ and remains there (i.e. $s_{T'+1} = \cdots = s_{T-1} = s_T = s_{\text{terminal}}$) with all future rewards being zero (i.e. $r_{T'+1} = \cdots = r_{T-1} = r_T = 0$). Under this assumption, we may assume without loss of generality that all episodes have length T.

We will assume that the agent utilizes a stochastic policy π. Given a state $s \in S$ as input, the policy π assigns a probability to each possible action $a \in A$:

$$\pi(a|s) \in [0, 1], \quad \text{s. t.} \sum_{a \in A} \pi(a|s) = 1. \tag{35}$$

[4] To simplify notation, we will assume that at a time $t = 0$ the agent is always in a unique *null state* s_0, which cannot be reached by the agent at any other point in time, and where the agent always takes the *null action* a_0. This in turn allows us to rewrite the initial state distribution as the conditional transition distribution: $P(s_1) = P(s_1|s_0, a_0)$.

The goal of the agent is to learn a policy maximizing the discounted sum of rewards:

$$R = \sum_{t=1}^{T} \gamma^t r_t, \tag{36}$$

which is called the *cumulative return* (or simply the *return*). Here, $\gamma \in (0, 1]$ is the discount factor.

Given a policy π, the *state-value function* V^π is defined as the expected return of the policy π starting in state $s \in S$ and completing the episode:

$$V^\pi(s) = \mathrm{E}_\pi \left[\sum_{t=1}^{T} \gamma^t r_t \mid s_1 = s \right]. \tag{37}$$

The *state-action-value function* Q^π is defined as the expected return of taking action $a \in A$ in state $s \in S$, and then following policy π and completing the episode:

$$Q^\pi(s, a) = \mathrm{E}_\pi \left[\sum_{t=1}^{T} \gamma^t r_t \mid s_1 = s, a_1 = a \right]. \tag{38}$$

A policy π^* is *optimal* if it satisfies $\forall s \in S, a \in A$:

$$V^{\pi^*}(s) = V^*(s) = \max_\pi V^\pi(s). \tag{39}$$

The state-action-value function w.r.t. the optimal policy is:

$$Q^{\pi^*}(s, a) = Q^*(s, a) = \mathrm{E}_{\pi^*} \left[\sum_{t=1}^{T} \gamma^t r_t \mid s_1 = s, a_1 = a \right] \forall s \in S, a \in A. \tag{40}$$

Given Q^*, one may recover the optimal policy as a (discrete) Dirac-delta distribution:

$$\pi^*(s) = \arg\max_{a \in A} Q^*(s, a), \tag{41}$$

$$\pi^*(a|s) = 1_{(\arg\max_{a' \in A} Q^*(s,a') = a)}, \tag{42}$$

where $1_{(\cdot)}$ equals one if the statement $(\cdot)$ is true, and otherwise equals zero. Here, to simplify notation, we have assumed that there is a unique optimal action in each state.[5]

The optimal policy can be found via dynamic programming using the Bellman optimality equations (Bertsekas and Tsitsiklis, 1995; Sutton and Barto, 1998). These equations state that $\forall s \in S, a \in A$ it holds that:

$$V^*(s) = \max_{a \in A} Q^*(s, a), \tag{43}$$

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^*(s') \tag{44}$$

if and only if eq. (39) is satisfied. In problems where the state or action spaces are very large, it may not be possible to solve for these equations directly. In this case, approximate efficient learning algorithms may be used to find a solution. Some popular algorithms include SARSA, Q-learning, REINFORCE and actor-critic methods (Sutton and Barto, 1998).

[5]Here, it should be noted that $\pi^*(s)$ is a deterministic function, which maps a given state to the optimal action.

2.2.2 Tabular Reinforcement Learning with Q-Learning

It is instructive to give a more detailed example of a simplified setting. Let us assume that we are given a MDP, where the state space S and action space A are small discrete sets each containing a few hundred or a few thousand elements. In this case, it is possible to learn a *tabular policy*, which assigns a probability for each possible state and action pair:

$$\pi_\theta(a|s) = \theta_{s,a}, \tag{45}$$

where $\theta \in \mathbb{R}^{|S|\times|A|}$ is a parameter matrix denoting the probability for taking an action $a \in A$ in any given state $s \in S$, such that for all $s \in S$:

$$\sum_{a\in A} \theta_{s,a} = 1 \quad \text{and} \quad \theta_{s,a} \geq 0 \quad \forall a \in A \tag{46}$$

One can learn a tabular policy using the Q-learning algorithm (Watkins, 1989). Let $Q_\psi(s,a)$ be an approximate state-action-value function parametrized by $\psi \in \mathbb{R}^{|S|\times|A|}$:

$$Q_\psi(s,a) = \psi_{s,a}, \tag{47}$$

which represents the approximate expected return of taking action $a \in A$ in state $s \in S$, and then following the policy π_θ until the episode is completed. Given Q_ψ, the policy π_θ may be defined as the softmax function:

$$\pi_\theta(a|s) = \theta_{s,a} = \frac{\exp(\lambda^{-1}Q_\psi(s,a))}{\sum_{a'\in A}\exp(\lambda^{-1}Q_\psi(s,a'))} \tag{48}$$

where $\lambda > 0$ is the temperature parameter. A higher λ will lead to a more uniform stochastic policy (e.g. with more variety of actions taken).

In order to learn an effective policy, Q-learning aims to make the approximate state-action-value function Q_ψ satisfy the same optimality condition as given in eq. (44):

$$Q_\psi(s,a) = R(s,a) + \gamma \sum_{s'\in S} P(s'|s,a)V_\psi(s') \quad \forall s \in S, a \in A, \tag{49}$$

where, following eq. (43), it sets $V_\psi(s') = \max_{a \in A} Q_\psi(s, a)$. This equation can be rewritten as:

$$
\begin{aligned}
Q_\psi(s,a) &= R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_\psi(s') \quad \forall s \in S, a \in A \\
&\Leftrightarrow \\
Q_\psi(s,a) &= R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} Q_\psi(s',a') \quad \forall s \in S, a \in A \\
&\Leftrightarrow \\
Q_\psi(s,a) - R(s,a) - \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} Q_\psi(s',a') &= 0 \quad \forall s \in S, a \in A \\
&\Leftrightarrow \\
\psi_{s,a} - R(s,a) - \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} \psi_{s',a'} &= 0 \quad \forall s \in S, a \in A \\
&\Leftrightarrow \\
\psi_{s,a} - R(s,a) - \gamma \, \mathrm{E}_{s' \sim P(s'|s,a)}[\max_{a' \in A} \psi_{s',a'}] &= 0 \quad \forall s \in S, a \in A \\
&\Leftrightarrow \\
\Big(\psi_{s,a} - R(s,a) - \gamma \, \mathrm{E}_{s' \sim P(s'|s,a)}[\max_{a' \in A} \psi_{s',a'}]\Big)^2 &= 0 \quad \forall s \in S, a \in A,
\end{aligned} \tag{50}
$$

where $\mathrm{E}_{s' \sim P(s'|s,a)}[\cdot]$ denotes the expectation w.r.t. the distribution $P(s'|s,a)$. Finding a set of parameters ψ satisfying this condition can be posed as an iterative optimization problem. Given any state $s \in S$ and action $a \in A$, as well as the corresponding value $r = R(s,a)$ and a sample $s' \sim P(s'|s,a)$, the following gradient can be used to update the parameters ψ:

$$
\begin{aligned}
&\frac{\delta}{\delta \psi_{s,a}} \Big(\psi_{s,a} - \hat{\psi}_{s,a}\Big)^2, \\
&\text{where } \hat{\psi}_{s,a} = r + \gamma \max_{a' \in A} \psi_{s',a'},
\end{aligned} \tag{51}
$$

where $\hat{\psi}_{s,a}$ is assumed to be a constant value. This value is called the *target value*, since $\psi_{s,a}$ gets updated to be closer to it. This leads to the update equation:

$$
\begin{aligned}
\psi_{s,a} &\leftarrow \psi_{s,a} - \alpha(\psi_{s,a} - \hat{\psi}_{s,a}) \\
&= (1-\alpha)\psi_{s,a} + \alpha \hat{\psi}_{s,a}
\end{aligned} \tag{52}
$$

The previous update equation motivates how Q-learning learns the policy π_θ. At first, the policy is initialized to a uniform random policy. This is done by setting $\psi = \mathbf{0}$. Then the learning algorithm repeatedly alternates between two steps. In the first step, the policy π_θ is used to *roll out* an episode. In other words, a sequence of states, actions and rewards $(s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$ of the episode is recorded by having the agent follow the actions according to the policy π_θ. In the

second step, the approximate state-action-value function is updated according to the Q-learning update rule. For every tuple (s_t, a_t, r_t, s_{t+1}) in the episode, the parameters are updated using:

$$Q_\psi(s_t, a_t) \leftarrow (1-\alpha)Q_\psi(s_t, a_t) + \alpha\left(r_t + \gamma \max_{a' \in A} Q_\psi(s_{t+1}, a')\right), \tag{53}$$

where $\alpha > 0$ is the learning rate. Using eq. (47), we can rewrite this as an update w.r.t. ψ:

$$\psi_{s_t,a_t} \leftarrow (1-\alpha)\psi_{s_t,a_t} + \alpha\left(r_t + \gamma \max_{a' \in A} \psi_{s_{t+1},a'}\right), \tag{54}$$

By alternating between the two steps (i.e. rolling out episodes and updating the parameters), an effective policy can be learned. Furthermore, under certain assumptions of the MDP and learning rate, this learning procedure is guaranteed to converge to an optimal policy. See Bertsekas and Tsitsiklis (1995) and Sutton and Barto (1998).

2.2.3 Deep Reinforcement Learning with Q-Learning

At the intersection of deep learning and reinforcement learning lies a class of algorithms known as deep reinforcement learning. These are algorithms that combine the reinforcement learning framework, where an agent takes actions in an environment in order to maximize an objective function, and deep learning models, which help represent policies by parametrizing them as neural networks. For a detailed introduction, the reader should refer to François-Lavet et al. (2018).

One recent and widely-used approach is the deep Q-network model (DQN) introduced by Mnih et al. (2015). Similar to the tabular policy described above, here an approximate state-action-value function is used to parametrize the policy of the agent. However, unlike the tabular policy, the approximate state-action-value function is parametrized as a neural network. Let $Q_\psi(s, a)$ be the neural network with parameters ψ. The network takes as input a state $s \in S$ and an action $a \in A$ and outputs an estimate of the expected return of taking action a in state s, and then following the existing policy π_θ until the episode is terminated. As before, the deep Q-network model learns by alternating between rolling out episodes and updating its model parameters. Given a tuple (s_t, a_t, r_t, s_{t+1}) from an episode observed under policy π_θ, the parameters are updated by minimizing the squared error:

$$\begin{aligned} &||Q_\psi(s_t, a_t) - \hat{Q}_\psi(s_t, a_t)||^2, \\ &\text{where } \hat{Q}_\psi(s_t, a_t) = r_t + \gamma \max_{a'} Q_\psi(s_{t+1}, a'), \end{aligned} \tag{55}$$

where $\hat{Q}_\psi(s_t, a_t)$ is taken to be a constant variable, similar to the tabular example above. For example, the parameters may be updated by stochastic gradient descent:

$$\psi \leftarrow \psi - \alpha \nabla_\psi ||Q_\psi(s_t, a_t) - \hat{Q}_\psi(s_t, a_t)||^2, \tag{56}$$

where $\alpha > 0$ is the learning rate. The policy itself can be defined as the softmax function in eq. (48). Alternatively, the policy can be defined as an ϵ-greedy policy:

$$\pi_\theta(s,a) = (1-\epsilon)1_{(\arg\max_{a' \in A} Q_\psi(s,a')=a)} + \frac{\epsilon}{|A|}, \tag{57}$$

where $\epsilon \in (0,1)$ represents the proportion of actions taken at uniform random.

Deep Q-learning networks and its variants (sometimes referred to as deep Q-learning) have been successfully applied to a variety of tasks, such as game playing (Tesauro, 1995; Mnih et al., 2013), robotic control problems (Gu et al., 2016) and dialogue systems (Zhao and Eskenazi, 2016; Cuayáhuitl, 2017). For more details, the reader is referred to François-Lavet et al. (2018).

2.2.4 Model-based Reinforcement Learning

An important subfield of reinforcement learning is model-based reinforcement learning (Sutton, 1990; Moore and Atkeson, 1993; Peng and Williams, 1993). In model-based reinforcement learning, an explicit model of the environment is learned together with the policy. For example, an estimate of the state transition probability function may be learned simultaneously with the policy.

Model-based reinforcement learning is utilized in the second part of this thesis. For this reason, this section will provide the reader with a brief introduction. For a more detailed introduction, the reader may refer to François-Lavet et al. (2018, Ch. 6), Polydoros and Nalpantidis (2017) and Kaelbling et al. (1996).

Consider the following example, where we aim to learn an efficient policy for the MDP $\langle S, A, P, R, \gamma \rangle$, but without having access to the transition distribution P or reward function R. However, suppose that we still have access to the set of states and actions and the discount factor γ. Furthermore, suppose we have collected a dataset $D = \{(s^i, a^i, r^i, s'^i)\}_{i=1}^I$, where each example i consists of a state $s^i \in S$, where the system took action $a^i \in A$, received reward $r^i \in \mathbb{R}$ and transitioned to the new state $s'^i \in S$. We can use the dataset D to estimate an approximate transition distribution P_{Approx}:

$$P_{\text{Approx}}(s'|s,a) \approx P(s'|s,a) \quad \forall s, s' \in S, a \in A.$$

For example, P_{Approx} can be learned by counting co-occurrences in D (Moore and Atkeson, 1993):

$$P_{\text{Approx}}(s'|s,a) = \frac{\text{Count}(s,a,s')}{\text{Count}(s,a,\cdot)}, \tag{58}$$

where $\text{Count}(s,a,s')$ is the observation count for (s,a,s') and $\text{Count}(s,a,\cdot) = \sum_{s'} \text{Count}(s,a,s')$ is the observation count for (s,a) followed by any other state in D. In addition, we can estimate an approximate reward function R_{Approx}:

$$R_{\text{Approx}}(s,a) \approx R(s,a) \quad \forall s \in S, a \in A,$$

The approximate reward function can be learned by averaging the observed rewards in D:

$$R_{\text{Approx}}(s,a) = \frac{\sum_{i=1}^{I} 1_{(s^i=s,a^i=a)} r^i}{\text{Count}(s,a,\cdot)}, \tag{59}$$

Given P_{Approx} and R_{Approx}, we can construct an approximate MDP $\langle S, A, P_{\text{Approx}}, R_{\text{Approx}}, \gamma \rangle$. The approximate MDP can then be used to run simulations by drawing samples from the distributions P_{Approx} and R_{Approx}. By rolling out episodes (i.e. simulating episodes) in the approximate MDP, we may be able to learn an effective policy π. For example, we can apply the Q-learning algorithm described earlier to learn an approximate state-action-value function and a corresponding policy. Under appropriate assumptions, the policy we recover will be optimal w.r.t. the approximate MDP if it satisfies the Bellman equations of the approximate MDP. Specifically, $\forall s \in S, a \in A$ it must hold that:

$$V_{\text{Approx}}(s) = \max_{a \in A} Q_{\text{Approx}}(s,a), \tag{60}$$

$$Q_{\text{Approx}}(s,a) = R_{\text{Approx}}(s,a) + \gamma \sum_{s' \in S} P_{\text{Approx}}(s'|s,a) V_{\text{Approx}}(s'), \tag{61}$$

where V_{Approx} and Q_{Approx} are the the state-value and state-action-value functions associated with the approximate MDP. The hope is that if $P_{\text{Approx}}(s'|s,a) \approx P(s'|s,a)$ and $R_{\text{Approx}}(s,a) \approx R(s,a)$, then the recovered state-action-value function satisfies $Q_{\text{Approx}}(s,a) \approx Q(s',a') \; \forall s \in S, a \in A$ for policy π. This would imply that the policy derived from $Q_{\text{Approx}}(s,a)$ is close to the optimal policy.

The algorithm described above is very similar to the well-known Dyna-Q algorithm (Sutton, 1990; Kuvayev and Sutton, 1996). However, in the Dyna-Q algorithm the model of the environment (i.e. P_{Approx} and R_{Approx}) and the policy are learned simultaneously.

2.3 Dialogue Systems

Dialogue systems, also known as interactive conversational agents, virtual agents or sometimes chatbots, are computer programs which interact with humans through either written natural language or spoken natural language. They have been applied to a variety of applications ranging from customer and technical support services to language learning tools and entertainment (Young et al., 2013; Shawar and Atwell, 2007).

An important distinction should be made between goal-driven dialogue systems (e.g. technical support services), and non-goal-driven dialogue systems (e.g. chatting for entertainment) (Wallace, 2009; Serban et al., 2017c; Ram et al., 2017; Papaioannou et al., 2017). Both types of dialogue systems have some form of objective. Goal-driven dialogue systems tend to have a well-defined, specific performance measure, which is explicitly related to task completion. Although often non-goal-driven dialogue systems do not have such a specific performance measure, they are usually built to maximize user engagement (e.g. duration or length of the dialogue).

Early dialogue systems were built using rule-based methods. An example of such a system is the famous text program ELIZA, a system based on text parsing rules which aimed to mimic a *Rogerian* psychotherapist by persistently rephrasing statements or asking questions (Weizenbaum, 1966). Similarly, the dialogue system PARRY aimed to mimic the pathological behaviour of a paranoid patient, which it managed to do so well that clinical analysts could not differentiate it from real human patients (Colby, 1981). Later, a more sophisticated rule-based dialogue system, called ALICE, was developed based on AIML (artificial intelligence markup language) templates to produce a responses when given a dialogue history (Wallace, 2009; Shawar and Atwell, 2007).

Following the seminal work on ELIZA and PARRY, researchers started to focus on data-driven systems for goal-driven dialogue. An example is the *How may I help you* system for routing telephone calls developed by (Gorin et al., 1997). Trained on a database of 10,000 spoken utterances, the system would combine speech recognition with natural language understanding to efficiently route telephone calls. An important line of work here were the systems developed for the Airline Travel Information System (ATIS) domain (Pieraccini et al., 1992; Seneff, 1992; Dowding et al., 1993; Miller et al., 1994; Ward and Issar, 1994). The research on these systems helped define many of the fundamental research problems, in particular machine learning problems, which need to be solved in order to construct real-world dialogue systems.

In the 90s, researchers began to formulate dialogue as a sequential decision making problem based on Markov decision processes (Singh et al., 1999; Young et al., 2013; Paek, 2006; Pieraccini et al., 2009). Some of the seminal work here include the NJFun system (Singh et al., 2002) and the Let's Go system (Raux et al., 2006). For a detailed overview of this and some of the later work, please see Lemon and Pietquin (2007), Young et al. (2013) and Chen et al. (2017).

Although this work has pushed the field towards data-driven approaches, modern commercial systems were and are still highly domain-specific and heavily based on hand-crafted features (Singh et al., 2002; Young et al., 2013). That's why a major motivation for developing dialogue systems using probabilistic generative models, is that such models may be trained on large, un-annotated corpora and therefore have the potential to scale to new domains. One of the most prominent examples of modern goal-driven dialogue systems is the emerging line of personal assistants, such as Apple Siri (Wit, 2014), Amazon Alexa (Stone and Soper, 2014), Microsoft Cortana (Bhat and Lone, 2017) and Google Now (Wortham, 2012). Although not much information about these systems is available to the public, it is widely agreed that these systems operate through a modular framework centered around service applications. Every time a user speaks an utterance to such a system, the utterance is routed to an appropriate service (such as a *weather service application* for identifying the user's intention and retrieving the appropriate weather report, or a *alarm clock service application* for identifying the user's intention and changing the appropriate alarm on the device). For a comparison between these systems, please see López et al. (2017). However, there are also goal-driven dialogue systems, which are not personal assistants. One example is the IBM Project Debater system, which is capable of coherently debating complex topics in natural language involving many back-and-forth turns (Debater, 2018; Slonim, 2018). At the other end of the spectrum, there are also prominent examples of non-goal-driven dialogues. Two of such examples are Microsoft Xiaoice (Markoff and Mozur, 2015) and Hugging Face (Dillet, 2016). Both of these two systems are capable of having casual natural language conversations about everyday topics. In contrast to the goal-driven dialogue systems, which focus on and evaluate success by measuring goal completion, these systems focus on engaging the user as much as possible (e.g. by encouraging long conversations and by encouraging users to return frequently).

2.3.1 System Components

As discussed above, there are many different approaches to building dialogue systems. As illustrated in Figure 7, a typical dialogue system can be decomposed into the following components: a speech recognizer, a natural language interpreter, a state tracker, a response generator, a natural language generator and a speech synthesizer. For text-only dialogue systems, the speech recognizer and speech synthesizer would be left out. In general, it is possible to develop or improve all components of the dialogue system using data-driven approaches. Further, it should be noted that the natural language interpreter and natural language generator are fundamental NLP problems with many applications outside the scope of dialogue systems.

In the first part of this thesis, the approach taken is to combine all four components in a single model, which jointly does natural language interpretation, dialogue state tracking and natural lan-

guage generation. In the second part of this thesis, the approach taken is to recompose the dialogue system into a system with two components: a component generating a set of candidate responses, and a component selecting the appropriate candidate response to emit.

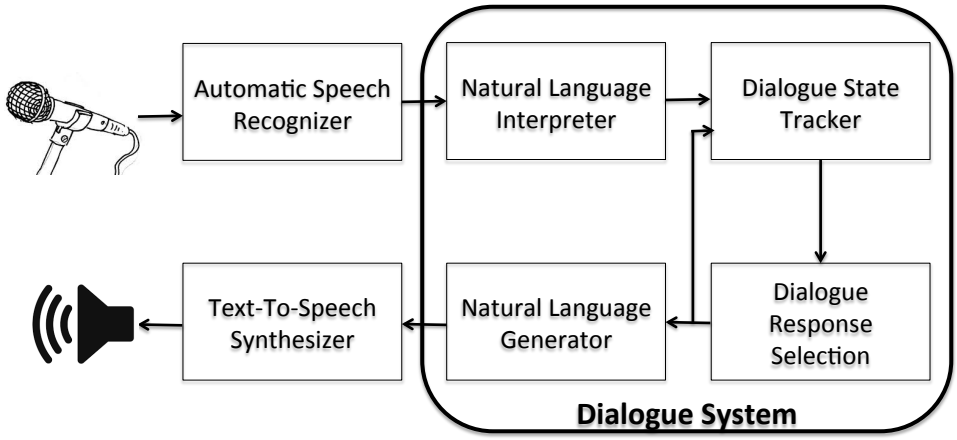

Figure 7: An overview of components in a dialogue system, reproduced from Serban et al. (2018).

2.3.2 System Learning

Several dialogue components can be optimized in a supervised learning framework. As an example, consider the problem of learning a user intent classification model, which is a sub-model of the natural language interpreter. The goal of this classification model is to map the user's utterance to the corresponding intention class. The intent class is called the label (or target), and the conditioned utterances are called the conditioning variables (or input). An example of such a model might be a multinomial logistic regression classifier which, conditioned on a set of features extracted from the speech recognition system output of the user's utterance, assigns a probability to each intention class. In this case, the model parameters θ can be learned by applying gradient

descent to maximize the log-likelihood:

$$\hat{\theta} = \arg\max_{\theta} \sum_{m=1}^{M} P_\theta(y_m|x_m),$$

where $\{x_m, y_m\}_{m=1}^{M}$ is a set of examples pairs, with y_m and x_m being the label and conditioning input respectively. Such types of models have allowed goal driven dialogue systems to make significant progress (Williams et al., 2013). Once trained on a given dataset, these models may be plugged into a deployed dialogue system. For example, the class predicted by the user intent model may be given as input to a dialogue state tracker.

We next discuss two types of data-driven response generation components. The first type deterministically selects the response from a fixed set of possible utterances, and the second type generates a response by computing a posterior probability over all possible utterances. The first type skips the natural language generator in Figure 7, by directly mapping the dialogue history, tracker outputs and external knowledge to a response utterance (Serban et al., 2018):

$$f_\theta : \{\text{dialogue history}, \text{ tracker outputs}, \text{external knowledge}\} \rightarrow \text{ utterance}. \tag{62}$$

Retrieval-based systems, such as the neural network retrieval model proposed by Lowe et al. (2015b) or the information retrieval model proposed by Banchs and Li (2012) are in this category.

The second type of response generation components explicitly compute a full posterior probability distribution over possible system responses at every turn:

$$P_\theta(\text{utterance} \mid \text{dialogue history}, \text{tracker outputs}, \text{external knowledge}).$$

Generative neural network models, the subject of the first part of this thesis, belong to this category. Reinforcement learning systems with stochastic policies, such as the *NJFun system* developed by Singh et al. (2002), also belong to this category. Unfortunately, these systems typically have only a tiny set of hand-crafted system states and actions, in order to make current reinforcement learning algorithms applicable. This critically limits their application area. Thus, as noted by Singh et al. (2002, p.5): "We view the design of an appropriate state space as application-dependent, and a task for a skilled system designer".

2.3.3 System Evaluation

Accurate evaluation of dialogue systems is important for measuring development progress and determining the utility of different dialogue models. However, this is known to be a very difficult problem (Galley et al., 2015; Pietquin and Hastie, 2013; Schatzmann et al., 2005). In general, the evaluation of a dialogue system depends heavily on whether it's a non-goal-driven system or a

goal-driven system. In particular, the evaluation of a goal-driven system depends heavily on the application domain which the system was designed to solve. This is in contrast with typical supervised machine learning problems, where ground truth labels are available and simple statistical metrics (e.g. accuracy, precision, recall and F1-score) may be used.

Evaluation for Goal-driven Systems Goal-driven dialogue systems have primarily been evaluated by their ability to solve their intended task by human participants. Typically a small number of human participants are recruited to evaluate the dialogue system. The participants are instructed to solve a series of specific tasks, and afterwards their success rate (i.e. how often they manage to solve each task) and interaction length (i.e. how long it took to solve each task) are measured. The higher the success rate and the shorter the interaction length, the better the dialogue system is presumed to be. For example the *NJFun system*, which provides users with access to information about fun things to do in New Jersey, was evaluated in this way (Singh et al., 2002). One of the six tasks participants were instructed to complete was stated as: *Task 1. You are bored at home in Morristown on a rainy afternoon. Use NJFun to find a museum to go to.* After completing the tasks, the average success rate was measured across participants. The success rates were then compared across dialogue systems in order to determine the most effective dialogue system. Instead of measuring task success rate, it is also possible to ask participants to directly rate the usefulness of the dialogue system (Kamm, 1995). Unfortunately, there are some problems with both of these two approaches to evaluation. The first problem is that it is expensive and time-consuming to carry out human experiments, which slows down research. The second problem is that the number of participants is limited and the experimental conditions are often not replicable, which introduces variance into the results or, even worse, makes the results irreproducible. The third problem is that the evaluation is fundamentally biased. The recruited participants differ significantly from the real users of the dialogue system, because ultimately they do not care about completing the actual dialogue task (Young et al., 2013). Finally, these approaches cannot directly be applied to non-goal-driven dialogue systems.

Automatic Evaluation Metrics For evaluating a one-turn dialogue system response, researchers have recently proposed to use automatic evaluation metrics adopted from the field of machine translation (Li et al., 2016a; Galley et al., 2015; Sordoni et al., 2015b; Ritter et al., 2011a). Such an approach requires a test set of dialogue contexts and dialogue responses, typically extracted from conversations between humans. For each test example, the dialogue model conditions on the dialogue context and generates a response. The similarity between the generated response and the ground truth response is then estimated, for example by measuring the number of words they have in common or, more generally, the number of n-grams they have in common. Here, an n-gram is

a sequence of n consecutive words. One of the most popular metrics for estimating the similarity between responses is the BLEU metric, which measures the n-gram overlap, for $n = 1, 2, 3, 4$ and also takes into account response length (Papineni et al., 2002). Another popular metric is the METEOR metric, which uses a database called WordNet to take into account semantic similarity between words (Banerjee and Lavie, 2005). Therefore, even if the generated response has no words in common with the ground truth response, METEOR may still yield a non-zero score because some of the words may be related according to WordNet. This approach may seem useful for evaluating machine translation models, but for dialogue systems it is critically flawed as discussed by (Liu et al., 2016). Liu et al. (2016) computed the correlation between automatic evaluation metrics and human annotators across different dialogue systems and tasks, and found the correlations to be very low. The main problem with automatic evaluation metrics seems to be that for most real-world dialogue problems the set of appropriate dialogue responses is huge, and therefore it is unlikely that any words will overlap between the generated response and the ground truth response. Even when there exists an overlap, the overlap is rarely between the topic-related words, but more often between pronouns and punctuation marks.[6] This is highly misleading, because the metric is biased away from favouring responses with topic-related words, which arguably determines the relevance of the response.

In an effort to overcome these issues Liu et al. (2016) propose three metrics based on word embeddings. The first proposed metric is called greedy matching. Given two responses 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:

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

where $\cos(e_w, e_{\hat{w}})$ represents the cosine similarity between e_w and $e_{\hat{w}}$. Since G is asymmetric w.r.t. r and $\hat{r}$, the metric averages the greedy matching scores G for each ordering of r and $\hat{r}$. The greedy scoring metric was originally introduced for intelligent tutoring systems (Rus and Lintean, 2012). It favours generated responses with words, which are similar to the words in the ground truth response. The second metric Liu et al. (2016) propose is called embedding average. This metric computes 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 score the similarity between a ground truth response r and generated response $\hat{r}$, the metric computes the cosine similarity between their respective sentence level embeddings: $\text{EA}(r, \hat{r}) :=$

[6]The BLEU and METEOR metrics consider punctuation marks as separate words.

$\cos(\bar{e}_r, \bar{e}_{\hat{r}})$. The third metric is called vector extrema based on the metric proposed by Forgues et al. (2014). For each response, this metric computes a new vector by taking the largest absolute value among all word embeddings in the response. Afterwards, it computes the cosine similarity between the two vectors. Unfortunately the experiments carried out by Liu et al. (2016) show that all metrics, including their proposed metrics, the BLEU metric and METEOR metric, have at best a low correlation with human evaluations of what constitutes good dialogue responses. This means that they cannot fairly be interpreted as a proxy for human evaluation. However, due to the distributional hypothesis, the embedding metrics can be interpreted as a measuring topic similarity. If the generated response contains words on the same topic as the ground truth response, all embedding-based metrics will yield high scores. For this reason, we will use these metrics to measure topic relevance.[7]

[7]It is possible to remove stop words before calculating these three metrics. This may help the metrics to further emphasize topic relevance. However, this was not done in this thesis.

3 Generative Dialogue Models

This is the first part of the thesis work, which investigates representation learning for generative dialogue models. Conditioned on a sequence of turns from a text-based dialogue, the models proposed here are tasked with generating the next, appropriate response in the dialogue.

In this part, we ask a number of open research questions. The first set of questions we ask is related to model architectures appropriate for dialogues with long-term temporal structure. Which type of model architectures are appropriate for building dialogue systems operating in complex real-world domains? Which type of architectures are capable of incorporating discourse-level context? What effect does modelling discourse-level context have on model performance w.r.t. goal-driven and non-goal-driven dialogue tasks? How does modelling discourse-level context change the generated model responses compared to simpler models, which do not aim to capture discourse-level context? The second set of questions we ask is related to modelling higher level semantic structure. How important is it to model high-level semantic structure? How can high-level semantic structure be modelled while also incorporating the discourse-level context? What are the appropriate structured representations and how can these representations help to facilitate generalization of the model to unseen topics? What are the advantages and disadvantages of modelling high-level semantic structure w.r.t. goal-driven and non-goal-driven dialogue tasks? How do these relate to other forms of composition, such as compositional semantic structure? The third set of questions we ask is related to modelling uncertainty and ambiguity in human language. Which type of architectures are capable of modelling the uncertainty and ambiguity inherent in real-world dialogue settings? How can uncertainty and ambiguity be modelled as latent or observed stochastic processes? Which model architectures are able to generate meaningful and semantically relevant responses when confronted with high amounts of uncertainty?

We aim to answer some of these questions in this chapter. Motivated by these questions, we will propose novel model architectures and learning algorithms. These will be compared to existing models proposed in the literature and then implemented in practice. The model architectures and learning algorithms will then be applied and evaluated for building systems for goal-driven and non-goal-driven dialogue tasks. The quantitative and qualitative evaluation of these model architectures and learning algorithms, as well as the steps leading to their successful implementation, should help to shed light on some of the open research questions.

3.1 Hierarchical Recurrent Encoder-Decoder

3.1.1 Author's Contribution

The work in this section covers the author's work published in the conference publication:

> "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models" by Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville and Joelle Pineau, p. 3776–3784, Association for the Advancement of Artificial Intelligence, 2016.

The conference publication can be accessed at: https://aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11957/12160. [8]

The author of this thesis did the majority of the work related to the development of the models, the experiments and the writing up of the conference publication. The author received help in building and evaluating the models from Alessandro Sordoni. The author received help on writing up the paper from Alessandro Sordoni, Yoshua Bengio, Aaron Courville and Joelle Pineau.

3.1.2 Motivation

The work presented here is in the direction of building end-to-end trainable, non-goal-driven systems based on generative probabilistic models, which can better incorporate long-term discourse-level context for generating responses. Examples of long-term discourse-level context include: the conversation topic, the opinions and statements made by each interlocutor and responses to these made by other interlocutors, the entities and events mentioned by each interlocutor and the acceptance, objections and confusions raised by other interlocutors in response, the points of agreement or disagreement between interlocutors and so on. To this end, this section adapts the Hierarchical Recurrent Encoder-Decoder (HRED) model, originally proposed by Sordoni et al. (2015a) for web query suggestion, to dialogue response generation.

We define the generative dialogue problem as modelling the utterances and interactive structure of the dialogue (i.e. modelling the words and speaker turns in a dialogue). As such, the model we describe next may be viewed as a cognitive system, which has to carry out natural language understanding, reasoning, decision making and natural language generation in order to replicate or emulate the behaviour observed in the training corpus. This approach differs from previous work on learning dialogue systems through interaction with humans (Young et al., 2013; Gasic et al., 2013; Cantrell et al., 2012; Mohan and Laird, 2014), because it learns off-line through examples of human-human dialogues and aims to emulate the dialogues in the training corpus instead of maximizing a task-specific objective function.

[8] The source code for the model described next is available at: https://github.com/julianser/hed-dlg-truncated.

3.1.3 Prior Related Work

Next, we will discuss some of the related work conducted prior to the research in this chapter.

Modelling conversations on micro-blogging websites with generative probabilistic models was first proposed by Ritter et al. (2011a), who view the response generation problem as a translation problem, where a post needs to be translated into a response. Generating responses was found to be considerably more difficult than translating between languages. This was likely due to the wide range of plausible responses and lack of phrase alignment between the post and the response. Nonetheless, Ritter et al. (2011a) found that the statistical machine translation approach was superior to an information retrieval approach.

In the same vein, Shang et al. (2015) propose to use the neural network encoder-decoder framework for generating responses on the micro-blogging website *Weibo* (Sutskever et al., 2014; Cho et al., 2014). They also formulate the problem as conditional generation, where given a post, the model generates a response. Unfortunately, generation using their model has a complexity scaling linearly with the number of dialogue turns.[9] Following this, Sordoni et al. (2015b) propose to generate responses to posts on *Twitter* using a new way to incorporate dialogue context. Sordoni et al. (2015b) concatenate three consecutive Twitter messages, representing a short conversation between two users, and define the problem as predicting each word in the conversation given all preceding words. They encode a bag-of-words context representation with a multilayer neural network and generate a response with a standard RNN. They then combine their generative model with a machine translation system, and show that the hybrid system outperforms the machine translation system proposed by Ritter et al. (2011a). Since their approach for incorporating discourse-level context is based on a bag-of-words representation, it is likely that the model will be sub-optimal for many real-world dialogue applications.

It is also worth mentioning related work dealing with movie scripts and movie subtitles. To the best of our knowledge, Banchs and Li (2012) were the first to suggest using movie scripts to build dialogue systems. Conditioned on one or more utterances, their model searches a database of movie scripts and retrieves an appropriate response. This was later followed up by Ameixa et al. (2014), who demonstrate that movie subtitles could be used to provide responses to out-of-domain questions using an information retrieval system.

3.1.4 Model

We consider a dialogue as a sequence of M utterances $D = (U_1, \ldots, U_M)$ involving two interlocutors. Each U_m contains a sequence of N_m tokens, i.e. $U_m = (w_{m,1}, \ldots, w_{m,N_m})$, where $w_{m,n}$

[9]The model will therefore require more computational resources as the dialogue advances, which in general is an unwanted property.

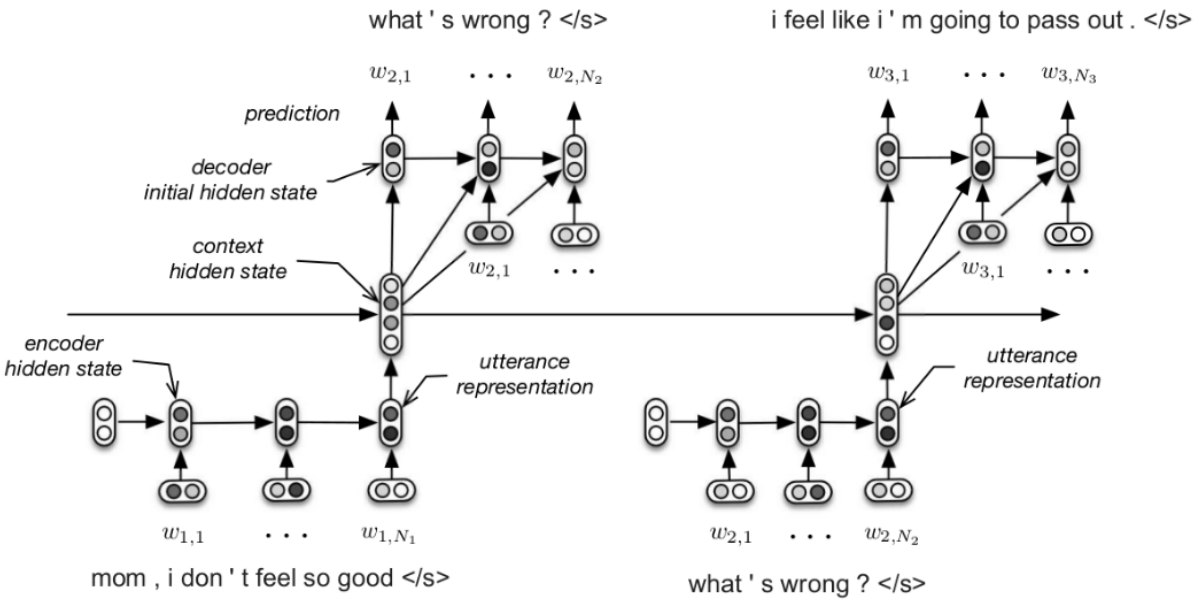

Figure 8: The computational graph of the HRED architecture for a dialogue composed of three turns. Each utterance is encoded into a dense vector and then mapped into the dialogue context, which is used to decode (generate) the tokens in the next utterance. The encoder RNN encodes the tokens appearing within the utterance. The context RNN encodes the discourse-level context of the utterances appearing so far in the dialogue, allowing information and gradients to flow over longer time spans. The decoder predicts one token at a time using a RNN. This figure was adapted from Sordoni et al. (2015a).

is a random variable taking values in the vocabulary V and representing the token at position n in utterance m. The tokens represent both words and *dialogue acts*, e.g. pause and end of turn tokens. The Hierarchical Recurrent Encoder-Decoder (HRED) model, with parameters θ, decomposes the probability of a dialogue D similarly to a recurrent neural network language model (RNNLM) described earlier:

$$\begin{aligned} P_\theta(U_1, \ldots, U_M) &= \prod_{m=1}^{M} P_\theta(U_m | U_{<m}), \\ &= \prod_{m=1}^{M} \prod_{n=1}^{N_m} P_\theta(w_{m,n} | w_{m,<n}, U_{<m}), \end{aligned} \tag{63}$$

where $U_{<m} = (U_1, \ldots, U_{m-1})$ and $w_{m,<n} = (w_{m,1}, \ldots, w_{m,n-1})$, i.e. the tokens preceding n in the utterance U_m. One critical difference compared to the standard RNNLM is that *dialogue acts* are included as separate tokens. Sampling from the model can be performed by sampling one word at a time from the conditional distribution $P_\theta(w_{m,n} | w_{m,<n}, U_{<m})$ conditioned on the previously sampled words.

In the original framework for web query suggestion, HRED predicts the next web query given

the queries already submitted by the user. The history of past submitted queries is considered as a sequence at two levels: a sequence of words for each web query and a sequence of queries. HRED models this hierarchy of sequences with two RNNs: one at the word level and one at the query level. We make a similar assumption for dialogue, namely, that a dialogue can be seen as a sequence of utterances which, in turn, are sequences of tokens.

HRED consists of three components: the *encoder* RNN, the *context* RNN and the *decoder* RNN. These are illustrated in Figure 8. In the following we will assume that the utterance tokens $w_{1,1}, w_{1,2}, \ldots, w_{2,1}, w_{2,2}, \ldots w_{M,N_M}$ have been unfolded into one long sequence of tokens $w_1, \ldots, w_T$. As before, let $d_e \in \mathbb{N}$ be the word embedding dimensionality. The *encoder* RNN maps each utterance to an utterance vector. The utterance vector is the hidden state obtained after the last token of the utterance has been processed by a GRU RNN. Let $h_{t,enc}$ be the hidden state of the *encoder* RNN at time step t, and $d_{h,enc} \in \mathbb{N}$ be its dimensionality. The utterance vector is then defined as:

$$r_{t,enc} = \sigma(H_{ir,enc} I_{w_t} + H_{r,enc} h_{t-1,enc}), \tag{64}$$

$$u_{t,enc} = \sigma(H_{iu,enc} I_{w_t} + H_{u,enc} h_{t-1,enc}), \tag{65}$$

$$\bar{h}_{t,enc} = \tanh(H_{i,enc} I_{w_t} + H_{enc}(r_{t,enc} \cdot h_{t-1,enc})), \tag{66}$$

$$h_{t,enc} = (1 - u_{t,enc}) \cdot h_{t-1,enc} + u_{t,enc} \cdot \bar{h}_{t,enc}, \tag{67}$$

where $I \in \mathbb{R}^{d_e \times |V|}$, $H_{r,enc}, H_{u,enc}, H_{enc} \in \mathbb{R}^{d_{h,enc} \times d_{h,enc}}$ and $H_{ir,enc}, H_{iu,enc}, H_{i,enc} \in \mathbb{R}^{d_{h,enc} \times d_e}$ are the parameters. The higher-level *context* RNN is a GRU RNN, which keeps track of past utterances by iteratively processing the utterance vectors:

$$r_{t,con} = \sigma(H_{ir,con} h_{t,enc} + H_{r,con} h_{t-1,con}), \tag{68}$$

$$u_{t,con} = 1_{(w_t \text{ is not end-of-utterance token})} \sigma(H_{iu,con} h_{t,enc} + H_{u,con} h_{t-1,con}), \tag{69}$$

$$\bar{h}_{t,con} = \tanh(H_{i,con} h_{t,enc} + H_{con}(r_{t,con} \cdot h_{t-1,con})), \tag{70}$$

$$h_{t,con} = (1 - u_{t,con}) \cdot h_{t-1,con} + u_{t,con} \cdot \bar{h}_{t,con}, \tag{71}$$

where $H_{ir,con}, H_{iu,con}, H_{i,con} \in \mathbb{R}^{d_{h,con} \times d_{h,enc}}$ and $H_{con}, H_{r,con}, H_{u,con} \in \mathbb{R}^{d_{h,con} \times d_{h,con}}$ are its parameters, and $d_{h,con} \in \mathbb{N}$ is its hidden state dimensionality. These equations are similar to the GRU equations presented earlier, but with the two differences. First, the input tokens have been replaced with the hidden state of the *encoder*. Second, $u_{t,con}$ is zero whenever w_t is not an end-of-utterance token, i.e. whenever it w_t is not the last token in an utterance. This implies that the state is only updated at the end of a turn in the dialogue. The hidden state $h_{t,con}$ is then given as input to the

decoder RNN, which is also parametrized as the GRU:

$$r_{t,dec} = \sigma(H_{ir,dec} I_{w_t} + H_{ir,dec} h_{t,con} + H_{r,dec} h_{t-1,dec}), \tag{72}$$

$$u_{t,dec} = \sigma(H_{iu,dec} I_{w_t} + H_{iu,dec} h_{t,con} + H_{u,dec} h_{t-1,dec}), \tag{73}$$

$$\bar{h}_{t,dec} = \tanh(H_{i,dec} I_{w_t} + H_{i,dec} h_{t,con} + H_{dec}(r_{t,dec} \cdot h_{t-1,dec})), \tag{74}$$

$$h_{t,dec} = (1 - u_{t,dec}) \cdot h_{t-1,dec} + u_{t,dec} \cdot \bar{h}_{t,dec}, \tag{75}$$

where $H_{ir,dec}, H_{iu,dec}, H_{i,dec} \in \mathbb{R}^{d_{h,dec} \times d_{h,con}}$, $H_{dec}, H_{r,dec}, H_{u,dec} \in \mathbb{R}^{d_{h,dec} \times d_{h,dec}}$ and $H_{ir,dec}, H_{iu,dec}, H_{i,dec} \in \mathbb{R}^{d_{h,dec} \times d_e}$ are its parameters, and $d_{h,dec} \in \mathbb{N}$ is its hidden state dimensionality. The probably distribution over the values of token w_{t+1} is given by eq. (3), where the hidden state at time t is taken to be $h_{t,dec}$. Several extensions are possible to this architecture. For example, the decoder may be parametrized as the LSTM RNN instead of the GRU RNN, and it is also possible to pass the hidden state $h_{t,con}$ through a hyperbolic tangent one-layer neural network before giving it as input to the *decoder* RNN.

For modelling dialogues, the HRED model is expected to be superior to the standard RNN model for two reasons. First, because the *context* RNN allows the model to better capture long-term discourse-level context. The *context* RNN also allows the model to represent a form of common ground between speakers, e.g. by representing topics and concepts shared between the speakers using a distributed vector representation, which we hypothesize to be important for building an effective dialogue system (Clark and Brennan, 1991). Second, because the number of computational steps between utterances is reduced. This makes the objective function more stable w.r.t. the model parameters, and helps propagate the training signal for first-order optimization methods (Sordoni et al., 2015a).

3.2 Multiresolution Recurrent Neural Network

3.2.1 Author's Contribution

The work in this section covers the author's work published in the conference publication:

> "Multiresolution Recurrent Neural Networks: An Application to Dialogue Response Generation" by Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio and Aaron Courville, p. 3288–3294, Association for the Advancement of Artificial Intelligence, 2017.

The conference publication can be accessed at: http://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/download/14571/14217.

The author of this thesis did the majority of the work related to the development of the models, the experiments and the writing up of the conference publication. The author received help for conducting the human evaluation (described later) from Tim Klinger, Gerald Tesauro and Kartik Talamadupula. The author received help on writing up the paper from Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio and Aaron Courville.

3.2.2 Motivation

This section aims to improve the HRED model by learning an improved representation of higher level semantic structure. For example, such higher-level semantic structure might capture information about the conversation topic and about the entities and events mentioned throughout the conversation. Specifically, we propose to do this by generalizing the sequential framework for generative modelling to model multiple parallel sequences. The majority of the previous work on sequential modelling with recurrent neural networks (RNNs), including work on machine translation, speech recognition and question answering (Kumar et al., 2016; Bahdanau et al., 2015; Chorowski et al., 2015; Weston et al., 2015; Luong et al., 2015), has focused on developing new neural network architectures within a deterministic framework. In other words, it has focused on changing the parametrization of the deterministic function mapping input sequences to output sequences, a parametrization still trained by maximizing the log-likelihood of the observed output sequence. Instead, this section pursues a complimentary research direction aimed at generalizing the sequential framework to multiple input and output sequences, where each sequence exhibits its own stochastic process and represents the underlying semantic structure at a distinct granular level. The hope is that this will enable the model to learn high-level abstractions, which will improve the overall generation process.

The model in this section was inspired from the work on jointly modelling dialogue acts and natural language words in a stochastic process by Stolcke et al. (2000). In their work, the authors

define a set of dialogue acts, which are used to classify the high-level intention of an interlocutor's utterance (e.g. *STATEMENT*, *QUESTION*, *BACKCHANNEL*, *AGREEMENT*, *DISAGREEMENT* AND *APPOLOGY*). These authors then propose a hidden Markov model (HMM), where the hidden states are defined as the dialogue acts and where the observations are the corresponding natural language words.[10] Consequently, their HMM is a probabilistic generative model over both high-level abstractions (i.e. dialogue acts) and natural language words.

The model proposed in this section is also motivated in part by the theory of compositional semantics, where the meaning of a phrase depends on the meaning of its constituent parts and their combination (Baroni et al., 2014; Werning et al., 2012; Partee, 2008). This implies that, in many cases, the meaning of a phrase may be derived from the bottom-up by composing together words into higher level semantic units, where the higher-level semantic units represent more complex meaning. In this work we explore a hypothesis based on a different, but related, idea of compositionality, where we posit that a generative model can benefit by explicitly composing the meaning of a phrase bottom-up, starting from an abstract, underspecified representation of the phrase and then incrementally making it more concrete.

We propose a new class of RNN models, called Multiresolution Recurrent Neural Networks (MrRNNs), which model multiple parallel sequences by factorizing the joint probability over the sequences. In particular, we impose a hierarchical structure on the sequences, such that information from high-level (abstract) sequences flows to low-level sequences (e.g. natural language sequences of words). In other words, the high-level sequences represent high-level semantic structure

This architecture exhibits a new objective function for training: the joint log-likelihood over all observed parallel sequences. In contrast to the log-likelihood objective function over a single, long sequence, this objective function biases the model towards modelling high-level abstractions. At test time, the model generates first the high-level sequence and afterwards the natural language sequence of words.

3.2.3 Prior Related Work

We now discuss some of the related work conducted prior to the research in this section. Closely related to our proposal is the model proposed by Ji et al. (2016), which jointly models natural language text and high-level discourse structure. However, it only models a discrete class per sentence at the higher level, which must be manually annotated by humans. On the other hand, the model we propose models a sequence of automatically extracted high-level tokens. Recurrent neural network models with stochastic latent variables, such as the Variational Recurrent Neural Networks by Chung et al. (2015), are also related. These models also attempt to learn the high-level

[10] In Stolcke et al. (2000), the authors also include acoustic features, including prosodic features.

representations, while simultaneously learning to model the generative process over high-level sequences and low-level sequences, which arguably is a more difficult optimization problem. In addition to this, such models assume the high-level latent variables follow continuous distributions.

Recent dialogue-specific neural network architectures, including the model proposed by Wen et al. (2017), are also related. Different from the model we propose, they require domain-specific hand-crafted high-level representations (for example, a hand-crafted dialogue state) learned from human-labelled examples. They also usually consist of several sub-components each trained with its own objective function.

Finally, the idea of modelling a higher-level semantic structure in a dialogue is closely related to the dialogue state tracking challenge (Williams et al., 2013, 2016). This is a goal-driven dialogue system challenge, where the task is to map the dialogue history to a discrete state representing the salient information necessary for the goal-driven dialogue system to attain its goal.

3.2.4 Model

We consider the problem of modelling two parallel sequences. As for the Hierarchical Recurrent Encoder-Decoder (HRED) model, each sequence consists of a sequence of utterances, which consists of a sequence of tokens. Formally, let $(\mathbf{w}_1, \ldots, \mathbf{w}_N)$ be the first sequence of length N where $\mathbf{w}_n = (w_{n,1}, \ldots, w_{n,K_n})$ is the n'th constituent sequence (utterance) consisting of K_n discrete tokens (words) from vocabulary V^w. Similarly, let $(\mathbf{z}_1, \ldots, \mathbf{z}_N)$ be the second sequence, also of length N, where $\mathbf{z}_n = (z_{n,1}, \ldots, z_{n,L_n})$ is the n'th constituent (utterance) sequence consisting of L_n discrete tokens (words) from vocabulary V^z. In our experiments, each sequence $\mathbf{w}_n$ will consist of the words in a dialogue utterance, and each sequence $\mathbf{z}_n$ will contain the coarse tokens w.r.t. the same utterance (e.g. the nouns or activities and entities found in the utterance). In other words, the sequence $(\mathbf{z}_1, \ldots, \mathbf{z}_N)$ will help represent higher level semantic structure, either by representing the abstract semantic content directly (e.g. activities and entities found in the utterance) or by representing an underspecified variant of the actual natural language phrase focused on the most salient aspects (e.g. nouns).

We will define the generation process as follows. First, the sequence $\mathbf{z}_1$ is generated. Then sequence $\mathbf{z}_2$ is generated conditioned on $\mathbf{z}_1$ and so on. This generation process is analogous to the probabilistic graphical model of the standard RNN. Once $\mathbf{z}_1, \ldots, \mathbf{z}_N$ have been generated, the sequence $\mathbf{w}_1$ is generated conditioned on $\mathbf{z}_1$. Then $\mathbf{w}_2$ is generated conditioned only on $\mathbf{w}_1, \mathbf{z}_1$ and $\mathbf{z}_2$, and so on. In other words, each low-level sequence is conditioned on the previous low-level sequences and the current and previous high-level sequences. Formally, we therefore assume that $\mathbf{w}_n$ is independent of $\mathbf{z}_{n'}$ conditioned on $\mathbf{z}_1, \ldots, \mathbf{z}_n$ for $n' > n$. Let θ be the parameters of the

generative model and factor the probability over sequences:

$$P_\theta(\mathbf{w}_1, \ldots, \mathbf{w}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) = \prod_{n=1}^{N} P_\theta(\mathbf{z}_n|\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}) \prod_{n=1}^{N} P_\theta(\mathbf{w}_n|\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, \mathbf{z}_1, \ldots, \mathbf{z}_n)$$
$$= \prod_{n=1}^{N} P_\theta(\mathbf{z}_n|\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}) P_\theta(\mathbf{w}_n|\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, \mathbf{z}_1, \ldots, \mathbf{z}_n), \tag{76}$$

where we define the conditional probabilities over the tokens in each constituent sequence as:

$$P_\theta(\mathbf{z}_n|\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}) = \prod_{m=1}^{L_n} P_\theta(z_{n,m}|z_{n,1}, \ldots, z_{n,m-1}, \mathbf{z}_1, \ldots, \mathbf{z}_{n-1})$$
$$P_\theta(\mathbf{w}_n|\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, \mathbf{z}_1, \ldots, \mathbf{z}_n) = \prod_{m=1}^{K_n} P_\theta(w_{n,m}|w_{n,1}, \ldots, w_{n,m-1}, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, \mathbf{z}_1, \ldots, \mathbf{z}_n)$$

We call the distribution over $(\mathbf{z}_1, \ldots, \mathbf{z}_N)$ the coarse sub-model, and the distribution over $(\mathbf{w}_1, \ldots, \mathbf{w}_N)$ the natural language sub-model. For the coarse sub-model, we parametrize the conditional distribution $P_\theta(\mathbf{z}_n|\mathbf{z}_1, \ldots, \mathbf{z}_{n-1})$ as the HRED model described in section 3.1 on the sequence $(\mathbf{z}_1, \ldots, \mathbf{z}_N)$. For the natural language sub-model, we parametrize $P_\theta(\mathbf{w}_n|\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, \mathbf{z}_1, \ldots, \mathbf{z}_n)$ as the HRED model on the sequence $(\mathbf{w}_1, \ldots, \mathbf{w}_N)$. However, here there is one difference. The *coarse prediction encoder* GRU-gated RNN encodes all the previously generated tokens $\mathbf{z}_1, \ldots, \mathbf{z}_n$ into a real-valued vector, as defined by eq. (5), which is concatenated with the *context* RNN and given as input to the natural language *decoder* RNN. The *coarse prediction encoder* RNN is important because it encodes the high-level information, which is transmitted to the natural language sub-model. At generation time, the coarse sub-model generates a coarse sequence (e.g. a sequence of nouns), which corresponds to a high-level decision about what the natural language sequence should contain (e.g. nouns to include in the natural language sequence). Conditioned on the coarse sequence, the natural language sub-model then generates a natural language sequence (e.g. a dialogue utterance). The model is illustrated in Figure 9.

Since $(\mathbf{z}_1, \ldots, \mathbf{z}_N)$ and $(\mathbf{w}_1, \ldots, \mathbf{w}_N)$ are both observed in the training set, we optimize the parameters w.r.t. the joint log-likelihood over both sequences. At test time, to generate a response for sequence n we approximate the most likely response:

$$\arg\max_{\mathbf{w}_n, \mathbf{z}_n} P_\theta(\mathbf{w}_n, \mathbf{z}_n|\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, \mathbf{z}_1, \ldots, \mathbf{z}_{n-1})$$
$$\approx \arg\max_{\mathbf{w}_n} P_\theta(\mathbf{w}_n|\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, \mathbf{z}_1, \ldots, \mathbf{z}_{n-1}, \mathbf{z}_n) \arg\max_{\mathbf{z}_n} P_\theta(\mathbf{z}_n|\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}), \tag{77}$$

where we further approximate the MAP for each constituent sequence using beam search (Graves, 2012).

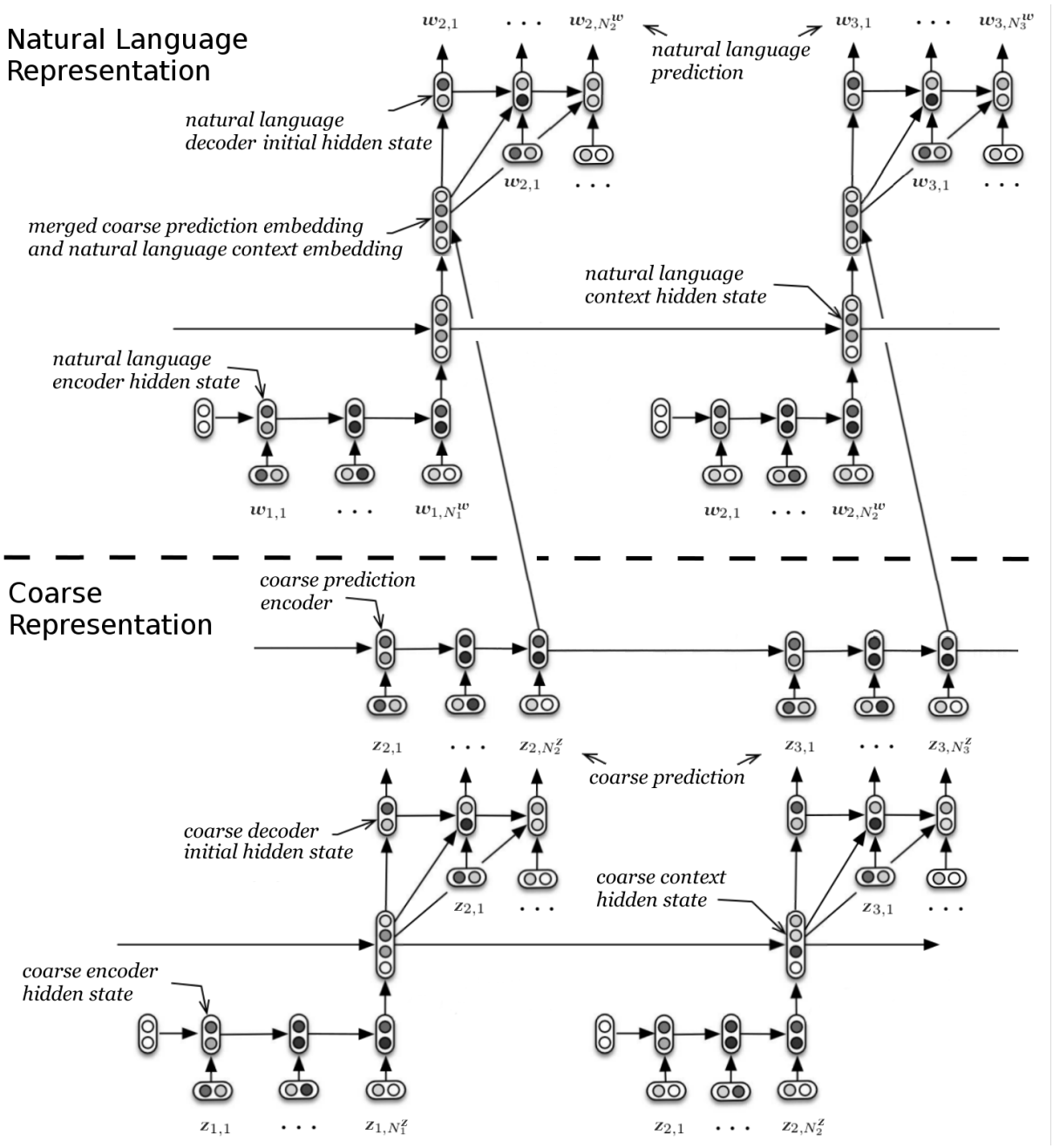

Figure 9: Computational graph for the Multiresolution Recurrent Neural Network (MrRNN). The lower part models the stochastic process over coarse tokens, and the upper part models the stochastic process over natural language tokens. The rounded boxes represent (deterministic) real-valued vectors, and the variables z and w represent the coarse tokens and natural language tokens respectively.

3.3 Latent Variable Recurrent Encoder-Decoder

3.3.1 Author's Contribution

The work in this section covers the author's work published in the conference publication:

> "A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues"
> by Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio, p. 3295–3301, Association for the Advancement of Artificial Intelligence, 2017.

The conference publication can be accessed at: https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14567/14219. [11]

The author of this thesis did the majority of the work related to the development of the models, the experiments and the writing up of the conference publication. The author received technical assistance from Alessandro Sordoni and Ryan Lowe on some parts of the implementation. The author received help on writing up the paper from Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio.

3.3.2 Motivation

This section is motivated by the observation that in many RNN architectures the variability or stochasticity in the model occurs only when an output is sampled. This is often an inappropriate place to inject variability (Boulanger-Lewandowski et al., 2012; Chung et al., 2015; Bayer and Osendorfer, 2014). This is especially true for sequential data, such as speech and natural language, which possess a hierarchical generation process with complex intra-sequence dependencies. For instance, natural language dialogue involves at least two levels of structure; within a single utterance the structure is dominated by local statistics of the language, while across utterances there is a distinct source of uncertainty (or variance) characterized by aspects such as conversation topic, speaker goals and speaker style. The Multiresolution Recurrent Neural Network (MrRNN), proposed earlier, overcomes this problem by modelling several observed sequences in parallel while conditioning the low-level (natural language) sequences on the high-level (abstract) sequences. However, high-level sequences may not always be available and even when they are available they may be ineffective. For example, they may not capture important aspects of the data distribution.

This section is further motivated by the observation that natural language, in particular real-world dialogue, exhibits a high amount of uncertainty and ambiguity. We posit that this high

[11] The source code for the model described next is available at:
https://github.com/julianser/hed-dlg-truncated.

amount of uncertainty and ambiguity cannot be modelled effectively by a model where all variability is modelled at the word or token level.

This section introduces a novel hierarchical stochastic latent variable neural network architecture to explicitly model generative processes that possess multiple levels of variability. The model we propose for generative dialogue modelling has two levels of stochastic variables: a high-level latent stochastic variable and a low-level variable (e.g. a dialogue utterance). The latent stochastic variable has a normal distribution. The model samples the latent stochastic variable, and then conditioned on it the model samples the low-level variable. In other words, the stochastic process has two levels: one latent and one observed.

3.3.3 Prior Related Work

We next discuss some of the related work conducted prior to the research in this section. The use of a stochastic latent variable learned by maximizing a variational lower-bound is inspired by the variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014). Such models have been used predominantly for generating images in the continuous domain (Gregor et al., 2015; Bachman and Precup, 2015). However, there has also been recent work applying these architectures for generating sequences, such as the Variational Recurrent Neural Networks (VRNN) (Chung et al., 2015), which was applied for speech and handwriting synthesis, and Stochastic Recurrent Networks (STORN) (Bayer and Osendorfer, 2014), which was applied for music generation and motion capture modelling. Both the VRNN and STORN incorporate stochastic latent variables into RNN architectures, but unlike the model we will propose they sample a separate latent variable at each time step of the decoder. Their models do not exploit the hierarchical structure in the data, and thus does not model higher-level variability, which is clearly important in dialogue.

Most similar to our proposed model is the Variational Recurrent Autoencoder (Fabius and van Amersfoort, 2014) and the Variational Autoencoder Language Model (Bowman et al., 2016), which apply encoder-decoder architectures to model music and text. The model we will propose is different in several ways. In the model we will propose, the latent variable is conditioned on all previous sub-sequences (sentences). This enables the model to generate multiple sub-sequences (sentences), but it also makes the latent variables co-dependent through the observed tokens. The model builds on the hierarchical architecture of the Hierarchical Recurrent Encoder-Decoder (HRED) model, which makes it applicable to generation conditioned on long contexts. Unlike the previous models in the literature, it also has a direct deterministic connection between the *context* and *decoder* RNN, which allows the model to transfer deterministic pieces of information between its components.

3.3.4 Model

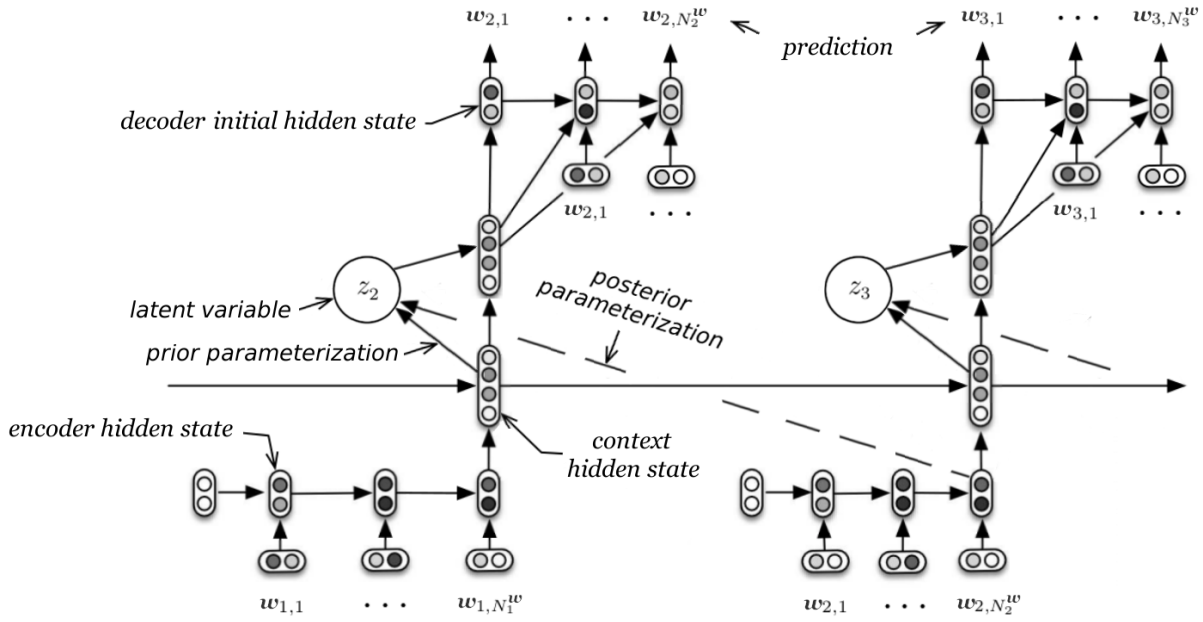

Figure 10: Computational graph for VHRED model. Rounded boxes represent (deterministic) real-valued vectors. Variables $\mathbf{z}$ represent latent stochastic variables.

The model we propose is called the Latent Variable Hierarchical Recurrent Encoder-Decoder (VHRED) model. This model augments the HRED model with a latent variable at the decoder, which is trained by maximizing a variational lower-bound on the log-likelihood. This allows it to model hierarchically-structured sequences in a two-step generation process—first sampling the latent variable, and then generating the output sequence—while maintaining long-term context.

As before when describing the MrRNN model, let $(\mathbf{w}_1, \ldots, \mathbf{w}_N)$ be a sequence consisting of N constituent sequences, where $\mathbf{w}_n = (w_{n,1}, \ldots, w_{n,M_n})$ is the n'th constituent sequence and $w_{n,m} \in V$ is the m'th discrete token in that sequence. The VHRED model uses a stochastic latent variable $\mathbf{z}_n \in \mathbb{R}^{d_z}$ for each constituent sequence $n = 1, \ldots, N$ conditioned on all previous observed tokens. Given $\mathbf{z}_n$, the model next generates the n'th constituent sequence tokens $\mathbf{w}_n = (w_{n,1}, \ldots, w_{n,M_n})$:

$$P_\theta(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \mathcal{N}_{\mathbf{z}_n}(\boldsymbol{\mu}_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}), \Sigma_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1})), \tag{78}$$

$$P_\theta(\mathbf{w}_n \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \prod_{m=1}^{M_n} P_\theta(w_{n,m} \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, w_{n,1}, \ldots, w_{n,m-1}), \tag{79}$$

where $\mathcal{N}_{\mathbf{z}_n}(\boldsymbol{\mu}, \Sigma)$ is the multivariate normal distribution density of variable $\mathbf{z}_n$ with mean $\boldsymbol{\mu} \in \mathbb{R}^{d_z}$ and covariance matrix $\Sigma \in \mathbb{R}^{d_z \times d_z}$, which is constrained to be a diagonal matrix.

As shown in Figure 10. the VHRED model contains the same three components as the HRED model. The *encoder* RNN deterministically encodes a single constituent sequence into a fixed-size

real-valued vector, as described in section 3.1. The *context* RNN deterministically takes as input the output of the *encoder* RNN, and encodes all previous constituent sequences into a fixed-size real-valued vector, as described in section 3.1. This vector is transformed through a two-layer feed-forward neural network with hyperbolic tangent gating function. A matrix multiplication is applied to the output of the feed-forward network, which defines the multivariate normal mean $\boldsymbol{\mu}_{\text{prior}}$. Similarly, for the diagonal covariance matrix Σ_{prior} a different matrix multiplication is applied to the net's output followed by the softplus function, which ensures positive values (Chung et al., 2015).

The model's latent variables are inferred by maximizing the variational lower-bound, which factorizes into independent terms for each constituent sequence:

$$\begin{aligned}\log P_\theta(\mathbf{w}_1, \ldots, \mathbf{w}_N) \geq \sum_{n=1}^{N} &-\text{KL}\left[Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_n) || P_\theta(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_{n-1})\right] \\ &+ \mathbb{E}_{Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_n)}\left[\log P_\theta(\mathbf{w}_n \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1})\right], \end{aligned} \tag{80}$$

where $\text{KL}[Q||P]$ is the Kullback-Leibler (KL) divergence between distributions Q and P. The distribution $Q_\psi(\mathbf{z} \mid w_1, \ldots, w_M)$ is the approximate posterior distribution (also known as the *encoder model* or *recognition model*), which aims to approximate the intractable true posterior distribution:

$$\begin{aligned} Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_N) &= Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_n) \\ &= \mathcal{N}(\boldsymbol{\mu}_{\text{posterior}}(\mathbf{w}_1, \ldots, \mathbf{w}_n), \Sigma_{\text{posterior}}(\mathbf{w}_1, \ldots, \mathbf{w}_n)) \\ &\approx P_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_N), \end{aligned} \tag{81}$$

where $\boldsymbol{\mu}_{\text{posterior}}$ defines the approximate posterior mean and $\Sigma_{\text{posterior}}$ defines the approximate posterior covariance matrix (assumed diagonal) as a function of the previous constituent sequences $\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}$ and the current constituent sequence $\mathbf{w}_n$. The posterior mean $\boldsymbol{\mu}_{\text{posterior}}$ and covariance $\Sigma_{\text{posterior}}$ are determined in the same way as the prior, via a matrix multiplication with the output of the feed-forward network, with a softplus function applied for the covariance.

At test time, conditioned on the previous observed constituent sequences $(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1})$, a sample $\mathbf{z}_n$ is drawn from the prior $\mathcal{N}(\boldsymbol{\mu}_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}), \Sigma_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}))$ for each constituent sequence. This sample is concatenated with the output of the *context* RNN and given as input to the *decoder* RNN as in the HRED model, which then generates the constituent sequence token-by-token. At training time, for $n = 1, \ldots, N$, a sample $\mathbf{z}_n$ is drawn from the approximate posterior $\mathcal{N}(\boldsymbol{\mu}_{\text{posterior}}(\mathbf{w}_1, \ldots, \mathbf{w}_n), \Sigma_{\text{posterior}}(\mathbf{w}_1, \ldots, \mathbf{w}_n))$ and used to estimate the gradient of the variational lower-bound given by eq. (80). The approximate posterior is parametrized by its own one-layer feed-forward neural network, which takes as input the output of the *context* RNN at the current time step, as well as the output of the *encoder* RNN for the next constituent sequence.

As before, assume that the utterance tokens $w_{1,1}, w_{1,2}, \ldots, w_{2,1}, w_{2,2}, \ldots w_{M,N_M}$ have been unfolded into one long sequence of tokens $(w_1, \ldots, w_T)$. Formally, let $h_{t,con} \in \mathbb{R}^{d_{h,con}}$ be the hidden state of the *context* encoder at time t, and define:

$$\bar{h}_{t,con} = \tanh(H_{l_2,prior} \tanh(H_{l_1,prior} h_{t,con})), \tag{82}$$

$$\boldsymbol{\mu}_{t,\text{prior}} = H_{\mu,prior} \bar{h}_{t,con}, \tag{83}$$

$$\Sigma_{t,\text{prior}} = \text{diag}(\log(1 + \exp(H_{\Sigma,prior} \bar{h}_{t,con}))), \tag{84}$$

where $H_{l_1,prior} \in \mathbb{R}^{d_z \times d_{h,con}}$ and $H_{\Sigma,prior}, H_{\mu,prior}, H_{l_2,prior} \in \mathbb{R}^{d_z \times d_z}$ are its parameters, and where $\text{diag}(\mathbf{x})$ is a function mapping a vector $\mathbf{x}$ to a matrix with diagonal elements $\mathbf{x}$ and all off-diagonal elements equal to zero. At generation time, these quantities are computed at the time step corresponding to the end of each utterance, i.e. when w_t is the end-of-utterance token, and afterwards the latent variable is sampled $\mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{t,\text{prior}}, \Sigma_{t,\text{prior}})$. The equations for the approximate posterior are similar. Let $h_{t,pos} \in \mathbb{R}^{d_{h,con}+d_{h,enc}}$ be the concatenation of $h_{t,con}$ and the hidden state of the *encoder* RNN at the end of the next constituent sequence, which we assume has dimensionality $d_{h,enc}$. The approximate posterior is given as:

$$\bar{h}_{t,pos} = \tanh(H_{l_2,posterior} \tanh(H_{l_1,posterior} h_{t,pos})), \tag{85}$$

$$\boldsymbol{\mu}_{t,\text{posterior}} = H_{\mu,posterior} \bar{h}_{t,pos}, \tag{86}$$

$$\Sigma_{t,\text{posterior}} = \text{diag}(\log(1 + \exp(H_{\Sigma,posterior} \bar{h}_{t,pos}))), \tag{87}$$

where $H_{l_1,posterior} \in \mathbb{R}^{d_z \times (d_{h,con}+d_{h,enc})}$ and $H_{\Sigma,posterior}, H_{\mu,posterior}, H_{l_2,posterior} \in \mathbb{R}^{d_z \times d_z}$ are its parameters. At training time, the latent variable is sampled at the end of each utterance:
$\mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{t,\text{posterior}}, \Sigma_{t,\text{posterior}})$.

The VHRED model helps to reduce the problems with the generation process used by the RNNLM and HRED model outlined above. The variation of the output sequence is now modelled in two ways: at the sequence-level with the conditional prior distribution over $\mathbf{z}$, and at the constituent sequence-level (token-level) with the conditional distribution over tokens $w_1, \ldots, w_M$. The variable $\mathbf{z}$ helps model long-term output trajectories, by representing high-level information about the sequence, which in turn allows the variable h_m to primarily focus on summarizing the information up to token M. One interpretation of this model is that the randomness injected by the variable $\mathbf{z}$ corresponds to higher-level decisions, like the topic or the sentiment of the sentence.

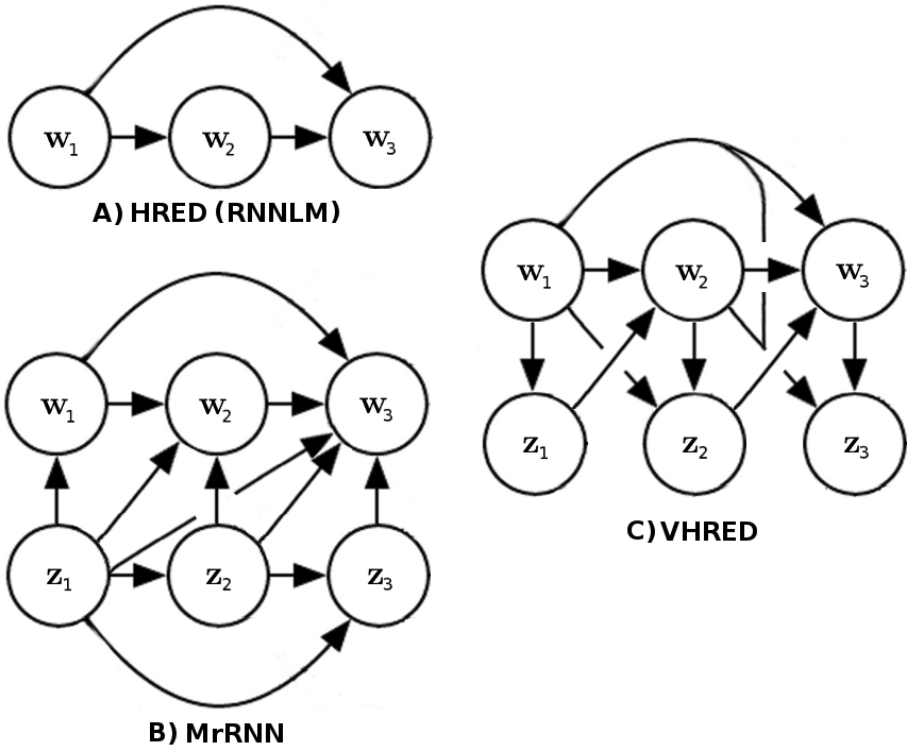

Figure 11: Probabilistic graphical models for dialogue response generation. Variables $\mathbf{w}$ represent natural language utterances. Variables $\mathbf{z}$ represent discrete or continuous stochastic latent variables. (A): HRED (and RNNLM) uses a shallow generation process. This is problematic because it has no mechanism for incorporating uncertainty and ambiguity at a higher level, and because it forces the model to generate compositional and long-term structure incrementally on a word-by-word basis. (B): MrRNN expands the generation process by adding a sequence of observed, discrete stochastic variables for each utterance, which helps generate responses with higher level semantic structure. (C): VHRED expands the generation process by adding one learned latent variable for each utterance, which helps incorporate uncertainty and ambiguity in the representations and generate meaningful, diverse responses.

3.3.5 Comparing VHRED to HRED and MrRNN

To better understand the motivation and structure of the VHRED model, it is useful to compare it to the MrRNN and HRED models. Figure 11 illustrates the probabilistic directed graphical models of the VHRED, MrRNN and HRED models. As discussed earlier, HRED uses a hierarchical structure in order to incorporate long-term discourse-level context. However, similar to the RNNLM, it implements a shallow stochastic generation process where the output is generated directly one word at a time. This is problematic because there is no mechanism for incorporating uncertainty and ambiguity and because it forces the model to generate compositional and long-term structure incrementally on a word-by-word basis. MrRNN overcomes these problems by expanding the stochastic generation process to also output a sequence of high-level discrete stochastic variables for each utterance, which represent high-level semantic structure. In contrast to MrRNN, VHRED overcomes the problems of HRED by expanding the stochastic generation process to include one continuous, latent variable for each utterance, which helps to incorporate uncertainty and ambiguity in the representations and helps to generate meaningful, diverse responses.

3.4 Experiments

3.4.1 Tasks

In order to investigate the performance of the previously presented models, several empirical experiments have been carried out. The experiments focus on the task of conditional response generation: given a dialogue context, consisting of one or more utterances, the model must generate the next response in the dialogue. To get a broad view of the performance, we present results on two tasks: a goal-driven dialogue task and a non-goal driven dialogue task.

Ubuntu Dialogue Corpus The goal-driven dialogue task we consider is technical support for the Ubuntu operating system, where we use the Ubuntu Dialogue Corpus (Lowe et al., 2015b).[12][13] The corpus consists of about 0.5 million natural language dialogues extracted from the *#Ubuntu* Internet Relayed Chat (IRC) channel. Users entering the chat channel usually have a specific technical problem. The users first describe their problem and afterwards other users try to help them resolve it. The technical problems range from software-related issues (e.g. installing or upgrading existing software) and hardware-related issues (e.g. fixing broken drivers or partitioning hard drives) to informational needs (e.g. finding software with specific functionality). For more details, the reader is referred to Lowe et al. (2017b).

Twitter Dialogue Corpus The next task we consider is the non-goal-driven task of generating responses to Twitter conversations. We use a Twitter dialogue corpus extracted in the first half of 2011 using a procedure similar to (Ritter et al., 2011a). Unlike the Ubuntu domain, Twitter conversations are often more noisy and do not necessarily center around a single topic. However, due to their open-ended nature, it is interesting to investigate how different dialogue models perform in this domain. We perform a minimal preprocessing on the dataset to remove irregular punctuation marks and afterwards tokenize it using the Moses tokenizer.[14] The dataset is split into training, validation and test sets containing respectively 749,060, 93,633 and 10,000 dialogues.

[12] We use the Ubuntu Dialogue Corpus v2.0 extracted January, 2016: http://cs.mcgill.ca/~jpineau/datasets/ubuntu-corpus-1.0/

[13] For the context and response pairs in the official Ubuntu Dialogue Corpus, there is no distinction between the users having technical problems and other users helping them resolve their problems. Therefore the model must learn to act as both a user in need of technical support and as a user providing technical support.

[14] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl, Retrieved June, 2015

3.4.2 Multiresolution RNN Representations

For the MrRNNs, we experiment with two procedures for extracting the coarse sequence representations:

Noun Representation This procedure aims to exploit the basic high-level structure of natural language discourse. It is motivated by the hypothesis that dialogues are topic-driven and that these topics may be characterized by the observed nouns. In addition to a tokenizer, used by both the HRED and VHRED model, it requires a part-of-speech (POS) tagger to identify the nouns in the dialogue. The procedure uses a set of 84 and 795 predefined stop words for Ubuntu and Twitter respectively. It maps a natural language utterance to its coarse representation by extracting all the nouns using the POS tagger and then removing all stop words and repeated words (keeping only the first occurrence of a word). Dialogue utterances without nouns are assigned the "no_nouns" token. The procedure also extracts the tense of each utterance and adds it to the beginning of the coarse representation.

Activity-Entity Representation This procedure is specific to the Ubuntu technical support task, for which it aims to exploit domain knowledge related to technical problem solving. It is motivated by the hypothesis that the majority of dialogues are centered around *activities* and *entities*. For example, in the Ubuntu technical support task, it is very common for users to state a specific problem they want to resolve (e.g. *how do I install program X?* or *My driver X doesn't work, how do I fix it?*). In response to such questions, other users often respond with specific instructions (e.g. *Go to website X to download software Y* or *Try to execute command X*). In such cases, it is clear that the principal information resides in the technical entities (*X* and *Y*) and in the verbs (e.g. *install*, *fix*, *download*), and therefore that it will be advantageous to explicitly model this structure. Motivated by this observation, the procedure uses a set of 192 activities (verbs), created by manual inspection, and a set of 3115 technical entities and 230 frequent terminal commands, extracted automatically from available package managers and from the web. The procedure uses the POS tagger to extract the verbs from the each natural language utterance. It maps the natural language to its coarse representation by keeping only verbs from the activity set, as well as entities from the technical entity set (irrespective of their POS tags). If no activity is found in an utterance, the representation is assigned the "none_activity" token. The procedure also appends a binary variable to the end of the coarse representation indicating if a terminal command was detected in the utterance. Finally, the procedure extracts the tense of each utterance and adds it to the beginning of the coarse representation.

Both extraction procedures are applied at the utterance level, therefore there exists a one-to-

one alignment between coarse sequences and natural language sequences (utterances). There also exists a one-to-many alignment between the coarse sequence tokens and the corresponding natural language tokens, with the exception of a few special tokens. In other words, one sequence of coarse tokens may correspond to many different natural language utterances. Further details are given in Appendix I.

3.4.3 Model Training & Testing

We implement all models using the Theano library (Al-Rfou et al., 2016). We optimize all models based on the training set joint log-likelihood over coarse sequences and natural language sequences using the first-order stochastic gradient optimization method Adam (Kingma and Ba, 2015). We train all models using early stopping with patience on the joint-log-likelihood (Bengio, 2012). We choose our hyperparameters based on the joint log-likelihood of the validation set. We define the 20,000 most frequent words as the vocabulary and the word embedding dimensionality to size 300 for all models, with the exception of the RNNLM and HRED on Twitter, where we use embedding dimensionality of size 400. We apply gradient clipping to stop the parameters from exploding (Pascanu et al., 2012). At test time, we use a procedure called beam search to find the response with the highest probability (Graves, 2012). In particular, we use a beam search of size 5.

Baselines We compare our models to the standard RNNLM with LSTM gating function (Mikolov et al., 2010) (RNNLM), which at test time is similar to the Seq2Seq LSTM model (Sutskever et al., 2014). For both Ubuntu and Twitter, we specify the RNNLM model to have 2000 hidden units with the LSTM gating function.

In addition, we also include a non-neural network baseline for Twitter, specifically the TF-IDF retrieval-based model proposed by Lowe et al. (2015b).

HRED We experiment with the HRED model with LSTM gating function for the *decoder* RNN and GRU gating function for the *encoder* RNN and *context* RNN. For Ubuntu, we specify the HRED model to have 500, 1000 and 500 hidden units respectively for the *encoder* RNN, *context* RNN and *decoder* RNN. For Twitter, we specify the HRED model to have 2000, 1000 and 1000 hidden units respectively for the *encoder* RNN, *context* RNN and *decoder* RNN.

Multiresolution RNN We experiment with the MrRNN model, where the coarse sub-model is parametrized as the Bidirectional-HRED model (Serban et al., 2016) with 1000, 1000 and 2000 hidden units respectively for the coarse-level *encoder*, *context* and *decoder* RNNs. The natural language sub-model is parametrized as a conditional HRED model with 500, 1000 and 2000 hid-

den units respectively for the natural language *encoder*, *context* and *decoder* RNNs. The *coarse prediction encoder* RNN GRU RNN is parametrized with 500 hidden units.

HRED + Act.-Ent. Features For Ubuntu, we also experiment with another model, called *HRED + Activity-Entity Features*, which has access to the past activity-entity pairs. This model is similar to to the natural language sub-model of the MrRNN model, with the difference that the natural language *decoder* RNN is conditioned on a real-valued vector, produced by a GRU RNN encoding *only* the past coarse-level activity-entity sub-sequences. This baseline helps differentiate between a model which observes the coarse-level sequences only as as additional features and a model which explicitly models the stochastic process of the coarse-level sequences. We specify the model to have 500, 1000, 2000 hidden units respectively for the *encoder* RNN, *context* RNN and *decoder* RNN. We specify the GRU RNN encoding the past coarse-level activity-entity sub-sequences to have 500 hidden units.

VHRED We experiment with the Latent Variable Hierarchical Recurrent Encoder-Decoder (VHRED). The *encoder* and *context* RNNs for the VHRED model are parametrized in the same way as the corresponding HRED models. The only difference in the parametrization of the *decoder* RNN is that the *context* RNN output vector is now concatenated with the generated stochastic latent variable. Furthermore, we initialize the parameters of the feed-forward networks of the prior and posterior distributions with values drawn from a zero-mean normal distribution with variance 0.01 and with biases equal to zero. We also multiply the diagonal covariance matrices of the prior and posterior distributions with 0.1 to make training more stable, because a high variance makes the gradients w.r.t. the reconstruction cost unreliable, which is fatal at the beginning of the training process. The VHRED *encoder* and *context* RNNs are initialized to the parameters of the corresponding converged HRED models. We also use two heuristics proposed by Bowman et al. (2016): we drop words in the decoder with a fixed drop rate of 25% and multiply the KL terms in eq. (80) by a scalar, which starts at zero and linearly increases to 1 over the first $60,000$ and $75,000$ training batches on Twitter and Ubuntu respectively. Applying these heuristics helped substantially to stabilize the training process and make the model use the stochastic latent variables.

3.4.4 Ubuntu Experiments

Evaluation Methods We carry out an in-lab human study to evaluate the Ubuntu models. We recruit 5 human evaluators, and show them each $30 - 40$ dialogue contexts with the ground truth response and 4 candidate responses (HRED, HRED + Activity-Entity Features and MrRNNs). For each context example, we ask them to compare the candidate responses to the ground truth response and dialogue context, and rate them for fluency and relevancy on Likert-type scale 0-4.

Table 3: Ubuntu evaluation using precision (P), recall (R), F1 and accuracy metrics w.r.t. activity, entity, tense and command (Cmd) on ground truth utterances. The superscript * indicates scores significantly different from baseline models at 95% confidence level.

	Activity			Entity			Tense	Cmd
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**Acc.**	**Acc.**
RNNLM	1.7	1.03	1.18	1.18	0.81	0.87	14.57	94.79
HRED	5.93	4.05	4.34	2.81	2.16	2.22	22.2	92.58
VHRED	6.43	4.31	4.63	3.28	2.41	2.53	20.2	92.02
HRED + Act.-Ent.	7.15	5.5	5.46	3.03	2.43	2.44	28.02	86.69
MrRNN Noun	5.81	3.56	4.04	**8.68**	**5.55**	**6.31****	24.03	90.66
MrRNN Act.-Ent.	**16.84**	**9.72**	**11.43****	4.91	3.36	3.72	**29.01**	**95.04**

Our setup is very similar to the evaluation setup used by (Koehn and Monz, 2006) and comparable to (Liu et al., 2016). Further details are given in appendix III.

In addition to these experiments, we propose a new set of metrics for evaluating model responses on Ubuntu, which compare the activities and entities in the model generated response with those of the ground truth response. That is, the ground truth and model responses are mapped to their respective activity-entity representations, using the automatic procedure discussed in section 3.4.2, and then the overlap between their activities and entities are measured according to precision, recall and F1-score. Based on a careful manual inspection of the extracted activities and entities, we believe that these metrics are particularly suited for the goal-driven Ubuntu Dialogue Corpus. The activities and entities reflect the principal instructions given in the responses, which are key to resolving the technical problems. Therefore, a model able to generate responses with actions and entities similar to the ground truth human responses – which often do lead to solving the users problem – is more likely to yield a successful dialogue system.

Results The results on Ubuntu are given in Table 3 and Table 4. The MrRNNs clearly perform substantially better than all the other models w.r.t. both human evaluation and automatic evaluation metrics. The MrRNN with noun representations achieves two to three times higher scores w.r.t.

Table 4: Ubuntu evaluation using human fluency and relevancy scores given on a Likert-type scale 0-4. The superscript * indicates scores significantly different from baseline models at 90% confidence level. The RNNLM and VHRED models are excluded, since they were not part of the human evaluation.

	Human Scores	
Model	**Fluency**	**Relevancy**
HRED	2.98	1.01
HRED + Act.-Ent.	2.96	0.75
MrRNN Noun	$\mathbf{3.48}^*$	$\mathbf{1.32}^*$
MrRNN Act.-Ent.	$\mathbf{3.42}^*$	1.04

entities compared to other models, and the human evaluators also rate its fluency and relevancy substantially higher than other models. The MrRNN with activity representations achieves two to three times higher scores w.r.t. activities compared to other models and nearly two times higher scores w.r.t. entities compared to all baselines. Human evaluators also rate its fluency substantially higher than the other models. However, its relevancy is rated only slightly higher compared to the HRED model, which we believe is because human evaluators are more likely to notice software entities than actions in the dialogue responses (even though actions are critical to solving the actual technical problem). Overall, the results demonstrate that the MrRNNs have learned to model some amount of high-level goal-driven sequential structure on Ubuntu. In comparison, the VHRED model performs better than both the HRED model and the MrRNN model with noun representations on activities, which suggests that the latent variables help to model frequently occurring activity patterns. However, the VHRED model performs worse than both MrRNN models w.r.t. entities. There are two explanations for these observations. First, the latent variables might have a difficult time modelling rare patterns, since the optimization problem is difficult. Second, the normally-distributed latent variables may not be able to represent discrete entities well. This motivates the use of approximate discrete latent variables, which we will discuss later.

Model responses are shown in Table 5. In general, the MrRNN responses are more coherent and topic-oriented compared to the VHRED, HRED and RNNLM responses, which usually produce very generic responses (Serban et al., 2016). In particular, the MrRNN with activity-entity representation appears to give more goal-driven instructions compared to the MrRNN with noun

Table 5: Ubuntu model examples. The arrows indicate a change of turn. The examples were chosen from a set of short, but diverse dialogues, in order to illustrate cases where different MrRNN models succeed in generating a reasonable response.

Context	**Response**
Hey guys what do you in general use for irc something ubuntu xchat or xchat-gnome ? → without -gnome. that is just cut down → you mean drop xchat-gnome and go with xchat ?	**MrRNN Act. -Ent.:** im using xchat right now **MrRNN Noun:** what is xchat-gnome ? **VHRED:** correct **HRED:** yes
when setting up rules with iptables command only writes changes this file " etciptables. rules "? i ask this so i can backup before messing anything → sudo iptables-save something . dat to backup your rules restore with sudo iptables-restore < something . dat	**MrRNN Act. -Ent.:** I don't see any reason why you need iptables to do that **MrRNN Noun:** are you using ubuntu ? **VHRED:** thx **HRED:** thanks
Hello I tried to install ubuntu studio but I get W : Failed to fetch <url >404 Not Found when apt-get update → the ppa you added is not online	**MrRNN Act. -Ent.:** try a different mirror **MrRNN Noun:** something **VHRED:** but I dont know how to add it **HRED:** how do I do that ?
hi community.. is there any difference between " /.bashrc " " /.bashrc. local " → yes, they're different files :-) the default /.bashrc sources /.bashrc. local → sorry I could undersatnd... If I write on terminal " gedit /.bashrc " it opens a file but how I open " /.bashrc. local " → " gedit /.bashrc. local "... → but it just open blank file nothing inside it	**MrRNN Act. -Ent.:** open up gedit /.bashrc, there is a file called " something " **MrRNN Noun:** the file is empty.. **VHRED:** it's in /.bashrc or /.bashrc **HRED:** what are you trying to do ?

representation (see examples 2-4 in Table 5).

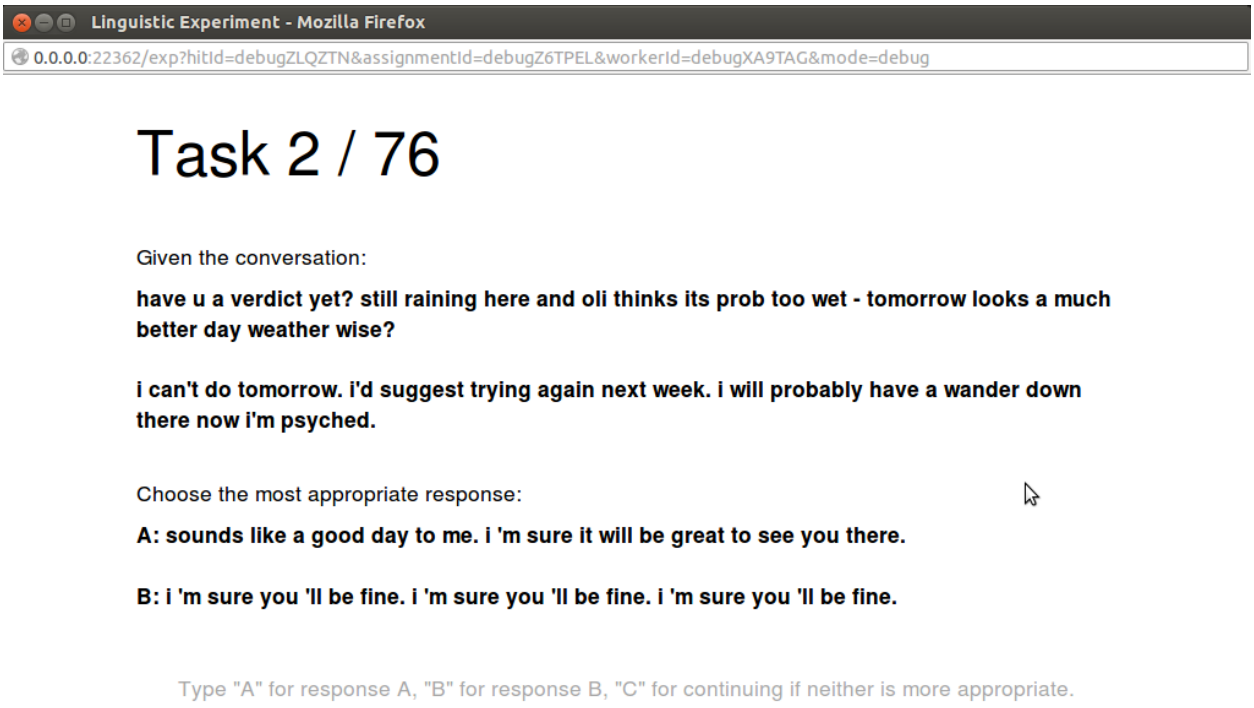

Figure 12: Screenshot of one dialogue context with two candidate responses, which human evaluators were asked to choose between.

3.4.5 Twitter Experiments

Evaluation Methods We carry out the human study for Twitter on Amazon Mechanical Turk (AMT). [15] We choose a crowdsourcing platform, because such experiments often involve a larger and more heterogeneous pool of annotators, which implies less cultural and geographic biases. Such experiments are also easier to replicate, which we believe is important for benchmarking future research on these tasks. We set up the evaluation study as a series of pairwise preference experiments. We show human evaluators a dialogue context along with two potential responses, one generated from each model conditioned on a given dialogue context. We ask participants to choose the response most appropriate to the dialogue context. If the evaluators are indifferent to

[15] We cannot conduct AMT experiments on Ubuntu as evaluating these responses usually requires technical expertise, which is not prevalent among AMT users.

either of the two responses, or if they cannot understand the dialogue context, they can choose "neither response". For each pair of models we conduct two experiments: one where the example contexts contain at least 80 unique tokens (*long context*), and one where they contain at least 20 (not necessarily unique) tokens (*short context*). This helps compare how well each model can integrate the dialogue context into its response, since it has previously been hypothesized that for long contexts hierarchical RNNs models fare better (Serban et al., 2016; Sordoni et al., 2015a). A screenshot is shown in Figure 12. Further details are given in Appendix II.

Next, we evaluate all proposed models using the three embedding-based similarity metrics proposed by (Liu et al., 2016) discussed earlier: *Embedding Average* (Average), *Embedding Extrema* (Extrema) and *Embedding Greedy* (Greedy). All three metrics are based on computing the textual similarity between the ground truth response and the model response using word embeddings. All three metrics measure a coarse form of topic similarity: if a model-generated response contains words semantically related to the ground truth response, then the metrics will yield a high score. This is a highly desirable property for dialogue systems on an open platform such as Twitter, however it is also substantially different from measuring the overall dialogue system performance, or the appropriateness of a single response. These performance measures require a human evaluation.

Finally, we quantify the amount of information content in the generated model responses by computing the response length (i.e. number of tokens) and the average unigram entropy. The unigram entropy is computed in bits on the preprocessed tokenized test set w.r.t. the maximum likelihood unigram model over the generated responses.

Since the work on the VHRED and MrRNN models was carried out in parallel, these experiments did not include the MrRNN models. However, following these experiments, we conducted a preliminary comparison between the VHRED and MrRNN models. Here, we asked one human evaluator to compare the candidate responses generated by the models to the ground truth response and dialogue context, and rate them for fluency and relevancy on a Likert-type scale 0-4. The setup was similar to that of the Ubuntu experiment. However, it only involved one human evaluator.[16]

Results The AMT pairwise preference experiments results are shown in Table 6. These results demonstrate that VHRED is clearly preferred in the majority of the experiments compared to the other models. In particular, VHRED is strongly preferred over the HRED and TF-IDF baseline models for both short and long context settings. VHRED is also preferred over the RNNLM baseline model for long contexts; however, the RNNLM is preferred over VHRED for short contexts. This is likely because the RNNLM baseline tends to output much more *generic* responses. Since it doesn't model the hierarchical input structure, the RNNLM model has a shorter memory span, and thus must output a response based primarily on the end of the last utterance. Such 'safe' responses

[16]The human evaluator was from outside the research lab.

Table 6: Wins, losses and ties (in %) of VHRED against baselines based on the human study (mean preferences $\pm$ 90% confidence intervals). The superscripts * and ** indicate statistically significant differences at 90% and 95% confidence level respectively.

	Opponent	Wins	Losses	Ties
Short Contexts	VHRED vs RNNLM	32.3 $\pm$2.4	**42.5 $\pm$2.6***	25.2 $\pm$2.3
	VHRED vs HRED	**42.0 $\pm$2.8***	31.9 $\pm$2.6	26.2 $\pm$2.5
	VHRED vs TF-IDF	**51.6 $\pm$3.3***	17.9 $\pm$2.5	30.4 $\pm$3.0
Long Contexts	VHRED vs RNNLM	**41.9 $\pm$2.2****	36.8 $\pm$2.2	21.3 $\pm$1.9
	VHRED vs HRED	**41.5 $\pm$2.8***	29.4 $\pm$2.6	29.1 $\pm$2.6
	VHRED vs TF-IDF	**47.9 $\pm$3.4***	11.7 $\pm$2.2	40.3 $\pm$3.4

are reasonable for a wider range of contexts, meaning that human evaluators are more likely to rate them as appropriate. However, a model that only outputs generic responses is undesirable for dialogue, as this leads to uninteresting and less engaging conversations. Conversely, the VHRED model is explicitly designed for long contexts and for outputting a diverse set of responses by sampling of the latent variable. Thus, the VHRED model generates longer sentences with more semantic content than the RNNLM model, which is confirmed by statistical metrics in Table 9. This can be 'riskier' as longer utterances are more likely to contain small mistakes, which can lead to lower human preference for a single utterance. However, we believe that response diversity is crucial to maintaining interesting conversations — in the dialogue literature, generic responses are used primarily as 'back-off' strategies in case the agent has no interesting response that is relevant to the context (Shaikh et al., 2010).

The above results are confirmed by qualitative assessment of the generated responses, shown in Table 7. VHRED generates longer and more meaningful responses compared to the RNNLM model, which generates mostly generic responses. Additionally, it was observed that the VHRED model has learned to better model smilies, slang (see first example in Table 7) and can even continue conversations in different languages (see fifth example).[17] Such aspects are not measured by the human pairwise preference study. Further, VHRED appears to be better at generating *stories* or *imaginative actions* compared to the generative baseline models (see third example). The last example in Table 7 is a case where the VHRED generated response is more interesting, yet may be less preferred by humans as it is slightly incompatible with the context, compared to the generic RNNLM response.

[17] There is a notable amount of Spanish and Dutch conversations in the corpus.

Table 7: Twitter model examples. The arrows indicates a change of turn. The examples were chosen from a set of short, but diverse dialogues, in order to illustrate cases where the VHRED model succeeds in generating a reasonable response.

Context	Response
jamaica blood!! <url > → lol. yes [name]!. [name]! → what's poppin in toronto homie	**VHRED:** nothing much. just chillin. what's good with you? **RNNLM:** something!!!!!!! **HRED:** something!!!!!!
whhhhhhyyyy is it suddenly cold in tallahassee today?? this is florida! how can i dream of the beach when my fingers are icey cold?? → sittin on deck in sun looking at lake travis in austin! enjoying day b4 birthday	**VHRED:** the sun is shining!!! **RNNLM:** i know!!!!!!!!! **HRED:** something!!!!!!!
reaches for [name]! → - approaches him warily - hey there buddy. - swings him into the air - how you doin? → tilts his head and your wariness and reaches for your face [name].	**VHRED:** - looks up at you and smiles - you're so cute. - kisses your cheek - **RNNLM:** - smiles - i'm fine thank you. how are you? **HRED:** - grins, looking at you as i look at you -
i think i want an ipad2. but i already have an ibook. want vs. need. → if you have an ibook. then you deserve an ipad!	**VHRED:** that's what i'm thinking. i'm going to buy a new one. **RNNLM:** i want one!!!! **HRED:** i don't know what to do with it. i don't know what to do.
ah echt? dan anrtwoord ik volgende keer via twitter als je me smst. → haha doe maar p. goed bezig hoor myrthe d → goed bezig? die snap ik even niet p	**VHRED:** haha, ik heb er geen zin in de something, maar ik heb er geen something meer something **RNNLM:** haha something something something **HRED:** something something something something something something something something
i love katie waissel more than like, anything in life. → i wanna c the show again! i need <number> meet her again! n i want a b-day tweet nxt week doubt that will happen x	**VHRED:** me too. i'm going to have to wait for her to come back. i'm so excited. **RNNLM:** i'm sure she will. i'm sure she'll be fine. **HRED:** i'm sure she'll be fine, i'm sure she'll be fine.
my math teacher over here talkn bout nickels nd i'm tryna make big money i need to be in calculus → lol	**VHRED:** i need to go to the gym and get my hair done lol **RNNLM:** lol **HRED:** how u been

Next, the automatic embedding-based evaluation metrics are shown in Table 8. In agreement with the previous discussion and our qualitative observations, the VHRED model responses score higher on these metrics compared to the HRED and RNNLM models. This indicates that VHRED is better capable of staying on topic on the Twitter domain.

Finally, the results of the preliminary experiment comparing MrRNN to the other models is given in Table 10. The results in this table should be interpreted with caution, as they involve only a single annotator. Nevertheless, by these results, it appears that the MrRNN model performs worse than all other models considered. This could indicate a problem caused by not selecting appropriate preprocessing or hyperparameters for the MrRNN model on Twitter. Investigating and quantifying these differences is part of future work.

Table 8: Twitter evaluation using embedding metrics (mean scores $\pm$ 95% confidence intervals)

Model	**Average**	**Greedy**	**Extrema**
RNNLM	51.24 ± 0.51	38.9 ± 0.39	36.58 ± 0.36
HRED	50.1 ± 0.52	37.83 ± 0.4	35.55 ± 0.37
VHRED	$\mathbf{53.26 \pm 0.45}$	$\mathbf{39.64 \pm 0.34}$	$\mathbf{37.98 \pm 0.32}$

Table 9: Twitter response information content on 1-turn generation as measured by average utterance length $|U|$, word entropy $H_w = -\sum_{w \in U} p(w) \log p(w)$ and utterance entropy H_U with respect to the maximum-likelihood unigram distribution of the training corpus p.

Model	$\lvert U \rvert$	H_w	H_U
RNNLM	11.21	6.75	75.61
HRED	11.64	6.73	78.35
VHRED	**12.29**	**6.88**	**84.56**
Human	20.57	8.10	166.57

Table 10: Twitter human evaluation w.r.t. fluency and relevancy scores by rating category.

	Fluency (0-4)					**Relevancy (0-4)**				
Rating Level / **Model**	0	1	2	3	4	0	1	2	3	4
RNNLM	0	0	4	1	34	7	6	5	7	14
HRED	1	1	8	1	28	11	4	7	1	16
VHRED	0	0	2	1	36	4	2	4	4	25
MrRNN	2	0	12	4	21	16	7	2	6	8

3.5 Discussion

In this first part of the thesis, three different probabilistic generative models were proposed. First, we proposed the Hierarchical Recurrent Encoder-Decoder (HRED) model, which incorporated the turn-taking structure of dialogue into its architecture in order to better model discourse-level context. Second, we proposed the Multiresolution Recurrent Neural Network (MrRNN) model. This is a stacked sequence-to-sequence model with an intermediate, stochastic representation (a "coarse representation") capturing the high-level semantic content of the dialogue. Third, we proposed the Latent Variable Recurrent Encoder-Decoder (VHRED) model, which is a variant of the HRED model with a continuous, stochastic, latent variable for modelling the ambiguity and uncertainty in human language communication.

We evaluated all these models extensively on two domains: the goal-driven technical response generation task on Ubuntu, and the non-goal-driven response generation task on Twitter. We evaluated all models w.r.t. human evaluation studies, on a crowdsourcing platform and in a laboratory setting, a qualitative evaluation of the responses and automated evaluation metrics. Here, each model was compared against multiple baseline models.

The experiment results show that all three proposed models have their own merits. The HRED model was found to outperform a recurrent neural network language model (RNNLM) with an LSTM gating function (RNNLM) on the Ubuntu domain, where it generated relevant responses of higher quality incorporating more dialogue context. The results here suggests that it is able to better capture discourse-level context. The MrRNN model was found to perform best among all models on the Ubuntu domain, where it outperformed the RNNLM, HRED, VHRED and another informed baseline model. Here, MrRNN was able to generate substantially more relevant and fluent responses compared to the other models. For example, in comparison to the RNNLM model, it obtained nearly an order of a magnitude better performance w.r.t. generating responses with appropriate *activities* on the Ubuntu domain. On the same metric, MrRNN obtained two to three times better performance compared to the HRED, VHRED and the informed baseline model. These results clearly indicate that MrRNN is a promising approach to goal-driven response generation on the Ubuntu domain, as of this writing. This confirms our earlier hypothesis on the importance of modelling high-level abstractions of semantic content in a stochastic framework. At the same time, the VHRED model obtained the best performance on the Twitter domain. Here, the VHRED model was significantly preferred over the RNNLM model, the HRED model and a TF-IDF baseline model In particular, the VHRED model appeared to be better capable of generating long and semantically coherent responses compared to other models. As such, VHRED appears to be a promising approach for non-goal-driven response generation on the Twitter domain. This confirms our other earlier hypothesis on the importance of modelling high-level latent structure in dialogues

through a stochastic generation processes. It also underlines the importance of building dialogue models, which explicitly model uncertainty and ambiguity.

However, it is important to discuss some of the major limitations and issues of the research conducted in this part of the thesis. A major limitation of the results presented is that the experiments only consider the evaluation of a single, next response in a dialogue. It is unclear how the models would perform and compare to each other if they were to conduct a complete dialogue. A closely related issue arises from the fact that the human evaluators provide annotations for single responses in a dialogue, without taking into account the past or the future of each dialogue. In addition, human evaluators do not have the same incentives as real-world users interacting with a dialogue system. For example, we observed that human evaluators had a strong tendency to prefer short, generic responses on the Twitter task. However, such responses are unlikely to maintain the engagement of real-world users. Consequently, the human annotations might not be representative of the appropriateness or utility in a real-world dialogue between a human user and a dialogue system. A different but important issue is caused by the high variance of the model training procedures. All the models presented are sensitive to their parameter initialization, hyperparameter configuration and random seed of the sampling procedures. Although this issue is mitigated to some extent by experimenting with multiple hyperparameter combinations, it still constitutes a significant confounding factor in the experiments presented. Adjusting the hyperparameters and rerunning the experiments could potentially have a significant impact on the results.

For each proposed model, we have already discussed relevant prior work in the literature. However, since the work in this chapter was completed and published, other researchers have conducted related work. It is beyond the scope of this thesis to cover all of the later related work, but we will briefly mention some of the major lines of research. Many researchers have focused on improving the neural network architecture, while keeping the same probabilistic generation process as the RNNLM and HRED models. For example, Bordes et al. (2016) propose to augment the neural network architecture with a memory component, in order to capture long-term discourse-context and external knowledge. As another example, Mei et al. (2017) propose to incorporate an attention mechanism into the model architecture. In another related line of research, researchers have proposed to modify the objective function of the neural network model. For example, Li et al. (2016a) propose to replace the well-established maximum log-likelihood objective function with an objective function which encourages diversity in the generated model responses. As another example, Li et al. (2017) propose to use an adversarial training loss. In yet another example, Lowe et al. (2017a) propose to learn an neural network for evaluating model responses, based on human annotations, which might then be utilized as the objective function for the generative neural network model. Other researchers have focused on improving the architecture with stochastic latent variables, such as the models proposed by Zhao et al. (2017) and Cao and Clark (2017). Last,

but not least important, researchers have also begun incorporating external knowledge sources into dialogue systems. For example, researchers have experimented with injecting knowledge ranging from software manuals to Wikipedia and common sense knowledge bases (Lowe et al., 2015a; Parthasarathi and Pineau, 2018; Dinan et al., 2019; Young et al., 2017).

Following the work presented in this chapter, the author of this thesis and his colleagues have also proposed an extension of the VHRED model involving continuous latent variables with piecewise constant structures. These latent variables are able to represent far more complex probability distributions, including multi-modal distributions, which may help to better model many aspects of real-world dialogue. The new model was applied to both tasks discussed in this chapter and then evaluated using automated evaluation metrics. Based on these experiment results, the new model did not perform substantially better than the VHRED model.[18] However, the new model was able to better generate entity phrases suggesting it may be useful for applications involving a large number of entities and a high amount of uncertainty and ambiguity. The reader is referred to the conference publication Serban et al. (2017a) for details.

[18] A model involving continuous latent variables with piecewise constant structures was also applied to three document modelling tasks. On all three tasks this model yielded state-of-the-art results.

3.6 Directions for Future Research

In this section, we will discuss several avenues for future research.

3.6.1 Hierarchical Models with Stochastic Latent Dynamics

In the latent variable model VHRED, the latent variables were assumed to be independent of each other given the observed tokens. This is a highly restrictive assumption, because it limits the effect of the latent variables to a single dialogue utterance. It seems quite likely that for many tasks there exists latent variables which are relevant across long time spans. For example, user goals should clearly be modelled as long-term latent variables in any task-driven dialogue model. As another example, topics should also clearly be modelled as long-term latent variables. This motivates building generative latent variable models, where the latent variables affect each other across utterances or, more generally, across sequences.

Models where the latent variables affect each other across time are said to have *latent dynamics*. Similar to the hidden Markov model (HMM) discussed in the beginning, the latent variables form a trajectory in latent space, which should capture the dependencies between the observed sequences.

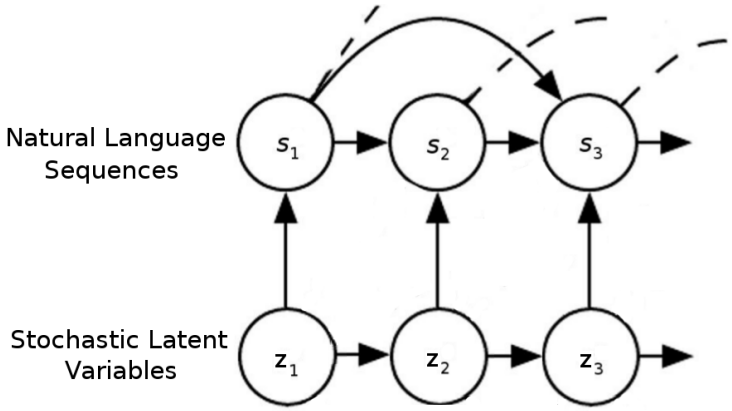

Figure 13: Probabilistic directed graphical model for Latent Variable Recurrent Encoder-Decoder RNN with stochastic latent dynamics.

A natural approach is to extend the VHRED model, such that the latent variables depend on the latent variables at the immediately preceding time step thereby maintaining the Markov property. Let $(\mathbf{w}_1, \ldots, \mathbf{w}_N)$ be a sequence consisting of N constituent sequences, where $\mathbf{w}_n = (w_{n,1}, \ldots, w_{n,M_n})$ is the n'th constituent sequence and $w_{n,m} \in V$ is the m'th discrete token in that sequence. Define stochastic latent variables $\mathbf{z}_n \in \mathbb{R}^{d_z}$ for each constituent sequence $n = 1, \ldots, N$ conditioned on all previous observed tokens. Given $\mathbf{z}_n$, the new model we propose generates the

n'th constituent sequence tokens $\mathbf{w}_n = (w_{n,1}, \ldots, w_{n,M_n})$ as:

$$P_\theta(\mathbf{z}_n \mid \mathbf{z}_{n-1}, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \mathcal{N}_{\mathbf{z}_n}(\boldsymbol{\mu}_{\text{prior}}(\mathbf{z}_{n-1}, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}), \Sigma_{\text{prior}}(\mathbf{z}_{n-1}, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1})), \tag{88}$$

$$P_\theta(\mathbf{w}_n \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \prod_{m=1}^{M_n} P_\theta(w_{n,m} \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, w_{n,1}, \ldots, w_{n,m-1}), \tag{89}$$

where θ are the model parameters and $\mathcal{N}_{\mathbf{z}_n}(\boldsymbol{\mu}, \Sigma)$ is the multivariate normal distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^{d_z}$ and covariance matrix $\Sigma \in \mathbb{R}^{d_z \times d_z}$, constrained to be a diagonal matrix, which is followed by the stochastic variable $\mathbf{z}_n$. This model corresponds to the graphical model shown in Figure 13. In particular, we might further define the prior distribution as:

$$\boldsymbol{\mu}_{\text{prior}}(\mathbf{z}_{n-1}, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = W^z_{\text{prior}} \mathbf{z}_{n-1} + \hat{\boldsymbol{\mu}}_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}), \tag{90}$$

$$\Sigma_{\text{prior}}(\mathbf{z}_{n-1}, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \hat{\Sigma}_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}), \tag{91}$$

where $W^z_{\text{prior}} \in \mathbb{R}^{d_z \times d_z}$, and where $\hat{\boldsymbol{\mu}}$ and $\hat{\Sigma}$ are defined as in the former VHRED model. In this parametrization, the mean of $\mathbf{z}_n$ depends linearly on $\mathbf{z}_{n-1}$, which is similar to the Kalman filter model. Furthermore, the magnitude of its uncertainty in different directions, represented by the covariance matrix, does not depend at all on $\mathbf{z}_{n-1}$. To simplify the model, W^z_{prior} could further be fixed to the identity matrix. Depending on $\hat{\boldsymbol{\mu}}_{\text{prior}}$, this might correspond to the assumption that the distribution of $\mathbf{z}_n$ should be centered $\mathbf{z}_{n-1}$.

To train this model, the variational lower-bound could be applied as in the VHRED model. The approximate posterior could be defined as for the VHRED model, or it could optionally be conditioned on additional future utterances (since the current location will affect the remaining part of the trajectory). This parametrization would encourage the model to find a smooth trajectory in latent space, which can explain the dependence between the observed utterances.

Inspired by models such as the restricted Boltzmann machine and deep directed graphical models (Goodfellow et al., 2016), it is likely that a hierarchy of stochastic latent variables may help model natural language dialogues. Instead of one sequence of latent variables $(\mathbf{z}_1, \ldots, \mathbf{z}_M)$, there could be two sequences of latent variables $(\mathbf{z}_1, \ldots, \mathbf{z}_M)$ and $(\mathbf{r}_1, \ldots, \mathbf{r}_M)$, where $\mathbf{z}_m, \mathbf{r}_m \in \mathbb{R}^{d_h}$, such that the sequence $\mathbf{z}_m$ is conditioned on $\mathbf{r}_m$, for $m = 1, \ldots, M$. One such possible model is illustrated in Figure 14. In this case, we would say that the model possess deep stochastic latent dynamics. The hierarchical ordering of the latent variables could possibly encourage the sequence $(\mathbf{r}_1, \ldots, \mathbf{r}_M)$ to capture long-term temporal dependencies and let the sequence $(\mathbf{z}_1, \ldots, \mathbf{z}_M)$ capture shorter term dependencies. Due to the increased depth of the model, the gradient signal will be affected by the additional computational steps and the effect of compounding noise. This will likely make the model more difficult to train.

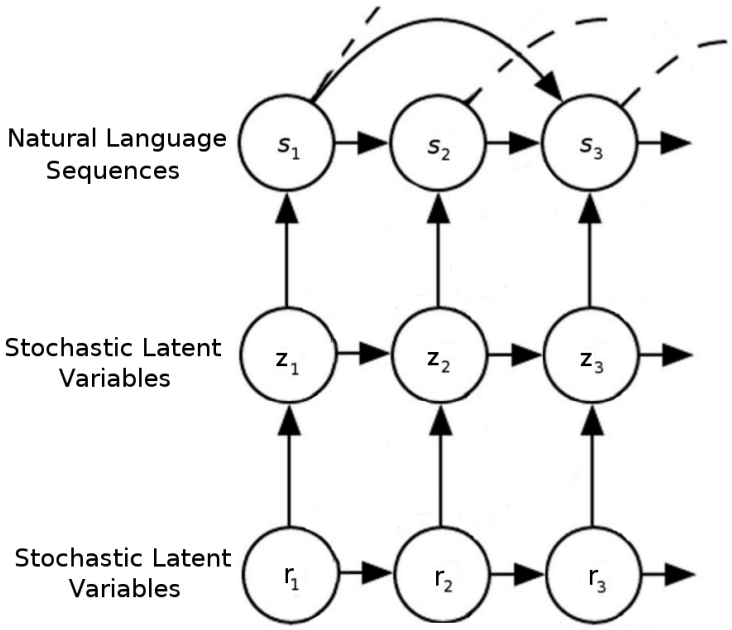

Figure 14: Probabilistic directed graphical model for Latent Variable Recurrent Encoder-Decoder RNN with deep stochastic latent dynamics.

Another possible extension is to incorporate structure specific to the generative dialogue modelling problem. Each participant in a dialogue could be assigned a separate latent variable, with the hope that this variable would represent specific information about that participant, such as the participant's goal or the speaking style. This might be particularly useful for dialogues with multiple or returning participants. This latent variable would be initialized randomly the first time a new participant is encountered and updated with new information as the system interacts with the user. This is related to the work of (Li et al., 2016b), which attempts to model the dialogue participants in a deterministic framework. However, due to the scarce participant-specific data and the large amount of ambiguity in natural language dialogue, it may be beneficial to model dialogue participants using stochastic latent variables.

3.6.2 End-to-end Multiresolution RNNs

In the MrRNN model, the coarse (high-level) sequences were extracted a priori using a simple procedure. This allowed us to train the model using the exact joint log-likelihood. However, for many real-world applications it may not be possible to build a procedure to extract useful high-level sequences capturing semantic information pertinent to the domain. Therefore, an obvious extension of the MrRNN model is to treat the coarse sequence tokens as discrete latent variables and optimize the model parameters w.r.t. the marginal log-likelihood over the observed natural language sequences.

One way to train such a model this might be to approximate the marginal log-likelihood with Markov chain Monte Carlo sampling (Goodfellow et al., 2016, Ch. 17). However, we need to consider that the dimensionality of the latent variables may be huge for real-world applications (for example, in the Ubuntu and Twitter experiments the MrRNN models had thousands of parameters). In this case, such an approach would require sampling many variables from an approximate posterior distribution and might therefore not be effective in practice.

An alternative approach to train such a model might be to work at the word embedding level of the generated coarse tokens. Suppose a pretrained MrRNN model is given. We may extend this model with a separate GRU or LSTM-gated RNN, which is conditioned on the hidden state of the *context* RNN in the natural language sub-model. This RNN is then tasked with predicting the sequence of word embeddings corresponding to the embeddings given as input to *coarse prediction encoder* of the MrRNN model. As such, the parameters of the RNN are optimized to minimize the squared error between the predicted embeddings and the ground truth embeddings. Once it has been trained to convergence, a new end-to-end MrRNN model may be trained to maximize the marginal log-likelihood without the use of the coarse sequences. The new model is initialized from the previous MrRNN model, where the *coarse prediction encoder* now takes as input the embeddings predicted by the RNN described earlier. This eliminates the discrete coarse sequences completely. Since the coarse embeddings are not tied to specific discrete tokens, the model could potentially learn new aspects of the data distribution, for example, activities or entities not included in the hand-crafted extraction procedure.

4 A Deep Reinforcement Learning Dialogue System

4.1 Author's Contribution

The work in this chapter covers the work published in the publication:

"The Octopus Approach to the Alexa Competition: A Deep Ensemble-based Socialbot"
by Iulian V. Serban, Chinnadhurai Sankar, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Sarath Chandar, Nan Rosemary Ke, Sai Rajeswar, Alexandre de Brebisson, Jose M. R. Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen and Yoshua Bengio, Alexa Prize Proceedings, 2017.

This publication can be accessed at: http://alexaprize.s3.amazonaws.com/2017/technical-article/mila.pdf. The work in this chapter also covers the work in the following two pre-print articles:

"A Deep Reinforcement Learning Chatbot"
by Iulian V. Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Rajeshwar, Alexandre de Brebisson, Jose M. R. Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau and Yoshua Bengio, arXiv:1709.02349, 2017.

"The Bottleneck Simulator: A Model-based Deep Reinforcement Learning Approach"
by Iulian Vlad Serban, Chinnadhurai Sankar, Michael Pieper, Joelle Pineau and Yoshua Bengio, arXiv:1807.04723, 2018.

The two pre-print articles may be accessed at: https://arxiv.org/abs/1709.02349 and https://arxiv.org/abs/1807.04723.

As we will discuss later, the author of this thesis led a team of researchers, including students, professors and other staff members from the University of Montreal, to participate in the Amazon Alexa Prize 2017 competition. All members of the team worked together in order to develop both the system and underlying models, as well as to execute the experiments discussed later. The author of this thesis did the majority of work related to developing the response selection policies, described below. Here, Chinnadhurai Sankar and Sai Rajeshwar implemented the two reinforcement learning policies based on the REINFORCE algorithm with help from the author of this thesis. The author also did the majority of the work related to designing and executing the A/B testing experiments, as well as the writing up of the publication and the two pre-print articles, with help and feedback from his co-authors. Michael Pieper carried out the experiment analyzing why users terminated the dialogue with help from the author of this thesis.[19] The other co-authors of

[19] In the second pre-print article, Chinnadhurai Sankar implemented and executed the experiments related to the text

the publications listed above contributed to the competition by collecting and annotating datasets, designing and implementing nearly two dozen response models, and helping to set up the server infrastructure and system pipeline.

4.2 Motivation

In the first part of this thesis, we explored different approaches to building generative dialogue models from corpora of text-based dialogues. Given a sequence of turns from a text-based dialogue, these models were given the task of generating the next, appropriate response. Although there is still much research left to be completed in this area, some of the central assumptions imposed by this framework may be limiting the scope of real-world applications of these models. Perhaps one of the most critical limitations arise from the assumption that generative dialogue models, applicable to real-world problems, may be learned exclusively from recorded text dialogues. As discussed earlier, this type of learning approach without external information may pose a significant problem for the model's ability to ground natural language (Harnad, 1990; Quine, 2013). Another assumption in the previous framework is that it is sufficient for the models to be trained to mimic the conversations between humans. This implies a type of symmetric relationship between the interlocutors, where any two interlocutors could be switched if their internal states and goals were switched as well. However, this is unlikely to be the case when a human converses with a machine. Humans have different expectations and behave differently when they converse with machines compared to when they converse with other humans. Researchers have explored this phenomenon extensively in *Wizard of Oz* experiments, where a dialogue system is being operated by a human controller. See, for example, Dahlbäck et al. (1993). Even if it were possible to solve the previous two issues (for example by training a model on conversations between humans machines and by including external information to facilitate the grounding the natural language grounding process), there exists a potential third issue. In the previous framework, the models are trained with a myopic learning signal. Here, the models are only optimized w.r.t. the immediate next turn in the dialogue.[20] This might potentially induce the model to generate responses meaningful only on a short time horizon, which could be catastrophic for many real-world applications. For example, consider the example of a dialogue system selling flight tickets. Suppose that this system has been trained on human-machine conversations and, as a result, has learned a highly myopic policy. If a human interlocutor were to ask "Which flights are flying from Montreal to New York tomorrow?", such a myopic system might utter a list of flights and then proceed to end

adventure game *Home World*. These experiments are not discussed in this thesis.

[20] It should be noted that given an unlimited amount of data and a model with sufficient capacity, it may be possible to learn an optimal policy from this training signal.

the conversation with “Thank you. Have a pleasant day!" without completing the sale.

Motivated by these issues, in this second part of the thesis, we investigate a different framework for building dialogue systems based on combining representation learning and reinforcement learning. In particular, we focus on building a non-goal-driven dialogue system, which learns from its own real-world interactions with human interlocutors.

The work presented here started in late 2016, when Amazon.com Inc. proposed an international university competition with the goal of building a socialbot (Ram et al., 2017). A socialbot is a spoken dialogue system capable of conversing coherently and engagingly with humans on popular topics, such as entertainment, fashion, politics, sports, and technology. In this competition, socialbots conversed through natural language speech using Amazon’s Echo device (Stone and Soper, 2014). In order to participate in this competition, researchers at University of Montreal and the author of this thesis assembled together a team. Our team members were as follows: Chinnadhurai Sankar, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Sarath Chandar, Nan Rosemary Ke, Sai Rajeswar, Alexandre de Brebisson, Jose M. R. Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Yoshua Bengio and the author of this thesis. The author of this thesis was chosen as the team leader. Our team wrote up an application to participate in the competition and submitted it to Amazon.com Inc.. The application was accepted at the end of 2016, and shortly after our team began working on building the dialogue system.

Our team considered the competition important, because it provided a special opportunity for training and testing state-of-the-art approaches from representation learning and reinforcement learning with real-world users in a comparatively unconstrained setting. In the field of machine learning, this type of setup is also known as *machine learning in the wild*. The ability to experiment with real-world users is rare in the machine learning community. As of this writing, the only formal experiment framework for evaluating any machine learning system involving human participants at the Quebec Artificial Intelligence Institute (Mila) are crowdsourcing platforms, such as Amazon Mechanical Turk and Figure Eight.[21] Unfortunately, the human participants on these platforms are not representative of real-world users and are constrained w.r.t. the amount of instruction and time-on-task available. As such, it is not surprising that the majority of research in the machine learning community involves experiments on fixed datasets (e.g. labelled datasets) and software simulations (e.g. game engines). In addition to providing the framework for real-world user evaluation, Amazon Inc. also provided the team with computational resources (e.g. cloud computing server instances), technical support and financial support, which helped to scale up the models developed by our team and test the limits of state-of-the-art methods.

Our team set out to to build the socialbot, called *Milabot*, as a large-scale ensemble system based on deep learning and reinforcement learning. The ensemble architecture brought several

[21] See www.mturk.com and www.figure-eight.com.

advantages with it, including the promise of making the system capable of conversing on a wide range of topics and the ability for groups of researchers to work more independently on their own models and algorithms. As will be discussed later, our team developed a new set of deep learning models for natural language retrieval and generation. These models were combined into an ensemble system. Given a text-based dialogue history (i.e. a sequence of utterances between the interlocutors), this ensemble system generates a set of candidate responses. Following this, the system considers all the candidate responses and selects one of them as the system's response. The module responsible for selecting the response is called the *response selection policy*. This policy may be learned based on crowdsourced data and based on interactions recorded between real-world users and the system in order to select the most appropriate response. This part of the thesis will focus on different approaches for learning the response selection policy and on the experiments relevant to it. Apart from managing the team and designing the ensemble system architecture, the main contribution of the author of this thesis lies in the implementation of different response selection policies and in the execution of the experiments with real-world users.

This chapter deals with a new framework combining deep learning and reinforcement learning, within an ensemble system, in order to learn from real-world interactions with human interlocutors. This brings with it a new set of open research questions. How can representation learning and reinforcement learning be combined to build a non-goal-driven dialogue system operating in a complex real-world domain? How can representation learning and reinforcement learning be utilized in order to improve the ability of a non-goal-driven dialogue system to initiate and maintain engaging conversations with human interlocutors? What reinforcement learning algorithms are suitable for learning an effective response selection policy? Can model-based reinforcement learning be applied to learn a response selection policy? How do different reinforcement learning algorithms influence the qualitative behaviour of the response selection policy? How many examples are needed and how does the sample efficiency of an algorithm affect the system's overall performance? How can probabilistic generative models be applied to learn an effective response selection policy? How is the qualitative behaviour of such a policy different from policies learned using other reinforcement learning algorithms? How does the sample efficiency of such an approach compare to other algorithms? Finally, how can such a system be evaluated effectively? What are the appropriate evaluation metrics, and how do they reflect the behaviour of the system and the subjective experience of human interlocutors?

4.3 Prior Related Work

Next we discuss some of the related work conducted prior or in parallel to the research in this chapter. Most real-world dialogue systems consist of a number of different modules, such as modules for querying databases, modules for querying external web-services, modules for handling daily chitchat conversations, and so on. In this respect, Milabot is similar to many existing dialogue system architectures (Bohus et al., 2007; Prylipko et al., 2011; Suendermann-Oeft et al., 2015; Zhao et al., 2016; Miller et al., 2017; Truong et al., 2017). These systems encapsulate individual modules into black boxes sharing the same interface. This modular design makes it easy to control each module through an executive component, such as a dialogue manager. This is similar to the response models in Milabot, where each response model takes the same input (a dialogue history) and outputs a candidate response, with the response selection policy deciding the final response.

There has been a lot of research applying reinforcement learning to training or improving dialogue systems. As discussed earlier, the idea of optimizing the behaviour of a dialogue system by formulating the problem as a sequential decision making problem appeared already in the 1990s for goal-driven dialogue systems (Singh et al., 1999, 2002; Williams and Young, 2007; Young et al., 2013; Paek, 2006; Henderson et al., 2008; Pieraccini et al., 2009; Su et al., 2015).

One highly relevant line of research is optimizing dialogue systems through simulations using abstract dialogue states and actions (Eckert et al., 1997; Levin et al., 2000; Chung, 2004; Cuayáhuitl et al., 2005; Georgila et al., 2006; Schatzmann et al., 2007; Heeman, 2009; Traum et al., 2008; Georgila and Traum, 2011; Lee and Eskenazi, 2012; Khouzaimi et al., 2017; López-Cózar, 2016; Su et al., 2016; Fatemi et al., 2016; Asri et al., 2016). The approaches taken here vary with how the simulator is created or estimated, and whether or not the simulator is also considered an agent trying to optimize its own reward. For example, Levin et al. (2000) consider the problem of building a flight booking dialogue system. Similar to an n-gram language model, Levin et al. (2000) estimate a user simulator model by counting transition probabilities between dialogue states and user actions. This simulator is then used to train a reinforcement learning policy. In their work, the states and actions are all abstract discrete variables, which minimizes the amount of natural language understanding and generation the policy has to learn. This is similar to the work in this chapter, where we explore estimating a simulated environment model utilizing a high-level stochastic variable representing abstract semantic information. In a related example, Georgila and Traum (2011) consider the problem of learning dialogue policies for negotiation games, where each interlocutor is an agent with its own reward function. In their work, each policy is also a de facto user simulator. These policies are learned by playing against other policies using reinforcement learning. In a more recent example, Yu et al. (2016) propose an open-domain, chitchat dialogue system optimized using reinforcement learning based on simulations with the rule-based system

ALICE (Wallace, 2009). In their work, the reward function is learned from crowdsourced human annotations. Yu et al. (2016) show that their system achieves substantial improvements w.r.t. the overall appropriateness of system responses and the conversational depth of the dialogues.

Researchers have also begun to investigate learning probabilistic generative neural network model policies operating directly on raw text through user simulations (Li et al., 2016c; Das et al., 2017; Lewis et al., 2017; Liu and Lane, 2017; Lewis et al., 2017). In contrast to earlier work, these policies are required to simultaneously perform natural language understanding and natural language generation. For example, Li et al. (2016c) train a probabilistic generative recurrent neural network using maximum log-likelihood, and then fine-tune it with a multi-objective function. The multi-objective function contains several terms, including a reinforcement learning term based on self-play simulation rollouts with a hand-crafted reward function. In another example, Lewis et al. (2017) apply reinforcement learning for learning a system capable of negotiation in a toy domain. They demonstrate that it's feasible to learn an effective policy by training a probabilistic generative recurrent neural network on crowdsourced data, and that the policy can be further improved using reinforcement learning through self-play and simulation rollouts. Self-play is a sensible approach for both Li et al. (2016c) and Lewis et al. (2017), because their problems are likely to be symmetric, or close to symmetric, w.r.t. the agent policies (for example, a policy performing well on one side of the negotiation game will likely also perform well on the other side). However, as discussed earlier, the interactions between humans and machines are unlikely to be symmetric. Humans are very likely to interact differently with Milabot compared to how they would interact with other humans. Therefore, self-play is unlikely to be an effective training method for the dialogue task considered in this chapter. In another example, Liu and Lane (2017) propose to use reinforcement learning to improve a system in a restaurant booking toy domain. For training the system policy, they use a user simulator trained on real-world dialogues between human interlocutors. By imposing the constraint that the system and the user share the same reward function, they demonstrate that reinforcement learning can be used to improve both the system policy and the user simulator. In a related example, Zhao and Eskenazi (2016) propose to train a neural network model for playing a quiz game using reinforcement learning, where the environment is a game simulator. In parallel to the work in this chapter, Peng et al. (2018) propose to train a dialogue system using a model-based reinforcement learning approach based on the Dyna-Q algorithm.

Finally, it should be noted that researchers have also applied reinforcement learning for training agents to communicate with each other in synthetic multi-agent environments (Foerster et al., 2016; Sukhbaatar et al., 2016; Lazaridou et al., 2016, 2017; Mordatch and Abbeel, 2018).

4.4 System Overview

The Milabot system architecture is inspired by the success of ensemble-based machine learning systems. One example of such a system is the winning system of the Netflix Prize competition (Koren et al., 2009), which utilized hundreds of machine learning models in order to predict user movie preferences. Another example is the IBM Watson system (Ferrucci et al., 2010), which applied a multitude of machine learning algorithms to win the quiz game *Jeopardy!* in 2011. More recently, Google released an article demonstrating substantial improvements on machine translation by utilizing an ensemble-based system (Wu et al., 2016).

Milabot consists of an ensemble of response models. The response models are given a dialogue history as input and must output a response in natural language text. In addition to the text response, the response models may also output one or several real-valued scalars indicating their internal confidence. The overall purpose of the ensemble is to provide a diverse set of responses across many topics. As such, each response model is designed to generate responses on one or several specific topics using its own unique approach.

The *dialogue manager* component is responsible for invoking the appropriate system modules and emitting the final response of the system. The dialogue manager receives as input the dialogue history (i.e. all utterances recorded in the dialogue so far, including the current user utterance) and confidence values of the speech recognition system. The dialogue manager follows a three-step procedure to generate a system response. First, it invokes all the response models in order to generate a set of candidate responses. Second, if the set of candidate responses contains a *priority* (i.e. a response which takes precedence over other responses), then this response will be emitted by the system.[22] Third, in case there are no *priority responses*, then a response will be selected by the *response selection policy*. For example, the response selection policy may score all the candidate responses and select the highest-scored response to emit. See Figure 7.

When the speech recognition confidences are below a pre-specified threshold, the system asks the user to repeat their last utterance. Otherwise, the system does not utilize the speech recognition confidences. However, the ASR system is far from perfect. Therefore, it is very probable that the system might be substantially improved by improving the ASR system and by conditioning the response models and the response selection policy on the ASR confidences.

Two demo videos of the Milabot system are available at `https://youtu.be/TCVbYpu9Llo` and at `https://youtu.be/LG482LzW77Y`.

[22] For example, if a user asked *"What is your name?"*, the ensemble would emit the candidate response *"I am an Alexa Prize socialbot"* labelled as a *priority response*.

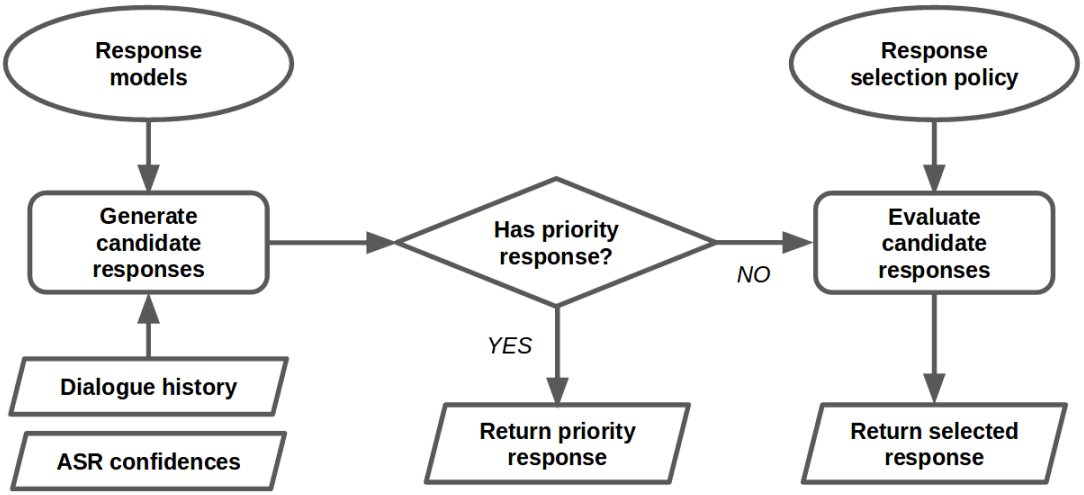

Figure 15: Dialogue manager control flow.

4.5 Response Models

The ensemble system consists of 22 response models, including retrieval-based neural networks, generation-based neural networks, knowledge base question answering systems and rule-based models. Examples of candidate model responses are shown in Table 11. Here, we will briefly review the most important response models. Further details are given in Appendix IV.

The ensemble contains several rule-based response models. One of the most important ones is *Alicebot* based on the rule-based system ALICE (Wallace, 2009; Shawar and Atwell, 2007). ALICE uses thousands of AIML (artificial intelligence markup language) templates to produce a response given the dialogue history and user utterance. Consequently, ALICE has some type of relevant response for the majority of dialogues and is able to converse on many different topics. However, ALICE is not particularly adept at tracking the discourse-level context (e.g. staying on topic or memorizing previous information about the user). ALICE is also prone to repeating itself. Another important rule-based response model is *Elizabot*, which performs string matching to select an answer from a set of template answers. *Elizabot* is based on a clone of the famous Eliza system, designed to mimic a Rogerian psychotherapist (Weizenbaum, 1966). As such, most of *Elizabot*'s responses are personal questions, which are meant to engage the user to continue the conversation, such as *"Is something troubling you?"* and *"Can you think of a specific example?"*. A related response model is the *Initiatorbot* model, which acts as a "conversation starter". This model selects an open-ended question from a list of questions, in order to get the conversation started and increase the engagement of the user. In many cases, the open-ended questions also suggest a particular topic. Some example questions the model emits are *"What did you do today?"*, *"Do you have pets?"* and *"What kind of news stories interest you the most?"*.

The ensemble also contains a question-answering model called *Evibot*, which forwards the user's utterance to Amazon Inc.'s question-answering web-service *Evi*.[23] *Evibot* is primarily capable of handling factual questions, such as *"How many people live in Greenland?"* and *"What do bears eat?"*. Therefore, *Evibot* returns a priority response when a direct question is detected (i.e. where a user utterance contains a wh-word, such as *"who"* and *"what"*).

There are also several retrieval-based models inside the ensemble system. The response model *BoWFactGenerator* retrieves a response from a set of about 2500 *interesting* and *fun facts*, including facts about animals, geography and history, based on Word2Vec word embeddings (Mikolov et al., 2013b). For example, given the user utterance *"I love dogs"!* the model will emit the response *"Here's a funny fact! Dogs have lived with humans for over 14,000 years."*. These fun fact responses serve as an excellent diversion tactic to keep a conversation engaging when the user starts a new topic or asks a question that cannot be handled by other response models. Another retrieval-

[23] Previously, Evi was available at: www.evi.com.

based response model is the *VHREDSubtitles* model, which uses a retrofitted VHRED model in order to retrieve responses from a corpus of movie subtitles. The ensemble system has several other neural network retrieval-based models, which retrieve responses from external sources such as *Google Search*, the American newspaper *The Washington Post*[24], and Reddit forums[25].

Finally, the ensemble system has a RNNLM called *GRUQuestionGenerator*, which was trained on Reddit forum discussions and generates question responses word-by-word using beam search (Graves, 2012). The aim of this model is to keep the conversation going by asking a topically-relevant question. For example, given the user utterance *"I like The God Father."* the model will emit the response *"How about the American people you know?"*.

Our team decided not to pursue developing additional generative probabilistic models, which generate responses word-by-word. There were two main reasons for this decision. The first reason was due to their overall complexity and lack of readiness for real-world applications. Implementing generative probabilistic models often requires weeks of development, training time and careful tweaking. Although generative probabilistic models have the power to generate entirely new responses, in practice they often generate nonsensical responses which need to be mitigated by other mechanisms. The second reason was because of their computational time requirements. The Amazon Alexa Prize 2017 competition required Milabot to return a response within a few seconds (usually less than one second). To maintain this response speed for several models generating responses word-by-word would have required a significant amount of engineering effort.[26]

[24]See www.washingtonpost.com.

[25]See www.reddit.com.

[26]This was complicated further by the system's software architecture, designed with the assumption that all the models in the ensemble system would run concurrently on a single server instance.

Table 11: Example dialogues and corresponding candidate responses generated by response models. The response selected by the system is marked in bold.

Dialogue	**Dialogue**
SYSTEM: Hi. This is an Alexa Prize socialbot. USER: Hi SYSTEM: Do you have pets? USER: Yes, I have two rabbits.	... USER: Let's talk about movies. SYSTEM: What's your favorite movie? USER: star wars movie
Candidate Responses	**Candidate Responses**
BoWEscapePlan: I don't have an answer for this.	BoWEscapePlan: I don't have an answer for this.
BoWFactGenerator: Did you know that male rabbits are called bucks, females are does?	BoWFactGenerator: Here's an interesting fact. The international telephone dialing code for Antarctica is 672.
GRUQuestionGenerator: How about the American family of it?	GRUQuestionGenerator: how about your New York City?
Initiatorbot: Do you believe in love at first sight?	Initiatorbot: How was your day?
LSTMClassifierMSMarco: Mar 23, 2017 Two Parts: Learning What to Feed Your Rabbit Learning How to...	LSTMClassifierMSMarco: A third Anthology film will be released in 2020.
DualEncoderRedditNews: bye doggie	DualEncoderRedditNews: Now whisper it to me , one more time bby.
DualEncoderRedditPolitics: Who's edger u shboys?	DualEncoderRedditPolitics: hahaha on mobile this ain't even close. I wish I could show you but this looks awful I'm sorry man.
Alicebot: Hurrah! Two is a good number of rabbits.	Alicebot: What did you like about the robots in that movie?
Elizabot: OK, but can you elaborate a bit?	Elizabot: How does that make you feel?
VHREDRedditMovies: Hmm. Nope, never heard of it.	**Evi: Star Wars movie a movie in the Star Wars series.**
VHREDSubtitles: Would you like to see them?	VHREDRedditMovies: Oh please. Please. Pleeeease. Let this happen.
HelpResponse: I would really like to talk about news, politics or movies.	VHREDSubtitles: What?
	HelpResponse: I would really like to talk about news, politics or movies.

4.6 Response Selection Policy

4.6.1 Reinforcement Learning Setup

After generating the candidate response set, the dialogue manager uses a *response selection policy* in order to select the response it returns to the user. The system must select a response which maximizes the satisfaction of the human user for the entire dialogue. In doing so, the system makes a trade-off between immediate and long-term user satisfaction. Consider the example where the system converses with a human user about politics. If the system decides to respond with a political joke, the user may be pleased for one turn. However, the user may later be disappointed with the system's inability to debate complex political issues. If the system instead decides to respond with a short news story, the user may be less pleased for one turn. However, the news story may prompt the user to follow up with factual questions, which the system may be better adept at handling. To make the trade-off between immediate and long-term user satisfaction, we frame the problem of selecting a candidate response as a sequential decision making problem.

We consider the dialogue system as an agent, which takes actions in an environment in order to maximize rewards. At each time step $t = 1, \ldots, T$, the agent observes the dialogue history $s_t \in S$ and must choose one of $K \in \mathbb{N}^+$ actions (responses): $A_t = \{a_t^1, \ldots, a_t^K\}$. Here, S is the infinite discrete set of all possible dialogue histories. After taking an action, the agent receives a reward $r_t \in \mathbb{R}$, which is a score given by the user or a proxy of the user's satisfaction. The agent then transitions to the next state $s_{t+1} \in S$. If the new state is the terminal state, then the conversation is over and there are no more actions to take. Otherwise, the user's next response is included in the next state and the agent is provided with a new set of $K' \in \mathbb{N}^+$ actions: $A_{t+1} = \{a_{t+1}^1, \ldots, a_{t+1}^{K'}\}$. The agent aims to maximize the discounted sum of rewards:

$$R = \sum_{t=1}^{T} \gamma^t r_t, \tag{92}$$

which is the cumulative return and where $\gamma \in (0, 1]$ is a discount factor, as described earlier. In this setting, it should be noted that the set of actions changes depending on the state. This stems from the fact that the candidate responses are generated by response models, which themselves depend on the dialogue history. Furthermore, some of the response models are stochastic. Consequently, two identical states $s, s' \in S$ (i.e. $s == s'$) may have different action sets A, A' (i.e. $A \neq A'$).[27] This stands in contrast to many typical reinforcement learning problems, where the set of actions is fixed given the state. In order to simplify notation, we will fix the number of actions to $K \in \mathbb{N}^+$ moving forward.

[27] In general, the number of candidate responses also changes depending on the state.

4.6.2 Parametrizing the Agent's Policy

We consider two approaches for parametrizing the agent's policy.

Action-value Parametrization: The first approach learns an approximate state-action-value function:

$$Q_\theta(s_t, a_t^k) \in \mathbb{R} \quad \text{where } s_t \in S \text{ and } a_t^k \in A_t, \text{for } k = 1, \ldots, K, \tag{93}$$

which estimates expected return of taking action a_t^k (candidate response k) given dialogue history s_t and given that the agent continues the episode by using the same policy. As before, θ are the parameters. For example, given two candidate responses and their corresponding Q_θ values, we would expect the one with the higher value to result in a higher user satisfaction. Given the approximate state-action-value function, the agent's policy selects the response with the highest estimated return:

$$\pi_\theta(a_t|s_t) = \begin{cases} 1 & \text{if } a_t = \arg\max_{k=1,\ldots,K} \; Q_\theta(s_t, a_t^k) \\ 0 & \text{otherwise.} \end{cases} \tag{94}$$

Stochastic Policy Parametrization: The second approach parameterizes the policy as a discrete probability distribution over actions. The agent's policy selects an action by sampling from the distribution:

$$\pi_\theta(a_t^k|s_t) = \frac{e^{\lambda^{-1} f_\theta(s_t, a_t^k)}}{\sum_{a_t'} e^{\lambda^{-1} f_\theta(s_t, a_t')}} \quad \text{where } s_t \in S \text{ and } a_t^k \in A_t, \tag{95}$$

where $f_\theta(s_t, a_t^k)$ is the *scoring function*, which outputs a real-valued score for each candidate response a_t^k given s_t, with parameters θ. The parameter λ is the temperature parameter, which controls the entropy of the distribution. The higher its value is, the more uniform the distribution will be. The stochastic policy can be transformed into a deterministic, greedy policy by selecting the action with the highest probability:

$$\pi_\theta(a_t|s_t) = \begin{cases} 1 & \text{if } a_t = \arg\max_{k=1,\ldots,K} \; \pi_\theta(a_t^k|s_t) = \arg\max_{k=1,\ldots,K} \; f_\theta(s_t, a_t^k) \\ 0 & \text{otherwise.} \end{cases} \tag{96}$$

4.6.3 A Neural Network Scoring Model

The state-action-value function $Q_\theta(s_t, a_t^k)$ and scoring function $f_\theta(s_t, a_t^k)$ serve a similar purpose. Given a state $s_t \in S$, both functions aim to rank the available actions, such that actions with higher values imply higher expected returns. In particular, if $Q_\theta(s_t, a_t^k) = f_\theta(s_t, a_t^k)$, the state-action-value function policy in eq. (94) is equivalent to the greedy policy in eq. (96). For this reason, we will simply use the same model architecture for both $Q_\theta(s_t, a_t^k)$ and $f_\theta(s_t, a_t^k)$. We will make both

functions take the same set of features as input and process them using the same neural network model architecture to give a real-valued output. Furthermore, we will refer to both functions as *scoring models*.

Specifically, each scoring model will be parametrized as a five-layered neural network.[28] The first layer is the input layer, which receives 1458 real-valued and integer features as input. The second layer has 500 hidden units, which are computed using a rectified linear activation function of the previous layer's units (Nair and Hinton, 2010; Glorot et al., 2011). The third layer has 20 hidden units, which are computed using an affine linear function of the previous layer's hidden units. In other words, this layer compresses the 500 hidden units in the second layer down to 20 hidden units. The fourth layer has 5 outputs units, which are computed using an affine linear function followed by a softmax transformation function of the previous layer's hidden units. This ensures that all the layer's values are positive and sum to one, which enables them to represent a discrete probability distribution with 5 classes. As we will discuss later, this layer's output will be optimized to predict Amazon Mechanical Turk labels. The fifth layer is the final output layer, which consists of a single real-valued scalar computed by an affine linear transformation applied to the concatenation of the units in the third and fourth layers. The model is shown in Figure 16.

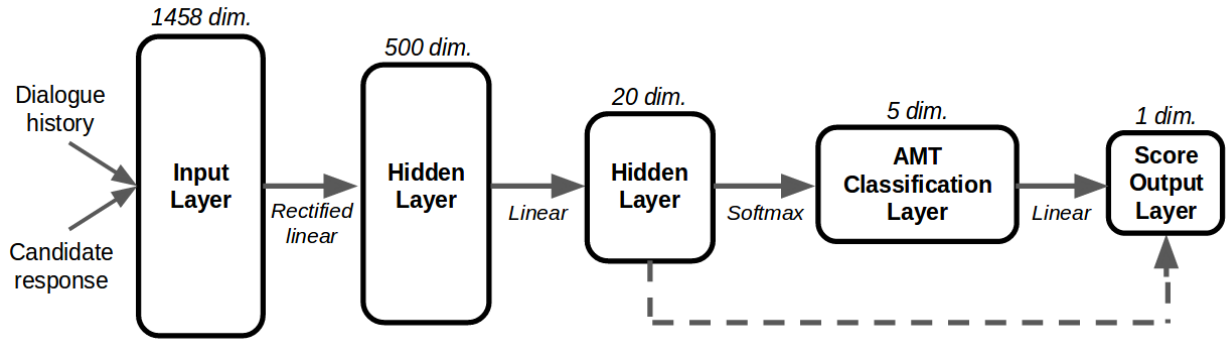

Figure 16: Computational graph for the scoring models, used for the response selection policies based on both state-action-value function and stochastic policy parametrizations. Each model consists of an input layer with 1458 features, a hidden layer with 500 hidden units, a hidden layer with 20 hidden units, a softmax layer with 5 output probabilities, and a scalar-valued output layer. The dashed arrow indicates a skip connection (the last hidden layer output is passed to the last output layer through an affine linear function).

[28] Before settling on the model architecture, we experimented with deeper and shallower models. However, we found that both deeper and shallower models performed either worse or similarly.

4.6.4 Input Features for Scoring Model

The scoring models receive as input 1458 real-valued and integer features, computed from the given dialogue history and candidate response. These features have been derived from word embeddings, dialogue acts, part-of-speech tags, unigram word overlap, bigram word overlap and attributes specific to each response model. The features include:

Word embeddings of model response:	Average of candidate response word embeddings (Mikolov et al., 2013b).[29]
Word embeddings of last user utterance:	Average of previous user utterance word embeddings.
Word embeddings of dialogue history:	Average of word embeddings of last 6 utterances in dialogue history.
Word embedding of user history:	Average of word embeddings of last 3 user utterances in dialogue history.
Word embedding similarity metrics:	Similarity metrics *Embedding Average*, *Embedding Extrema* and *Embedding Greedy* described earlier (Liu et al., 2016). Each metric is computed between 1) last user utterance and candidate response, 2) last 6 utterances and candidate response, 3) last 3 user utterances and candidate response, 4) last 6 utterances and candidate response with stop-words removed, and 5) last 3 user utterances and candidate response with stop-words removed.
Response model class:	One-hot vector with length equal to number of response models, where index i is equal to one if candidate response was generated by response model number i.
Part-of-speech response class:	Part-of-speech tags for candidate response, estimated using a maximum entropy tagger trained on the Penn Treebank corpus and mapped to a one-hot vector (Marcus et al., 1993).

[29] We use the pre-trained Word2Vec embeddings available at: https://code.google.com/archive/p/word2vec/.

Dialogue act response model class:	Outer-product between a one-hot vector representing dialogue act and a one-hot vector for indicating model class (Stolcke et al., 2000).[30]
Word overlap:	Binary feature equal to one only when any non-stop-words overlap between candidate response and last user utterance
Bigram overlap short-term:	Binary feature equal to one only when a bigram (two consecutive tokens) exists both in candidate response and in last user utterance.
Bigram overlap long-term:	Binary feature equal to one only when a bigram exists both in candidate response and in last utterances in dialogue history.
Named-entity overlap short-term:	Binary feature equal to one only when a named-entity exists both in candidate response and in last user utterance.[31]
Named-entity overlap long-term:	Binary feature equal to one only when a named-entity exists both in candidate response and in one of several last utterances in dialogue history.
Generic response:	Binary feature equal to one if candidate response consists of only stop-words or words shorter than 3 characters.
Wh-word response feature:	Binary feature equal to one if candidate response contains a wh-word (e.g. *what*, *where*, and so on).
Wh-word context:	Binary feature equal to one if last user utterance contains a wh-word.
Intensifier word response:	Binary feature equal to one if candidate response contains an intensifier word (e.g. *amazingly*, *crazy*, and so on).
Intensifier word context:	Binary feature equal to one if last user utterance contains an intensifier word.

[30] We consider 10 types of dialogue acts.

[31] Here, a named-entity is defined as an upper-cased word, which is not a stop-word.

Unigram response:	A group of binary features, which are triggered when candidate response contains a specific word, such as the words *I*, *you* and *thanks*.
Negation response:	Binary feature equal to one if candidate response contains a negation word, such as *not* or *n't*, and otherwise zero.
Non-stop-words response:	Binary feature equal to one if candidate response contains a non-stop-word.

Features based on the confidences of the speech recognition system are not included. This is in contrast to other work, where speech recognition confidences are often taken into account by the policy of the spoken dialogue system. Although such features may improve the performance of the spoken dialogue system, this decision is made in order to reduce the confounding factor of speech recognition errors in the experiments with real-world users. For example, if the confidences of the speech recognition system were included, one policy might be better adept at handling speech recognition errors than other policies simply by being optimized on a dataset incorporating such errors. This could in turn make that policy perform better w.r.t. overall user satisfaction. However, under a perfect speech recognition system, that same policy would not have any such advantage. Nevertheless, even if speech recognition confidence features are excluded, speech recognition errors still constitute a major confounding factor in the experiments presented later.

For reasons of computational speed and simplicity, no features based on recurrent neural networks embedding the dialogue history or candidate response are used.[32]

In the next sections, we discuss the algorithms for learning the parameters of the neural network scoring model.

[32]Due to the speed of the response models, the scoring models in Milabot had to complete their execution within 150ms.

4.7 Learning the Response Selection Policy with Supervised Learning on Crowdsourced Labels

We propose six algorithms for learning the response selection policy. Each algorithm uses a different approach to optimize the parameters of the neural network scoring model. This section describes the first approach.

The first approach estimates an approximate state-action-value function using supervised learning on crowdsourced labels. This approach also serves as a method for initializing the parameters of the neural network scoring model of the other five approaches, discussed in later sections.

Many researchers have turned to crowdsourcing platforms for collecting training data for their dialogue systems (Gašić et al., 2011; Gasic et al., 2013; Su et al., 2016). Crowdsourcing is a scalable and cost-efficient method for data collection, which can easily be applied to annotate both text-based and spoken conversations. With respect to Milabot, crowdsourcing offers the ability to annotate two orders of magnitude more examples compared to the user scores given through the Amazon Alexa Prize 2017 competition. Furthermore, crowdsourcing also makes it possible to annotate single responses in a conversation. In contrast, the users in the Amazon Alexa Prize 2017 competition only provide a single score at the end of the conversation. Such a large amount of data examples, annotated at the response-level, makes it feasible to train a neural network scoring model with hundreds of thousands of parameters.

4.7.1 Crowdsourcing Data Collection

We use Amazon Mechanical Turk (AMT) to collect training data for the scoring model by following a setup similar to Liu et al. (2016). We show human evaluators a dialogue followed by 4 candidate responses, and ask them to score how appropriate each candidate response is on a 1-5 Likert-type scale, where 1 indicates an inappropriate or nonsensical response and 5 indicates a highly appropriate and excellent response. In this setup, human evaluators rate only the overall appropriateness of the candidate responses.

The dialogues shown to the human evaluators are extracted from interactions between real-world Alexa users and several preliminary versions of the Milabot system. Dialogues with priority responses are excluded. The candidate responses are created for each dialogue by invoking the response models.

In total, we collect 199,678 labels. This dataset is split into training (train), development (dev) and testing (test) datasets consisting of respectively 137,549, 23,298 and 38,831 labels each. Further details are given in Appendix V.

4.7.2 Policy Training

The scoring model is trained by considering the fourth layer as a discrete probability distribution with 5 classes representing the corresponding AMT label classes. Given a dialogue history and candidate response as input, the neural network parameters are optimized by maximizing the log-likelihood of the unit in the fourth layer with index corresponding to the assigned AMT label. Let $D = \{(x_i, y_i)\}_{i=1}^{I}$ be the training dataset of the dialogue histories, candidate responses and corresponding AMT labels, where $x_i \in \mathbb{R}^{1458}$ is a vector of input features computed based on the dialogue history and candidate response and $y_i \in \{1, 2, 3, 4, 5\}$ is the corresponding AMT label class for $i = 1, \ldots, I$. The neural network parameters θ are optimized to maximize the quantity:

$$\sum_{(x,y)\in D} \log P_\theta(y|x), \tag{97}$$

where $P_\theta(y|x)$ is the model's predicted probability of class $y \in \{1, 2, 3, 4, 5\}$ given input features $x \in \mathbb{R}^{1458}$, computed in the fourth layer of the neural network.

The scoring model neural network is implemented using the Theano library (Al-Rfou et al., 2016). The parameters are updated using the first-order optimization method Adam (Kingma and Ba, 2015). Different hyperparameter combinations are tried out and the best one is selected based on the log-likelihood of the examples in the development dataset. For the first hidden layer, the following number of hidden units are evaluated: $\{500, 200, 50\}$. For the second hidden layer, the following number of hidden units are evaluated: $\{50, 20, 5\}$. L2 regularization is applied to all model parameters, except for bias parameters (Goodfellow et al., 2016). The following L2 regularization coefficients are evaluated: $\{10.0, 1.0, 10^{-1}, \ldots, 10^{-9}\}$.

The parameters of the last layer (output layer) in the neural network cannot be optimized based on this algorithm, because the gradients w.r.t. these parameters are zero. Therefore, the parameters from the fourth layer to the last layer are fixed to the vector $[1.0, 2.0, 3.0, 4.0, 5.0]$ while the parameters from the third layer to the last layer are fixed to zero. In other words, the last layer assigns a score of 1.0 when the fourth layer assigns 100% probability to the unit corresponding to the worst AMT label (i.e. an inappropriate or nonsensical response) and a score of 5.0 when the fourth layer assigns 100% probability to the unit corresponding to the best AMT label (i.e. a highly appropriate and excellent response). Since this scoring model is trained on AMT crowdsourced data, we name it *Supervised AMT*.

4.7.3 Preliminary Evaluation

A preliminary evaluation of *Supervised AMT* is carried out in order to validate the proposed approach and obtain an estimate of the learned policy's performance. The performance of *Supervised AMT* w.r.t. each AMT label class is given in Figure 17. The figure also shows the performance of

three simple, baseline policies: 1) *Random*, which selects a response at random, 2) *Alicebot*, which selects an *Alicebot* response if available and otherwise selects a response at random, and 3) *Evibot + Alicebot*, which selects an *Evibot* response if available and otherwise selects an *Alicebot* response. For each policy, the figure shows the frequency of selected responses belonging to each AMT label class.

First, we observe that *Supervised AMT* achieves a ~30% point reduction for the *"very poor"* AMT label class compared to *Random*. For the same AMT label class, *Supervised AMT* obtains a ~10% point reduction compared to *Alicebot* and *Evibot + Alicebot*. Next, we observe that *Supervised AMT* performs substantially better than the three baselines w.r.t. the AMT label classes *"good"* and *"excellent"*. In particular, *Supervised AMT* has ~8% of its responses for the class *"excellent"*, which is more than double compared to all three baseline policies. This clearly shows that *Supervised AMT* has learned to more often select *"good"* and *"excellent"* responses, while avoiding *"very poor"* and *"poor"* responses. The overall results show that *Supervised AMT* improves substantially over all baseline policies. Despite these improvements, over 45% of the *Supervised AMT* responses still belong to the classes *"very poor"* and *"poor"*.

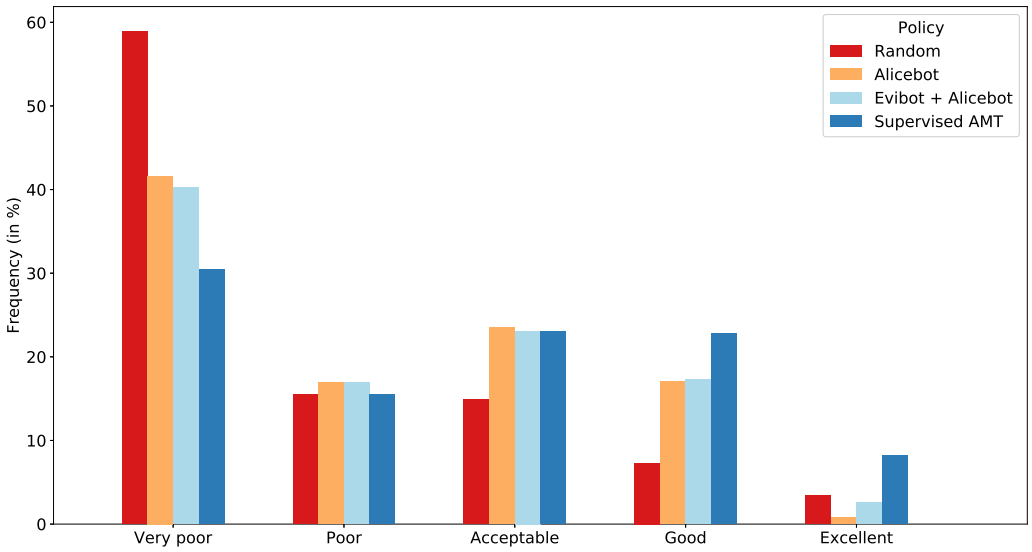

Figure 17: Amazon Mechanical Turk (AMT) class frequencies on the AMT test dataset w.r.t. candidate responses selected by different policies.

4.8 Learning the Response Selection Policy with Supervised Learning on Real-World User Scores

In the *Supervised AMT* model, the parameters from the fourth layer to the fifth layer are fixed to $[1.0, 2.0, 3.0, 4.0, 5.0]$. In other words, the last layer assigns a score of 1.0 when the fourth layer assigns 100% probability to the unit corresponding to the worst AMT label (i.e. an inappropriate or nonsensical response) and a score of 5.0 when the fourth layer assigns 100% probability to the unit corresponding to the best AMT label (i.e. a highly appropriate and excellent response). However, these labels are obtained by non-expert human annotators evaluating a single text-based candidate response in a dialogue and likely do not take into account the past or future of the conversation, the goals, personal interests and affective state of the human interlocutor, or the effect of the candidate response being used in a spoken conversation between a human and a spoken dialogue system with a monotone, synthetic voice. Therefore, these labels are probably not representative of the overall utility (i.e. user satisfaction) induced by their corresponding candidate responses. We propose to remedy this problem by learning to predict the Alexa user scores based on previously recorded dialogues.

4.8.1 Learned Reward Function

Let $D = \{(s_t^d, a_t^d, R^d)\}_{d,t}$ be a set of $|D|$ examples, where t denotes the time step (dialogue turn) and d denotes the dialogue index. Let s_t^d be a dialogue history and let a_t^d be the corresponding response, given by the system at time t for dialogue d. Let $R^d \in [1, 5]$ denote the observed real-valued Alexa user score for dialogue d.

We aim to learn a linear regression model g_ϕ, with model parameters ϕ. The aim of this model is to predict the Alexa user score from a given dialogue history and system response:

$$g_\phi(s_t, a_t) \in [1, 5]. \tag{98}$$

We refer to this model as a *reward model*, since it directly models the Alexa user score. From the reinforcement learning perspective, we shall interpret the Alexa user score as the return of the agent. In other words, the Alexa user score is equal to the discounted sum of rewards given in eq. (92) for a given episode.

Alexa users are prompted to give a score in the range $1 - 5$ at the end of a dialogue with Milabot, but they may opt out by stopping the application. Since not all Alexa users give scores, we exclude recorded dialogues without scores. However, ignoring dialogues without scores incurs a significant bias in the reward model. For example, it seems likely that users who do not provide a score either find the system to be poor or to lack particular functions/features they expected (e.g. *non-conversational activities*, such as playing games or taking quizzes, which were implemented

in other socialbots in the Amazon Alexa Prize 2017 competition). Furthermore, some of the users give decimal scores (e.g. 3.5). Therefore, we treat R^d as a real-valued number in the range $1-5$.

The linear regression model is optimized by minimizing the squared error between the model's prediction and the observed return:

$$\sum_{(s_t^d, a_t^d, R^d) \in D} (g_\phi(s_t^d, a_t^d) - R^d)^2. \tag{99}$$

Since training data is scarce, the reward model receives as input 23 features computed based on the dialogue history and system response:

AMT label class: A vector indicating the probability of the AMT label classes for the response, computed using *Supervised AMT*, and whether the response is a *priority response* or not. If the response is a *priority response*, all the vector elements are zero except the last element corresponding to the priority class (i.e. $[0.0, 0.0, 0.0, 0.0, 0.0, 1.0]$).

Generic response: A binary feature, which equals one if response contains only stop-words.

Response length: Number of words in response, and square root of the number of words in response.

Dialogue act: One-hot vector indicating whether last user utterance is a dialogue act *request*, a dialogue act *question*, a dialogue act *statement* or contains profanity (Stolcke et al., 2000).

Sentiment class: One-hot vector indicating if the sentiment of last user utterance is negative, neutral or positive.

Generic user utterance: Binary feature, which equals one if last user utterance only contains stop-words, and otherwise zero.

User utterance length: Number of words in last user utterance, and square root of the number of words in the utterance.

Confusion indicator: A binary feature, which equals one if last user utterance is less than three words and contains at least one word indicating the user is confused (e.g. *"what"*, *"silly"*, *"stupid"*).

Dialogue length: Number of dialogue turns, as well as square root and logarithm of number of dialogue turns.

We train the reward model using a dataset of recorded dialogues between real-world Alexa users and several preliminary versions of the Milabot system. The dataset consists of 4340 dialogues and is split into a training set of 3255 examples and a test set of 1085 examples.

Due to the scarcity of the training data, we increase data efficiency by learning an ensemble model through a variant of the bagging technique (Breiman, 1996). We construct 5 new training datasets, which represent shuffled versions of the original training dataset of 3255 examples. Each new shuffled dataset is split into a sub-training dataset and sub-hold-out dataset. The sub-hold-out datasets are constrained such that the examples in one sub-hold-out dataset do not overlap with other sub-hold-out datasets. We train a linear regression reward model on each sub-training dataset and select its hyperparameters based on the average squared error on the sub-hold-out dataset. This procedure increases data efficiency by reusing the sub-hold-out datasets for training, rather than only for hyperparameter optimization. The final reward model is an ensemble model, where the output is the average of the underlying 5 linear regression models.

We implement the reward model using the Theano library (Al-Rfou et al., 2016). The model parameters for each linear regression model is optimized using stochastic gradient descent with Adam. L2 regularization with the following coefficients are used:
$\{10.0, 1.0, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.0\}$. The L2 regularization coefficient with the smallest squared error on each sub-hold-out dataset is selected for each model.

4.8.2 Preliminary Evaluation of Learned Reward Function

We evaluate the performance of the learned reward model on the test set of 1085 examples. The reward model obtains a mean squared error of 0.96 and a Spearman's rank correlation coefficient of 0.19 w.r.t. Alexa user scores on the test set. For comparison, a model predicting with the average Alexa user score obtains a mean squared error of 0.99 and a Spearman's rank correlation coefficient of zero.[33] The reward model is better than predicting the average, but overall its correlation is low.

One reason for the low correlation might be because of the small amount of training data, which makes it difficult for the model to learn the relationships between the input features and the Alexa user scores. Another reason might be because the Alexa user scores are influenced by many different factors, which in turn leads to them having high variance. For example, the user score might in many cases be determined largely from a single turn in the dialogue, such as when the dialogue system has a fatal misunderstanding w.r.t. the conversation topic. As another example, the score of the user may be affected by the errors of the speech recognition system. More speech recognition errors will inevitably lead to frustrated users and lower user satisfaction.[34] As a last

[33] Since a model predicting with the average Alexa user score outputs the same value for every example, its correlation with any dependent variable will always be zero.

[34] For example, in a preliminary analysis, we observed that the Spearman's rank correlation coefficient between the

example, many extrinsic factors are likely to influence the user scores. The user scores may be affected by the user's profile (e.g. whether the user is an adult or a child), the interaction model (e.g. whether there is a single or a group of users conversing with the system), the user's expectations towards the system (e.g. whether the user expects that the system is capable of playing games) and the affective state of the user.

4.8.3 Policy Training

Given the learned reward model, we can train a corresponding neural network scoring model. However, the learned reward model obtained low correlation with Alexa user scores. Therefore, we will first initialize the neural network scoring model with the parameters of the *Supervised AMT* scoring model, and then fine-tune it with the learned reward model outputs by minimizing the squared errors between its prediction and the prediction of the learned reward model.

As before, the scoring model parameters are optimized with stochastic gradient descent using Adam. For training with the learned reward model, we use a separate dataset of several thousand recorded dialogue examples, where about 80% are used for training and about 20% are used as hold-out set. No L2 regularization is used. We early stop on the squared error of the hold-out dataset w.r.t. Alexa user scores predicted by the reward model. Since this scoring model is trained with AMT labels and a learned reward function, we name it *Supervised AMT Learned Reward*.

speech recognition system confidences and the Alexa user scores varied between $0.05 - 0.09$. Compared to other factors, this implies that speech recognition performance plays an important role in determining user satisfaction.

4.9 Learning the Response Selection Policy with Off-Policy REINFORCE

In the previous section, we discussed how the *Supervised AMT* model is trained using AMT labels unrepresentative of the overall utility or user satisfaction. One remedy to this problem is the *Supervised AMT Learned Reward* model, which learns an approximate state-action-value function from real-world user scores. In this section, we propose another remedy based on a class of reinforcement learning algorithms collectively known as *policy gradient reinforcement learning*.

4.9.1 Off-Policy REINFORCE

Our first policy gradient approach is based on a variant of the classical *REINFORCE* algorithm in the off-policy setting (Williams, 1992; Precup, 2000; Precup et al., 2001), which we call *Off-policy REINFORCE* or simply *REINFORCE*. As in eq. (95), let the policy's distribution over actions be parametrized as softmax function applied to a neural network scoring function f_θ with parameters θ. As before, let $D = \{(s_t^d, a_t^d, R^d, T^d)\}_{d,t}$ be a dataset of examples, where s_t^d is the dialogue history at time t, a_t^d is the agent's action at time t, R^d is the return and T^d is the total number of turns in dialogue d. Let $|D|$ be the number of dialogues. Then, let θ_d be the parameters of the stochastic policy π_{θ_d}, with parameters θ_d, used by the system during dialogue d. The vanilla *Off-policy REINFORCE* algorithm updates the policy parameters θ using:

$$\Delta\theta \;= \alpha \; c_t^d \; \nabla_\theta \log \pi_\theta(a_t^d|s_t^d) \; R^d, \quad \text{where } d \sim \text{Uniform}(1, D) \text{ and } t \sim \text{Uniform}(1, T^d), \tag{100}$$

where $\alpha > 0$ is the learning rate, and where c_t^d is the importance weight ratio:

$$c_t^d = \frac{\prod_{t'=1}^{t} \pi_\theta(a_{t'}^d|s_{t'}^d)}{\prod_{t'=1}^{t} \pi_{\theta_d}(a_{t'}^d|s_{t'}^d)}. \tag{101}$$

The importance weight ratio accounts for the discrepancy between the learned policy π_θ and the policy π_{θ_d}, under which the data was collected. The ratio increases the weights of examples with high probability under the learned policy and decreases the weights of examples with low probability under the learned policy. The algorithm can be understood from the point of view of a trial-and-error mechanism. When an example has a high return (i.e. a high Alexa user score), the term $\nabla_\theta \log \pi_\theta(a_t^d|s_t^d) \; R^d$ will be a vector pointing in a direction increasing the probability of taking action a_t^d. In this case, the policy π_θ will learn to take action a_t^d more frequently. However, when an example has low return (i.e. low Alexa user score), the term $\nabla_\theta \log \pi_\theta(a_t^d|s_t^d) \; R^d$ will be a vector close to zero or a vector pointing in the opposite direction. In this case, the policy π_θ will often learn to take action a_t^d less frequently.

Under certain conditions, the parameter update given by eq. (100) is an unbiased estimate of the true gradient of the return w.r.t. the policy parameters. This is in contrast with the policy *Supervised*

AMT Learned Reward, where the policy is trained with a learned reward model. In that case, there is no guarantee that the parameter update based on the learned reward model improves the policy's performance. As the learned policy diverges further away from the policy, which was used for training the learned reward model, the learned reward model is likely to become less accurate.

However, in practice, the importance weight ratio tends to suffer from a high amount of variance (Precup et al., 2001). Therefore, it is common to truncate the products in the numerator and denominator to only include the current time step t:

$$c^d_{t,\text{trunc.}} = \frac{\pi_\theta(a^d_t|s^d_t)}{\pi_{\theta_d}(a^d_t|s^d_t)}. \tag{102}$$

This technique reduces variance and acts as a regularizer. However, in general, it also introduces bias into the parameter update.

More generally, the parameter update given by eq. (100) suffers from high variance due to the so-called *credit assignment problem* (Sutton and Barto, 1998). Part of the reason is because the algorithm uses the return, observed at the very end of an entire episode, to update the policy's action probabilities for all the intermediate actions in an episode. With a small number of examples, the high variance of the gradient estimator may often cause the agent to over-estimate the utility of poor actions or, vice versa, to under-estimate the utility of good actions. This problem is exacerbated in the Amazon Alexa Prize 2017 competition, when the return is taken to be the Alexa user score.

One way to tackle this problem is through *reward shaping*, where the reward at each time step is estimated using an auxiliary function (Ng et al., 1999). The auxiliary function makes it possible to utilize prior knowledge and structural properties of the environment, which in turn can increase the sample efficiency of the learning algorithm. Here, we propose a simple variant of reward shaping which takes into account the sentiment of the user. When the user's utterance appears to have a negative sentiment (e.g. an angry comment), we will assume that the immediately preceding action (system utterance) was highly inappropriate and assign it a reward of zero. For each dialogue d and time step t, we assign reward r^d_t:

$$r^d_t = \begin{cases} 0 & \text{if user utterance at time } t+1 \text{ is classified as having negative sentiment,} \\ \dfrac{R^d}{T^d} & \text{otherwise.} \end{cases} \tag{103}$$

Incorporating the proposed reward shaping and truncated importance weights, the final parameter update equation is:

$$\Delta\theta = \alpha\, c^d_{t,\text{trunc.}} \nabla_\theta \log \pi_\theta(a^d_t|s^d_t)\, r^d_t, \quad \text{where } d \sim \text{Uniform}(1, D), t \sim \text{Uniform}(1, T^d), \tag{104}$$

where $\alpha > 0$ is the learning rate.

4.9.2 Off-Policy REINFORCE with Learned Reward Function

Similar to the *Supervised AMT Learned Reward* policy, we may use the learned reward model from the previous section to train the policy using the *REINFORCE* algorithm. In this case, we use the learned reward model g_ϕ, defined in eq. (98), to compute a new estimate for the reward at each time step in each dialogue:

$$r_t^d = \begin{cases} 0 & \text{if user utterance at time } t+1 \text{ is classified as having negative sentiment,} \\ g_\phi(s_t, a_t) & \text{otherwise.} \end{cases} \tag{105}$$

This new reward is substituted into eq. (104) for training. We name this policy *Off-Policy REINFORCE Learned Reward* or, more simply, *REINFORCE Learned Reward*.

4.9.3 Policy Training

During the training procedure, we evaluate the learned policies through an estimate of their expected returns (Precup, 2000):

$$\sum_{d,t} c_{t,\text{trunc.}}^d \, r_t^d. \tag{106}$$

The neural network scoring functions are initialized from the *Supervised AMT* model parameters, and then optimized using eq. (104) with stochastic gradient descent using Adam. As before, the training set consists of a few thousand dialogues recorded between Alexa users and several preliminary versions of the Milabot system. About 60% of these examples are used for training, and about 20% are used for development and testing respectively. To reduce the risk of overfitting, only the parameters of the second last layer are trained. The hyperparameters include the temperature parameter λ and the learning rate α. The hyperparameters are found via a random grid search and selected based on each policy's expected return on the development set.

4.10 Learning the Response Selection Policy with Model-Based Reinforcement Learning

The previous approaches proposed each have their own advantages and disadvantages, which can be quantified through the *bias-variance trade-off*. Specifically, we may evaluate each approach w.r.t. the amount of (statistical) bias and variance incurred either by its training procedure at each step (e.g. the bias and variance exhibited by the training procedure's parameter change compared to the true, unbiased gradient) or by a learned policy (e.g. the bias and variance exhibited by a set of learned parameters compared to the set of parameters that maximize the cumulative return). At one end of the spectrum, the *Supervised AMT* policy has low variance, because it was trained with hundreds of thousands of human annotations given for candidate responses. However, *Supervised AMT* suffers from a substantial bias, because the human annotations are likely unrepresentative of the real user satisfaction. At the other end of the spectrum, *REINFORCE* suffers from high variance, because it was trained with only a few thousand dialogues and corresponding user scores. This issue is exacerbated by the fact that the user scores are likely affected by multiple external factors, such as user profiles and expectations, and by the fact that they are only given at the end of an entire conversation. Nevertheless, *REINFORCE* incurs less bias by directly optimizing the relevant objective metric: the Alexa user score. By utilizing a learned reward function, *Supervised Learned Reward* and *REINFORCE Learned Reward* aim to find a better bias-variance trade-off. However, the learned reward function becomes increasingly inaccurate as the learned policy diverges away from the policy used to train the learned reward function. Furthermore, since the learned reward function has its own variance component, both *Supervised Learned Reward* and *REINFORCE Learned Reward* might also suffer from a significant amount of variance.

In this section we propose an algorithm for making a different bias-variance trade-off using *model-based reinforcement learning*. This algorithm learns a policy from simulations in an approximate Markov decision process (MDP). In particular, the approximate MDP incorporates a few high-level assumptions about the structural properties of the environment and prior knowledge, which aim to significantly reduce the variance of the learned policy, while only incurring a small additional amount of bias.

4.10.1 Bottleneck Simulator

The algorithm we propose learns both a policy and an explicit model of the environment. The explicit model of the environment is an approximate MDP. The approximate MDP utilizes a *bottleneck state*, which is why the algorithm is named the *Bottleneck Simulator*.

The approximate MDP follows the probabilistic generative model shown in Figure 18. At time t, the agent is in state $z_t \in Z$. This variable is a discrete random variable representing the

bottleneck state, which represents the abstract, high-level semantic structure related to the dialogue history. The set Z is defined by a Cartesian product:

$$Z = Z_{\text{Dialogue act}} \times Z_{\text{User sentiment}} \times Z_{\text{Generic user utterance}}, \tag{107}$$

where $Z_{\text{Dialogue act}}$, $Z_{\text{User sentiment}}$ and $Z_{\text{Generic user utterance}}$ are discrete sets. The first discrete set is a set of 10 dialogue acts: $Z_{\text{Dialogue act}} = \{\text{Accept}, \text{Reject}, \text{Request}, \text{Politics}, \text{Generic Question}, \text{Personal Question}, \text{Statement}, \text{Greeting}, \text{Goodbye}, \text{Other}\}$. The dialogue acts represent the high-level intention of the user's utterance (Stolcke et al., 2000). The second discrete set is a set of three sentiment labels: $Z_{\text{User sentiment}} = \{\text{Negative}, \text{Neutral}, \text{Positive}\}$. The third set is a binary set: $Z_{\text{Generic user utterance}} = \{\text{True}, \text{False}\}$. This variable is *True* only if the user's last utterance exclusively contains stop-words. We construct a hand-crafted, deterministic classifier, which given a dialogue history $s_t \in S$ outputs the corresponding classes in $Z_{\text{Dialogue act}}$, $Z_{\text{User sentiment}}$ and $Z_{\text{Generic user utterance}}$. We denote the classifier's function as $f_{s \to z}$. Although we only consider dialogue acts, sentiment and generic utterances, it is trivial to expand the set with other types of discrete or real-valued variables representing other types of semantic structure. The dialogue acts, sentiment and generic utterances are expected to be indicative of both the future trajectory of the dialogue and of the user satisfaction. By explicitly modelling these high-level semantic components, the *Bottleneck Simulator* captures structural properties of the environment in order to increase the overall sample efficiency and to reduce the variance of the learned policy.

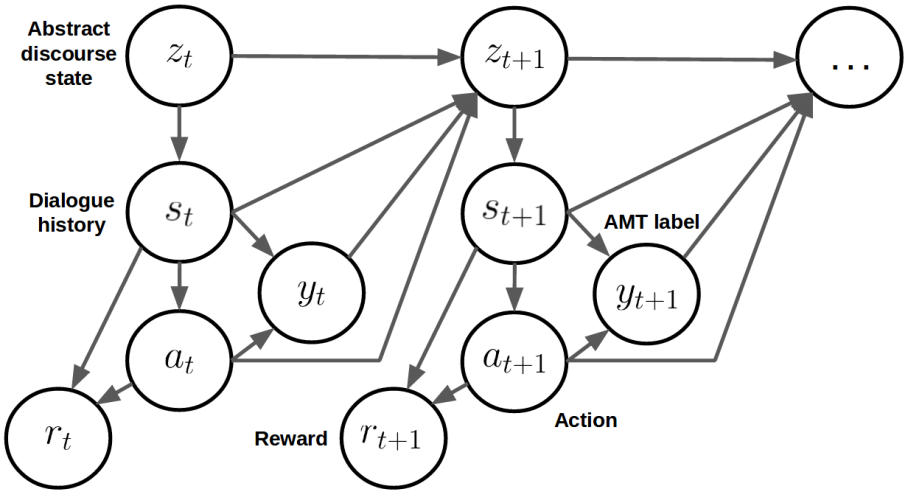

Figure 18: Probabilistic directed graphical model for the *Bottleneck Simulator*. For each time step t, z_t is a discrete random variable which represents the abstract state of the dialogue, s_t represents the dialogue history (i.e. the state of the agent), a_t represents the action taken by the system (i.e. the selected response), y_t represents the sampled AMT label and r_t represents the sampled reward.

Given a sample $z_t \in Z$, the *Bottleneck Simulator* samples a dialogue history $s_t \in S$ from a

finite set of recorded dialogue histories $\bar{S}$, where $\bar{S} \subseteq S$ and S is the set of all possible dialogue histories. Specifically, s_t is sampled at random uniformly from the set of dialogue histories where the last utterance is mapped to z_t:

$$s_t \sim P(s_t \mid \bar{S}, z_t) = \text{Uniform}(\{s \mid s \in \bar{S} \text{ and } f_{s \to z}(s) = z_t\}). \tag{108}$$

In other words, s_t is constrained to be a dialogue history where the dialogue act, user sentiment and generic property matches the corresponding discrete variable z_t.

In our experiments, the set $\bar{S}$ consists of recorded dialogues between Alexa users and preliminary versions of the Milabot system. The set $\bar{S}$ increases over time when the system is deployed in practice, which makes it possible to continuously improve the response selection policy as new data becomes available. Further, it should be noted that the set Z is defined as a small discrete set, such that every possible state $z \in Z$ occurs at least once in the set of recorded dialogues $\bar{S}$. This makes it possible to estimate transition probabilities between consecutive states.

Given a sample s_t, the agent chooses an action a_t according to its policy $\pi_\theta(a_t|s_t)$, with parameters θ. Then, a reward r_t is sampled such that $r_t \sim R(r_t|s_t, a_t)$, where R is a probability distribution. The probability distribution R is estimated using a supervised learning model trained on AMT labels, as given in eq. (97). In principle, the probability distribution can be sampled based on the estimated probabilities for each AMT label class in the range $1-5$. However, to reduce the amount of stochasticity, the probability distribution is set to the (discrete) Dirac-delta distribution:

$$R(r_t|s_t, a_t) = \begin{cases} 1 & \text{if } \; r_t = \sum_{y=1}^{5} P_\zeta(y|s_t, a_t)(y-3) \\ 0 & \text{otherwise,} \end{cases} \tag{109}$$

where $P_\zeta(y|s_t, a_t)$ is the probability of the AMT label class $y \in \{1, 2, 3, 4, 5\}$ conditioned on dialogue history s_t and action (candidate response) a_t, computed by the supervised learning model trained on the AMT labels with parameters ζ. The term $(y-3)$ ensures that, if 100% of the probability mass of $P_\zeta(y|s_t, a_t)$ is assigned to the middle AMT label $y = 3$ (i.e. the label *"acceptable"*), then the assigned reward is exactly zero: $r_t = 0$.

Following this, a discrete random variable $y_t \in \{1, 2, 3, 4, 5\}$ is sampled from $P_\zeta(y_t|s_t, a_t)$:

$$y_t \sim P_\zeta(y_t|s_t, a_t). \tag{110}$$

This variable represents how a user might judge the appropriateness of the given response a_t. The variable helps predict the future state z_{t+1}, because the response's appropriateness may have a significant impact on the user's next utterance (for example, poor or inappropriate responses often cause users to respond with confusion or to change topic).

Finally, a new state z_{t+1} is sampled according by:

$$z_{t+1} \sim P_\psi(z_{t+1} \mid z_t, s_t, a_t, y_t), \tag{111}$$

where $P_\psi(z_{t+1} \mid z_t, s_t, a_t, y_t)$ is the transition distribution of observing z_{t+1} given the other variables, with parameters ψ. This distribution is defined by three independent two-layer neural network models, which each take as input the same features as the neural network scoring models and the following additional features:

AMT label class:	One-hot vector representing the sampled AMT label class y_t.
Dialogue act:	One-hot vector representing the dialogue act of last user utterance.
Sentiment:	One-hot vector representing the sentiment of last user utterance.
Generic user utterance:	Binary feature equal to one if last user utterance only contains stop-words.
Wh-words:	Binary feature equal to one if last user utterance contained a wh-word (e.g. *what*, *who*).

The first neural network predicts the next dialogue acts by outputting a probability for each class in $Z_{\text{Dialogue act}}$. The second neural network predicts the next sentiment type by outputting a probability for each class in $Z_{\text{User sentiment}}$. The third neural network predicts whether the next user utterance is generic or not, by outputting a probability for each class in $Z_{\text{Generic user utterance}}$. The Cartesian product of their probabilities defines the probability $P_\psi(z_{t+1} \mid z_t, s_t, a_t, y_t)$ for every $z \in Z$.

The dataset for training the neural network models consists of 499,757 transitions, of which 70% are used for training and 30% for evaluation. As before, the model parameters are optimized w.r.t. log-likelihood with gradient decent using Adam. The models are trained with early stopping on the hold-out set. No other regularization is used. The three neural network models attain a joint perplexity of 19.51. A baseline model, which always assigns the average class frequency as the output probability, obtains a worse perplexity of 23.87. This indicates that roughly $3 - 4$ possible states $z_{t+1} \in Z$ can be eliminated by conditioning on the previous variables z_t, s_t, a_t and y_t on average. This suggests that the previous states z_t and s_t, together with the agent's action a_t, are bound to have an effect on the future state z_{t+1}, and that an agent trained in the *Bottleneck Simulator* may learn to take this effect into account. This is in contrast to both policies learned using supervised learning, which do not consider future dialogue states, and policies learned using REINFORCE, which only implicitly take into account future states of the dialogue.

4.10.2 Policy Training

Given the approximate MDP of the *Bottleneck Simulator*, we can learn a policy directly from rollout simulations in the approximate MDP. We use the Q-learning algorithm, described in section 2.2.3, with a so-called *experience replay buffer* in order to learn a neural network scoring model

policy (Mnih et al., 2013; Lin, 1993).[35] We use an ϵ-greedy policy with $\epsilon = 0.1$. We experiment with the following discount factors: $\gamma \in \{0.1, 0.2, 0.5\}$. As before, the neural network scoring model parameters are trained using Adam. We only train the parameters related to the final output layer and the skip-connection (shown in dotted lines in Figure 16) using Q-learning, in order to reduce the risk of overfitting.

As before, the neural network scoring model is initialized with the parameters of the *Supervised AMT* policy. Then, training is carried out in two alternating phases. First, the policy is trained for 100 episodes. Second, the policy is evaluated for 100 episodes w.r.t. average return. Afterwards, the policy is again trained for 100 episodes. For the evaluation, each dialogue history is sampled from a separate set of dialogue histories, which is disjoint from the set of dialogue histories used at training time. This ensures that the policy is not overfitting. A policy is trained between 400 and 600 episodes for each hyperparameter combination. The best hyperparameters are selected w.r.t. average return. To keep notation brief, we call this policy *Bottleneck Simulator*.

[35] For experience replay, we use a memory buffer of size 1000.

4.11 Learning the Response Selection Policy with Other Reinforcement Learning Algorithms

As discussed earlier, the work in this second part of the thesis is based on the pre-print article: "The Bottleneck Simulator: A Model-based Deep Reinforcement Learning Approach" by Iulian Vlad Serban, Chinnadhurai Sankar, Michael Pieper, Joelle Pineau and Yoshua Bengio, arXiv:1807.04723, 2018. The article was submitted to the International Conference on Machine Learning 2018 (ICML 2018). The reviewers rejected the article for the major reason that it was missing two popular reinforcement learning algorithms: vanilla Q-learning and *state abstraction*. This section describes the implementation of these two algorithms. The next section includes experiment results with these two algorithms.

4.11.1 Q-Learning Policy

The Q-learning algorithm, described in section 2.2.3, can be applied directly to the recorded dialogues. As before, let $D = \{(s_t^d, a_t^d, R^d, T^d)\}_{d,t}$ be a dataset of examples, where s_t^d is the dialogue history at time t, a_t^d is the agent's action at time t, R^d is the return and T^d is the total number of turns in dialogue d. Let $|D|$ be the number of dialogues. Then, let ψ be the parameters of the approximate state-action-value function Q_ψ. The parameters are updated by minimizing the squared error:

$$\sum_{(s_t^d, a_t^d, R^d, T^d) \in D} ||Q_\psi(s_t, a_t) - \hat{Q}_\psi(s_t, a_t)||^2,$$

$$\text{where } \hat{Q}_\psi(s_t, a_t) = r_t + \gamma \max_{a'} Q_\psi(s_{t+1}, a'), \tag{112}$$

$$\text{and } r_t = 1_{(t=T^d)} R^d \tag{113}$$

where $\hat{Q}_\psi(s_t, a_t)$ is taken to be a constant variable, and where γ is the discount factor. We parametrize Q_ψ as the neural network scoring model from above. We name this policy *Q-learning*.

The dataset for training the policy consists of 499,757 transitions, of which 70% are used for training and 30% for evaluation. The model parameters are initialized from the *Supervised AMT* parameters and then optimized w.r.t. its loss function with gradient decent using Adam. The model is trained with early stopping on the hold-out set. No other regularization is used. The following discount factors are considered: $\gamma \in \{0.1, 0.2, 0.5\}$. The hyperparameters are selected based on the squared error loss on the hold-out set given by (113).

4.11.2 State Abstraction Policy

State abstraction, or *state aggregation*, is a well-known approach in reinforcement learning, where similar states are grouped together to form a new, reduced MDP (Bean et al., 1987; Bertsekas and

Castanon, 1989; Dietterich, 2000; Jong and Stone, 2005; Jiang et al., 2015). After the states have been grouped together, a reinforcement learning algorithm (e.g. Q-learning) can then be applied to learn a policy in the reduced MDP with the hope that fewer iterations and examples are needed to learn an effective policy.

The state abstraction approach is related to the *Bottleneck Simulator*. In contrast to state abstraction, in the *Bottleneck Simulator* the grouping is applied exclusively within the approximate transition model while the agent's policy operates on the complete, observed state $s_t \in S$. This enables the *Bottleneck Simulator* to reduce the impact of errors caused by disparate states being grouped together. Since the *Bottleneck Simulator* policy has access to the complete, observed state, it may counter such disparate groupings by optimizing for myopic, next-step rewards. The *Bottleneck Simulator* allows a deep neural network policy to learn its own high-level, distributed representations of the complete, observed state. In particular, the *Bottleneck Simulator* policy may be initialized using the parameters of another policy operating on the complete, observed state, such as the *Supervised AMT* policy trained on AMT labels. Finally, it should be noted that both state aggregation and the *Bottleneck Simulator* may incorporate knowledge about the structural properties of the environment.

We propose a state abstraction policy, which learns an approximate, tabular state-action-value function Q_ψ operating on the set of abstract states and actions:

$$Q_\psi(s_t, a_t) = \psi_{f_{s\to z}(s_t),\, h_{a\to a_{\text{abs}}}(a_t)} \tag{114}$$

where $f_{s\to z}$ is the classifier mapping dialogue histories $s \in S$ to bottleneck states $z \in Z$ defined earlier with 60 bottleneck states, $h_{a\to a_{\text{abs}}}$ is a deterministic classifier mapping actions (candidate responses) a to abstract actions $a_{\text{abs}} \in \mathcal{A}$ with 52 abstract actions, and ψ are the policy parameters s.t. $\psi \in \mathbb{R}^{|Z|\times|\mathcal{A}|} = \mathbb{R}^{60\times 52}$. The set of abstract actions $\mathcal{A}$ is defined by the Cartesian product:

$$\mathcal{A} = \mathcal{A}_{\textit{Response model class}} \times \mathcal{A}_{\textit{Wh-question}} \times \mathcal{A}_{\textit{Generic response}}, \tag{115}$$

where $\mathcal{A}_{\textit{Response model class}}$ is a one-hot vector representing which of the 13 response model classes generated the response, $\mathcal{A}_{\textit{Wh-question}} = \{True, False\}$ is a binary variable, which is *True* if the response contains a wh-question (e.g. a *what* or *why* word), and $\mathcal{A}_{\textit{Generic response}} = \{True, False\}$ is a binary variable, which is *True* if the response only contains stop-words.[36]

Since this is a tabular policy, we update its parameters using eq. (52). Given a state s_t, an action a_t, a reward r_t, and a consecutive state s_{t+1}, the parameters are updated by:

$$\begin{aligned}\psi_{f_{s\to z}(s_t),\, h_{a\to a_{\text{abs}}}(a_t)} &\leftarrow (1-\alpha)\psi_{f_{s\to z}(s_t),\, h_{a\to a_{\text{abs}}}(a_t)} + \alpha\hat{\psi}_{f_{s\to z}(s_t),\, h_{a\to a_{\text{abs}}}(a_t)} \\ \text{where } \hat{\psi}_{f_{s\to z}(s_t),\, h_{a\to a_{\text{abs}}}(a_t)} &= r_t + \gamma \max_{a'_{\text{abs}}\in\mathcal{A}} \psi_{f_{s\to z}(s_{t+1}),\, a'_{\text{abs}}},\end{aligned} \tag{116}$$

[36] To reduce sparsity, some of the similar models have been grouped together in the one-hot vector representation.

where $\alpha > 0$ is the learning rate and γ is the discount factor.

Similar to the *Bottleneck Simulator* policy, this policy is trained with Q-learning using rollout simulations from the *Bottleneck Simulator* environment model. The discount factor is set to the same as the learned *Bottleneck Simulator* policy. The training procedure is the same as the *Bottleneck Simulator* policy. We name this policy *State Abstraction*.

4.12 Experiments

4.12.1 Evaluation Based on Crowdsourced Data and Rollout Simulations

We first evaluate the learned response selection policies by using the AMT crowdsourced dataset and by rollout simulations in the *Bottleneck Simulator* environment model. Since these experiments do not involve real-world users, we are able to evaluate all the policies presented earlier.

Crowdsourced Evaluation: We evaluate the learned policies using the AMT test dataset. For each dialogue history and corresponding set of labelled candidate responses, we measure the score label $1-5$ given to the candidate response selected by the learned policies.[37] We compute the average score for each policy across all unique dialogues in the AMT test dataset. We report 95% confidence intervals estimated under the assumption that the scores are normally-distributed. In addition, we also measure how often each policy selects each of the 22 response models inside the ensemble system. We evaluate all the policies presented earlier.

Rollout Simulation Evaluation: We evaluate the performance of each policy w.r.t. rollout simulations in the *Bottleneck Simulator* environment model. Although the *Bottleneck Simulator* environment model is not an accurate representation of the real world, it has been trained with maximum log-likelihood on nearly 500,000 recorded transitions. Therefore, the simulations may be interpreted as a first order approximation of how a policy would perform when interacting with real-world users. However, this interpretation does not apply to the *Bottleneck Simulator* and *State Abstraction* policies, which have utilized rollout simulations from the *Bottleneck Simulator* environment model during their training. It is possible that the these two policies might be overfitting the *Bottleneck Simulator* environment model, which in turn might cause this evaluation to overestimate their performance. Therefore, we should not consider a superior performance of either of these two policies here to imply better performance.

We rollout 500 simulated episodes under each policy and compute the average return and average reward per time step (i.e. per system response).[38] The rollouts are carried out on the held-out

[37] Due to the nature of the crowdsourcing evaluation, all neural network scoring modules select their candidate responses by fixing the parameters of the output layer to those of *Supervised AMT*, as described in section 4.7.

[38] The average reward per time step is computed by first averaging the rewards in each episode and then averaging the average rewards across episodes.

Table 12: Policy evaluation w.r.t. average crowdsourced scores ($\pm$ 95% confidence intervals), and average return and reward per time step computed from 500 rollouts in the *Bottleneck Simulator* environment model ($\pm$ 95% confidence intervals). Triangle ▲ indicates policy is initialized from Supervised policy feed-forward neural network and hence yield same performance w.r.t. crowdsourced human scores.

	Crowdsourced	**Simulated Rollouts**	
Policy	**Human Score**	**Return**	**Avg Reward**
Evibot + Alicebot	2.25 ± 0.05	-11.33 ± 1.09	-0.29 ± 0.02
Supervised AMT	$\mathbf{2.63 \pm 0.06}$	$\mathbf{-6.46 \pm 0.70}$	$\mathbf{-0.15 \pm 0.01}$
Supervised AMT Learned Reward	-▲	-24.19 ± 2.04	-0.73 ± 0.02
REINFORCE	-▲	-7.30 ± 0.78	-0.16 ± 0.01
REINFORCE Learned Reward	-▲	-10.19 ± 0.98	-0.28 ± 0.02
Bottleneck Simulator	-▲	-6.54 ± 0.70	$\mathbf{-0.15 \pm 0.02}$
Q-learning	-▲	-6.70 ± 0.65	$\mathbf{-0.15 \pm 0.01}$
State Abstraction	1.85 ± 0.05	-13.04 ± 1.18	-0.35 ± 0.02

dataset of recorded dialogue transitions (i.e. only states $s \in \bar{S}$, which occur in the held-out dataset are sampled during rollouts). We report 95% confidence intervals estimated under the assumption that the returns and rewards are normally-distributed. We evaluate all the policies presented earlier.

Results: The results are given in Table 12. On the crowdsourced evaluation, all policies significantly outperform the *Evibot + Alicebot* baseline policy, which achieves only a modest average score of 2.25, with the exception of the *State Abstraction* policy, which achieves an even lower average score of 1.85. This shows that supervised learning utilizing crowdsourced human annotations has helped learn effective policies. Further, it shows that *State Abstraction* is an inferior approach, which may be explained by the fact that it does not utilize the crowdsourced human annotations to learn a policy operating on the complete, observed state space and complete action space. More generally, state abstraction may be inadequate for complex sequential decision making problems involving very high dimensional state and action spaces, such as selecting responses for a non-goal-driven dialogue system.

On the rollout simulation evaluation, the three policies achieving the highest average returns are *Supervised AMT*, *Bottleneck Simulator* and *Q-learning*. These policies achieve average returns ranging from -6.70 to -6.47. Since the *Bottleneck Simulator* policy performs similarly to the other policies, it is possible that the policy has not overfitted the *Bottleneck Simulator* environ-

ment model. If this is the case, the results indicate that these three policies might perform best in real-world interactions with users. Following these policies, the *REINFORCE* and *REINFORCE Learned Reward* policies appear to perform best, with average returns ranging from -10.19 to -7.30. Last in comes the *Evibot + Alicebot*, *State Abstraction* and *Supervised AMT Learned Reward* policies, with lower average returns and lower average rewards. As noted earlier, this again shows that the *State Abstraction* policy is inferior compared to the other policies. Finally, the poor performance of the *Supervised AMT Learned Reward* policy suggests that the learned reward model might be inaccurate.

Figure 20 shows the frequency with which the *Supervised AMT*, *REINFORCE* and *Bottleneck Simulator* policies select candidate responses from the response models in the ensemble system. Figure 19 shows a contingency table indicating the difference in the response models selected by the *Supervised AMT* policy and *Bottleneck Simulator* policy. The first figure shows that *REINFORCE* tends to strongly prefer *Alicebot* responses over other models. The *Alicebot* responses are in general the most topic-dependent and generic responses in the ensemble system. This suggests that the *REINFORCE* policy has learned to follow a highly *risk averse strategy*. In contrast, the *Bottleneck Simulator* policy selects *Alicebot* responses with a substantially lower frequency compared to both the *REINFORCE* and *Supervised AMT* policies. It appears that the *Bottleneck Simulator* policy instead prefers responses involving interesting and fun facts (*"BoWFactGenerator"*) as well as responses retrieved from Washington Post (*"Washington Post Models"*) and from Google search results (*"LSTMClassifierMSMarco"*). These responses have more semantic content and may have the potential to increase user engagement while discussing a particular topic, but they are also more risky. An inappropriate response here could cause significant user frustration and lower the overall user satisfaction. This indicates that the *Bottleneck Simulator* policy has learned a more *risk tolerant strategy*. This might be explained by the fact that the *Bottleneck Simulator* policy is trained using simulations. By learning from simulations, the policy may have been able to explore new actions and discover high-level strategies lasting multiple dialogue turns. In particular, the policy may have been able to experiment with riskier actions and to learn *remediation* or *fall-back strategies*, which may be applied when a risky action fails. This might in turn explain its stronger preference for candidate responses containing interesting and fun facts (*"BoWFactGenerator"*), which might act as a fall-back strategy where the policy distracts the user to recover from an inappropriate response. Such a strategy would be difficult to learn for the *REINFORCE* policy, because the sequence of actions exhibited by such high-level strategies are rarely observed in the dataset. Even if such action sequences were observed frequently, the corresponding user scores would likely suffer from high variance. Finally, the first figure shows that among the three policies the *Bottleneck Simulator* policy has the strongest preference for *Initiatorbot*, which outputs *"conversation starter"* responses. This is explained by the second figure, which shows

that the *Bottleneck Simulator* policy has selected responses from *Initiatorbot* instead of responses from *Alicebot* and *Elizabot*. This could indicate that the *Bottleneck Simulator* policy aims to follow a *system-initiative strategy*, where the system maintains control of the conversation by asking questions, changing topics and so on.

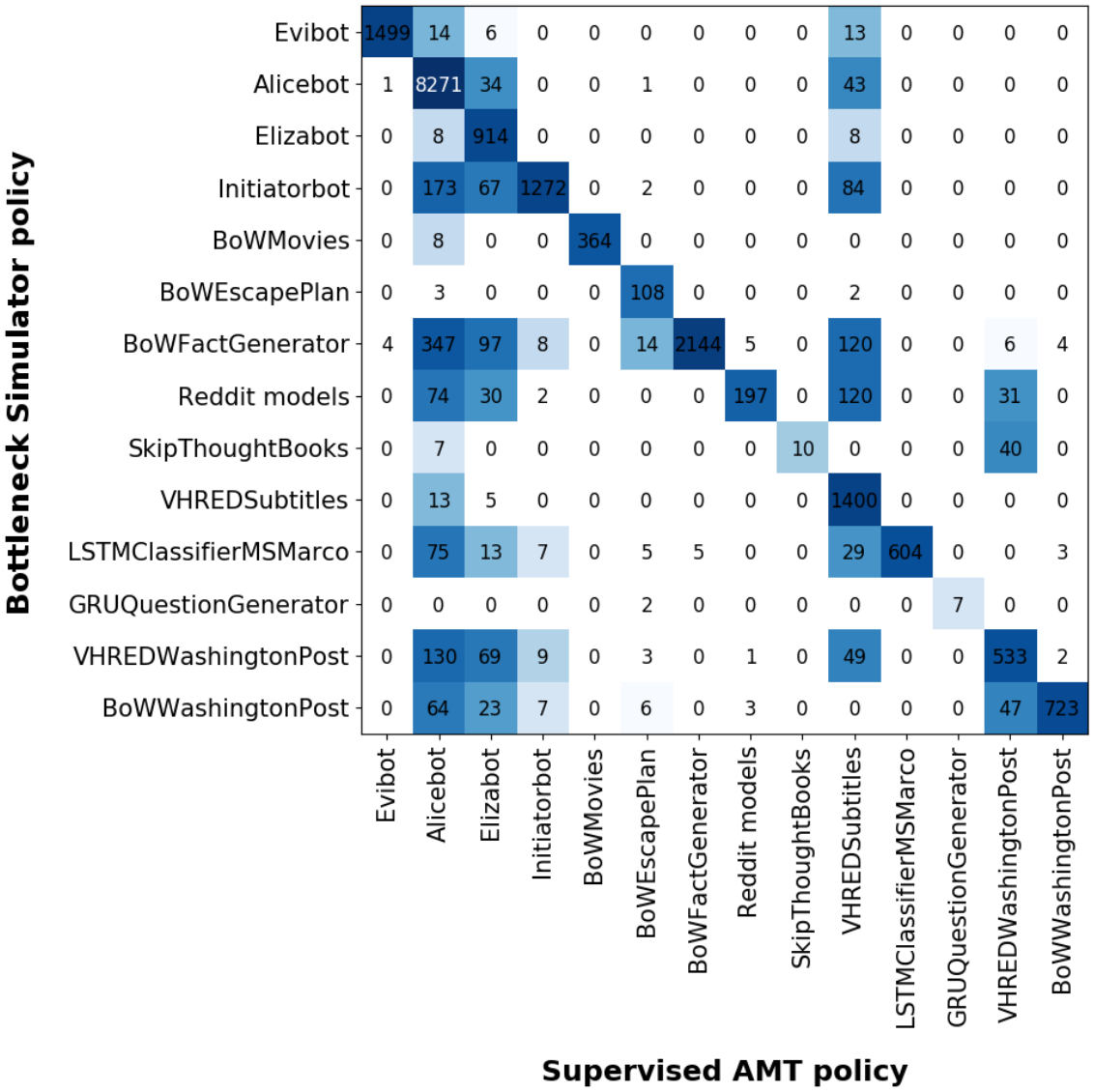

Figure 19: Contingency table comparing selected response models between *Supervised AMT* and *Bottleneck Simulator*. The cells in the matrix show the number of times the *Supervised AMT* policy selected the row response model and the *Bottleneck Simulator* policy selected the column response model. The cell frequencies were computed by simulating 500 episodes under the *Bottleneck Simulator* environment model. Further, it should be noted that all models retrieving responses from Reddit have been agglomerated into the class *Reddit models*.

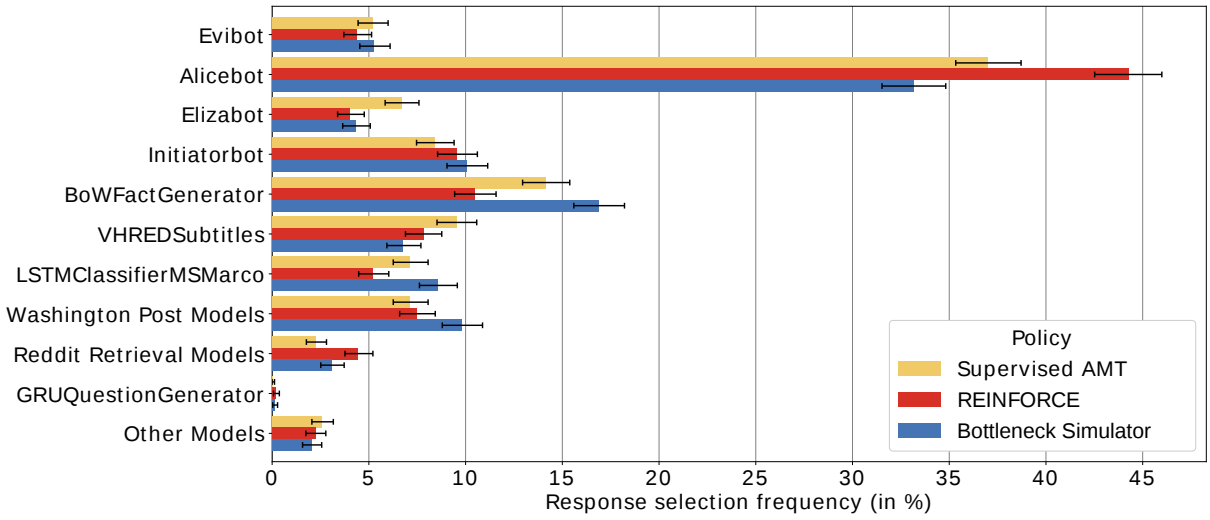

Figure 20: Response model selection probabilities across response models for *Supervised AMT*, *REINFORCE* and *Bottleneck Simulator* on the AMT label test dataset. 95% confidence intervals are shown based on the Wilson score interval for binomial distributions.

4.12.2 Real-World User Experiments

Setup: We next evaluate the response selection policies by carrying out A/B testing experiments with real-world users in the Amazon Alexa Prize 2017 competition. For each A/B testing experiment, we evaluate multiple response selection policies. Alexa users invoke the Alexa device's chatting functionality by saying *"Alexa, let's chat!"*. After invoking the chatting functionality, Alexa users start a conversation with a socialbot selected at random. Alexa users who start a conversation with the Milabot system are then assigned to a response selection policy at uniform random. After, their dialogues and their final subjective scores (in the range $1-5$) are recorded. We will consider the subjective user scores and the length of the dialogues as the main performance indicators, since they are both reflective of the overall user experience.

A/B testing enables us to accurately compare different response selection policies by keeping most other experimental factors constant. This is the advantage of A/B testing compared to other evaluation methods, such as the alternative method of evaluating system performance over time as the system changes.[39] Nevertheless, the distribution of Alexa users still changes through time both during and across A/B testing experiments. The distribution of users is likely to depend on the times of day, weekday and holiday season. In addition, user expectations change through time as

[39]When different systems are being evaluated at different points in time, it is often difficult to assess the improvement or degradation of performance w.r.t. specific system modifications.

users interact with other socialbots in the competition. In other words, the Alexa user distribution follows a non-stationary stochastic process. Motivated by this observation, we take two steps to reduce confounding factors in the experiments. First, we evaluate all policies of interest simultaneously during each A/B testing experiment. This should result in approximately the same number of users interacting with each policy w.r.t. time of day and weekday, and minimize the effect of a changing user distribution on user scores given in that period. However, since the user distribution changes between A/B testing experiments, it not possible to accurately compare policy performance across A/B testing experiments. Second, we discard returning users in the experiments. In other words, if a user has interacted with the Milabot system multiple times, we will only consider their first interaction. The reason is that returning users are likely to be influenced by their previous interactions with the system. For example, a user who previously had a positive experience with the system may be biased towards giving a high score in their next interaction. Furthermore, users who return to the system are likely to belong to a particular subpopulation of users, who may have more free time and be more willing to engage with socialbots than other users. By discarding returning users, we ensure that the evaluation is not biased towards this subpopulation of users. Finally, we exclude dialogues where users did not provide a score at the end. This biases the evaluation, since users who do not provide a score are likely to have been dissatisfied with the system or to have been expecting different functionality (e.g. *non-conversational activities*, such as playing games or taking quizzes). In principle, this last issue can be solved by having all the dialogues evaluated by a third-party (for example, asking human annotators on AMT to evaluate the dialogue). This is left for future work.

Policies Evaluated: The A/B testing experiments evaluate the following policies: *Evibot + Alicebot*, *Supervised AMT*, *Supervised AMT Learned Reward*, *REINFORCE*, *REINFORCE Learned Reward* and *Bottleneck Simulator*. The A/B testing experiments do not include the *Q-learning* and *State Abstraction* policies presented earlier. The reason for not including them was three-fold. First, during the Amazon Alexa Prize 2017 competition, only a small number of users interacted with the system on a daily basis. Therefore, it was infeasible to both evaluate all policies with real-world users and obtain statistically significant results. Second, there were several prior arguments for why both the *Q-learning* and *State Abstraction* policies would not work well for this problem. The Q-learning algorithm tends to be sample inefficient in many practical applications (Deisenroth and Rasmussen, 2011; Schulman et al., 2015) (see also (Tu and Recht, 2018)). Similarly, state abstraction methods may be inadequate for complex sequential decision making problems involving very high dimensional state and action spaces. In particular, state abstraction methods with discrete state and action spaces may suffer significantly from the curse of dimensionality (Richard, 1961; Bishop, 2006). Finally, the experiments in section 4.12.1 were conducted after the A/B testing experiments presented here. As such, there were no prior experiments for deciding on the set of

response selection policies to include in the A/B testing experiments.

Chronological Overview: The first A/B testing experiment was conducted between July 29th, 2017 and August 6th, 2017. This experiment evaluated the six policies: *Evibot + Alicebot*, *Supervised AMT*, *Supervised AMT Learned Reward*, *REINFORCE*, *REINFORCE Learned Reward* and *Bottleneck Simulator*. The *REINFORCE* and *REINFORCE Learned Reward* policies used the greedy action selection mechanism defined in eq. (96). This experiment occurred early in the Amazon Alexa Prize 2017 competition, which might imply that Alexa users had few expectations towards the capabilities of the socialbot systems. The experiment period also overlapped with the summer holidays in the United States, which might have lead to more children interacting with system than during other seasons.

The second A/B testing experiment was conducted between August 6th, 2017 and August 15th, 2017. This experiment evaluated the two policies: *REINFORCE* and *Bottleneck Simulator*. As before, the *REINFORCE* policy used the greedy action selection mechanism defined in eq. (96). This experiment occurred at the end of the competition semi-finals. At this point many Alexa users had already interacted with other socialbots in the competition, and therefore were likely to have developed expectations towards the socialbots. The period August 6th - August 15th overlapped with the end of the summer holidays and the beginning of the school year in the United States, which might have lead to less children interacting with the system than in the first A/B testing experiment.

The third A/B testing experiment was conducted between August 15th, 2017 and August 21st, 2017. Based on the results in the first and second A/B testing experiments, we decided to continue testing the policies *REINFORCE* and *Bottleneck Simulator*. As before, the *REINFORCE* policy used the greedy action selection mechanism defined in eq. (96). This experiment occurred after the end of the competition semi-finals. As pointed out earlier, this means that it is likely that many Alexa users had already developed expectations towards the socialbot systems. This A/B testing experiment period was entirely within the beginning of the school year in the United States, which might have lead to less children interacting with the system than in the previous two A/B testing experiments.

User Satisfaction Results: The main results for the three A/B testing experiments are given in Table 13. The table shows the average user scores, average dialogue length, average percentage of positive user utterances and average percentage of negative user utterances. In total, over 2800 user scores were collected after discarding returning users. Scores were collected after the end of the semi-finals competition, where all scores had been transcribed by human annotators. In addition, the aggregated results for the Amazon Alexa Prize 2017 competition are given in Table 14. However, it is impossible to accurately or fairly compare these aggregated results with the Milabot system results because the aggregated results are based on an average across many different

Table 13: A/B testing results ($\pm$ 95% confidence intervals). The superscript * indicates statistical significance at a 95% confidence level.

	Policy	User score	Dialogue length	Pos. utterances	Neg. utterances
Exp #1	*Evibot + Alicebot*	2.86 ± 0.22	31.84 ± 6.02	$2.80\% \pm 0.79$	$5.63\% \pm 1.27$
	Supervised AMT	2.80 ± 0.21	34.94 ± 8.07	$\mathbf{4.00\% \pm 1.05}$	$8.06\% \pm 1.38$
	Supervised AMT Learned Reward	2.74 ± 0.21	27.83 ± 5.05	$2.56\% \pm 0.70$	$6.46\% \pm 1.29$
	REINFORCE	2.86 ± 0.21	$\mathbf{37.51 \pm 7.21}$	$3.98\% \pm 0.80$	6.25 ± 1.28
	REINFORCE Learned Reward	2.84 ± 0.23	34.56 ± 11.55	$2.79\% \pm 0.76$	$6.90\% \pm 1.45$
	Bottleneck Simulator	$\mathbf{3.15 \pm 0.20}^*$	30.26 ± 4.64	$3.75\% \pm 0.93$	$\mathbf{5.41\% \pm 1.16}$
Exp #2	*REINFORCE*	$\mathbf{3.06 \pm 0.12}$	$\mathbf{34.45 \pm 3.76}$	$3.23\% \pm 0.45$	$7.97\% \pm 0.85$
	Bottleneck Simulator	2.92 ± 0.12	31.84 ± 3.69	$\mathbf{3.38\% \pm 0.50}$	$\mathbf{7.61\% \pm 0.84}$
Exp #3	*REINFORCE*	3.03 ± 0.18	30.93 ± 4.96	2.72 ± 0.59	7.36 ± 1.22
	Bottleneck Simulator	$\mathbf{3.06 \pm 0.17}$	$\mathbf{33.69 \pm 5.84}$	$\mathbf{3.63 \pm 0.68}$	$\mathbf{6.67 \pm 0.98}$

Table 14: Amazon Alexa Prize semi-finals average team statistics provided by Amazon.

Policy	User score	Dialogue length
All teams	2.92	22
Non-finalist teams	2.81	22
Finalist teams	**3.31**	**26**

systems, including the system variants in the A/B testing experiments and other socialbots which invoked *non-conversational activities* in order to increase user engagement (e.g. playing games and taking quizzes).

Table 13 shows that both *REINFORCE* and *Bottleneck Simulator* perform better than the other policies evaluated in the first A/B testing experiment. In particular, *Bottleneck Simulator* obtains an average user score of 3.15, which is significantly higher than all other policies at a 95% statistical significance level w.r.t. a one-tailed two-sample t-test. In comparison, the second best performing policies are the *Evibot + Alicebot* and *REINFORCE*, which both obtained an average user score of 2.86. To put this difference of $3.15 - 2.86 = 0.29$ score points in perspective, the average user score of the non-finalist and finalist teams in the competition are 2.81 and 3.31 respectively, yielding a difference of $3.31 - 2.81 = 0.50$ score points. This indicates that learning a high-performing response selection policy may play a significant role in improving the overall user satisfaction.

These results confirm the hypothesis that deep reinforcement learning and, in particular, model-based deep reinforcement learning utilizing a probabilistic generative model may be an effective method for learning a response selection policy. Furthermore, the results indicate that designing the *Bottleneck Simulator* environment model to include abstract, high-level semantic structure (i.e. a bottleneck state) could yield a sample efficient reinforcement learning approach.

For the second and third A/B testing experiments, the table shows that both the *REINFORCE* and *Bottleneck Simulator* policies perform well. In the second experiment, *REINFORCE* performs best obtaining an average user score of 3.06. In the third experiment, *Bottleneck Simulator* performs best obtaining an average user score of 3.06. However, the differences between the average scores of the two policies here are not statistically significant. The performance difference compared to the first A/B testing experiment could be due to the change in user profiles and user expectations. At this point in time, many of the Alexa users had interacted with socialbots from other teams and these socialbots had also been evolving. It is therefore likely that the user expectations were higher now. Furthermore, since the summer holidays in the United States ended during these experiments, less children and more adults were likely to interact with the socialbots. It is plausible that these adults had higher expectations towards the system and that they were less playful and less tolerant towards mistakes. Given this change in user profiles and expectations, the risk tolerant strategy learned by the *Bottleneck Simulator* policy is likely to have faired worse compared to the risk averse strategy learned by the *REINFORCE* policy. Although these results do not favour one policy over another, the results still support the hypothesis that deep reinforcement learning is a promising approach for learning an effective response selection policy.

The average dialogue length is also an important metric, because it reflects the overall user engagement and response coherency. On this metric, the *REINFORCE* policy performs best in the first and second A/B testing experiments by maintaining an average dialogue length of 37.51 and 34.45 respectively. This supports the hypothesis that deep reinforcement learning is a promising approach for learning an effective response selection policy. However, the result may also be explained by the risk averse strategy learned by *REINFORCE*. By utilizing a risk averse strategy, it is possible that the policy may be capable of continuing conversations longer by making generic, coherent responses and avoiding catastrophic mistakes. However, this advantage may come at the price of lower user engagement, since generic responses are likely to lead to lower user engagement. If this is the case, then the higher average dialogue length may not necessarily indicate a better performing policy.

Finally, Table 13 shows that the *Bottleneck Simulator* policy consistently obtains less user utterances with negative sentiment than all other policies. In the second and third A/B testing experiments, the *Bottleneck Simulator* policy also obtained more user utterances with positive sentiment compared to the *REINFORCE* policy. This also confirms the hypothesis that model-based deep re-

Table 15: First A/B testing experiment topical specificity and coherence by policy. The columns are average number of noun phrases per system utterance (System NPs), average number of overlapping words between the user's utterance and the system's response (This Turn), and average number of overlapping words between the user's utterance and the system's response in the next turn (Next Turn). Stop words are excluded. 95% confidence intervals are also shown.

		Word Overlap	
Policy	**System NPs**	**This Turn**	**Next Turn**
Evibot + Alicebot	1.05 ± 0.05	7.33 ± 0.21	2.99 ± 1.37
Supervised AMT	1.75 ± 0.07	10.48 ± 0.28	10.65 ± 0.29
Supervised AMT Learned Reward	1.50 ± 0.07	8.35 ± 0.29	8.36 ± 0.31
REINFORCE	1.45 ± 0.05	9.05 ± 0.21	9.14 ± 0.22
REINFORCE Learned Reward	1.04 ± 0.06	7.42 ± 0.25	7.42 ± 0.26
Bottleneck Simulator	$\mathbf{1.98 \pm 0.08}$	$\mathbf{11.28 \pm 0.30}$	$\mathbf{11.52 \pm 0.32}$

inforcement learning, based on an estimated environment model incorporating abstract, high-level semantic structure, may be an effective approach for learning a response selection policy.

Topical Specificity and Coherence Results: In order to further understand the differences between the learned policies, we carry out an analysis of the topical specificity and coherence of the different policies. The analysis aims to quantify how often each policy stays on topic (e.g. the frequency with which each policy selects responses on the current topic) and the topical specificity of the response content (e.g. how frequently the policy selects generic, topic-independent responses). This analysis is carried out at the utterance level, where the number of data examples is an order of magnitude larger compared to the number of user scores.

The results of this analysis are shown in Table 15. We measure topic specificity given by the average number of noun phrases per system utterance.[40] The more topic-specific a system utterance is, the higher we would expect this metric to be. We measure topic coherence by two metrics: the word overlap between the user's utterance and the immediate next system's response, and the word overlap between the user's utterance and the system's response at the next dialogue turn. The more a policy prefers to stay on topic, the higher we would expect these two metrics to be.

Table 15 shows that the *Bottleneck Simulator* policy obtains significantly higher metric scores w.r.t. both the average number of noun phrases per system utterance and the two word overlap metrics. This indicates that the *Bottleneck Simulator* policy selects system responses with the

[40]We use https://spacy.io version 1.9.0 to detect noun phrases with the package *"en_core_web_md-1.2.1"*.

highest topical coherency among all six policies, and that it generates the most topic-specific and semantically rich responses. This is in agreement with our previous conclusion that the *Bottleneck Simulator* policy follows a risk tolerant strategy. Next in line, comes the *Supervised AMT* policy, which also appears to maintain high topic specificity and coherence according to all three metrics. Then, come the *Off-policy REINFORCE* and *Off-policy REINFORCE Learned Reward*, policies, which select responses with significantly less noun phrases and word overlap compared to both the *Bottleneck Simulator* policy and the *Supervised AMT* policy. This is also in agreement with our previous conclusion, where we found that *REINFORCE* follows a risk averse strategy.[41] Finally, the baseline policy *Evibot + Alicebot* selects responses with the least noun phrases and word overlap among all policies. Overall, these results confirm that model-based deep reinforcement learning, based on an estimated environment model incorporating abstract, high-level semantic structure, is an effective approach for learning a response selection policy.

Analyzing Why Conversations Terminate: Finally, we conducted an experiment to investigate why users terminate their conversations with the Milabot system. This experiment aims to quantify if certain patterns are predictive of when a conversation will terminate. The ability to predict if a conversation is about to terminate could also serve as an important input feature for a policy. A policy could potentially use this input feature to adjust its responses in order to continue the conversation longer (e.g. by switching topics or by saying a joke to recover from an inappropriate response), which in turn might increase the overall user satisfaction. In addition, the same feature could be used to improve the reward shaping technique described in section 4.9.

The experiment is designed as follows. Given the dialogue history of six dialogue turns as input (three system utterances and three user utterances), a binary classification model is trained to predict if the next turn will be the last turn of the dialogue. A dataset is constructed for training the classification model based on the recorded dialogues between Alexa users and different variants of the Milabot system. For all recorded dialogues of sufficient length, a positive example is created from the last eight turns of the dialogue, and a negative example is created with a contiguous sequence of six turns selected at uniform random from the rest of dialogue. A neural network is given features of the first six dialogue turns as input and trained to predict the binary output by optimizing its parameters with maximum log-likelihood. For each input utterance five categorical variables are computed. The first variable is the normalized utterance length. The second variable

[41] For these experiments, we consider changing the topic of the dialogue to be a risk averse move. The reason is that many of the response models (e.g. *Alicebot*, *Initiatorbot*, *BoWFactGenerator*) respond with pre-defined answers as they change topics (for example, *BoWFactGenerator* might change topic by saying *"Did you know that male rabbits are called bucks, females are does?"*). These pre-defined answers are grammatically correct and will rarely if ever result in a disastrous user experience, but they limit the conversation to a superficial level and may not be entertaining to users.

Table 16: Accuracy of models predicting if a conversation will terminate using different features.

Random Baseline	Utterance Length	Sentiment	Speech Recognition Confidence	Word Embeddings	Combined
50.0%	70.1%	57.6%	51.4%	75.4%	75.8%

is the sentiment class. The third variable is a binary value indicating if the minimum confidence of the speech recognition system for any word in the utterance was less than 0.1 (on the interval $[0, 1]$). The fourth and fifth variables are the same as the third variable, but with thresholds set to 0.2 and 0.4 respectively. These variables are concatenated together with the average Word2Vec word embeddings (Mikolov et al., 2013b) of the utterance to form a feature vector. The feature vector for each utterance is a 305-length vector. The input to the neural network is the concatenation of these feature vectors for all three utterances of each interlocutor, yielding a $2 \times 3 \times 305$ length vector.

In order to quantify which patterns are predictive of when a conversation will terminate, variants of the model are trained by masking out different input features. The first model masks all features except for the utterance length feature. The second model masks all features except for the sentiment feature. The third model masks all features except for the speech recognition confidence features. The fourth model masks all features except for the Word2Vec word embedding features. The fifth model doesn't mask any features. In addition to these five models, a random baseline model is also included in the experiment. This model obtains 50% accuracy, since there is an equal amount of positive and negative examples in the test dataset.

Table 16 shows the accuracies different models obtain. The table shows that the features based on utterance length and word embeddings are most predictive of whether the dialogue will end in the next turn. Since the utterance length is predictive of whether the dialogue will terminate or not, it seems likely that short user utterances might indicate low user engagement and, in addition, make it difficult for the system to return an engaging response. Since word embeddings are predictive of whether the dialogue will terminate or not, it seems likely that the semantics of the user's utterance might be predictive of the user's engagement level. For example, a user changing topics might indicate boredom or frustration on the user's side. As another example, a user answering with a generic, topic-independent response (e.g. *"OK"* or *"I don't know"*) might also indicate boredom and, in addition, make it difficult for the system to return an entertaining response.

4.13 Discussion

In this second part of the thesis, we have explored different approaches to building a dialogue system combining representation learning and reinforcement learning. We have presented a deep reinforcement learning dialogue system, called Milabot. Milabot participated in the Amazon Alexa Prize 2017 competition, where the goal was to build a spoken non-goal-driven dialogue system (a socialbot) conversing coherently and engagingly with humans on popular topics, in order to maximize user satisfaction. Milabot processes the human interlocutor's input together with the dialogue history using an ensemble of 22 response models, including generative and retrieval-based models, in order to generate a set of candidate responses. After the candidate responses have been generated, the response selection policy selects a candidate response to emit to the user. The goal of the response selection policy is to select the response, which maximizes the overall user satisfaction. The problem of learning an effective response selection policy may be posed as a sequential decision making problem, since the system response emitted at one point in the dialogue will affect both the remainder of the dialogue and the overall user satisfaction.

We have proposed to parametrize the response selection policy as a neural network and proposed six different algorithms for learning the corresponding model parameters. The first approach learns the model parameters using supervised learning on crowdsourced human annotations. The second approach is based on deep reinforcement learning with the Q-learning algorithm. The third approach learns an approximate state-action-value function based on the observed user scores. The fourth and fifth approaches learn the response selection policy parameters using a class of reinforcement learning algorithms called REINFORCE. Finally, inspired both by the work in the first part of this thesis and state abstraction methods in reinforcement learning, the sixth approach learns the response selection policy parameters using Q-learning from rollout simulations in a simulated stochastic environment. The simulated environment is based on an estimated transition model of the environment, which utilizes an abstract, bottleneck state incorporating high-level semantic information about the dialogue in order to increase sample efficiency. This approach is called the *Bottleneck Simulator*. In addition to these six algorithms, we have also proposed two baseline response selection policies: one baseline policy based on a set of heuristic, hand-crafted rules, and one based on a state abstraction reinforcement learning algorithm.

We have evaluated all the proposed response selection policies against each other in multiple A/B testing experiments with real-world users. The results from these experiments show that the *Bottleneck Simulator* policy performs either better than or on par with all the other policies. A deeper analysis of the conversations with real-world users further reveals that the *Bottleneck Simulator* policy maintains the highest topical coherency and generates the most topic-relevant responses. These conclusions are supported by additional experiments on the crowdsourced hu-

man annotations and by rollout simulations in the simulated environment. Overall, the results suggest that there is much to be gained from learning an approximate environment simulating the interactions between human and machine interlocutors, which incorporates high-level semantic information.

In order to properly interpret the experiment results, it is important to also discuss the issues and limitations of the research presented in this chapter. As discussed earlier, the distribution over real-world users, including their expectations and prior experience, changed throughout the A/B testing experiments. In particular, it is highly likely that user expectations were affected by other socialbots in the competition, including their expectations to talk about certain topics (e.g. news article topics) or to play social games (e.g. personality quizzes), and by Amazon's marketing campaign to drive users to try the socialbots. Despite the fact that each A/B testing experiment evaluated the response selection policies in parallel, these issues represent a substantial group of confounding factors and may have biased the users to prefer the behaviour of one policy over others, which in turn would affect the experiment conclusions. Another important confounding factor is the effect of speech recognition errors on the observed user satisfaction w.r.t. different policies. Some policies may be more adept at handling speech recognition errors than others (for example, by emphasizing the conversation topic more than the words present in the user's last utterance). This could make one policy appear to perform better than other policies.

An important limitation of the experiments lies in the underspecified format of the conversations. All conversations begin from a blank slate with no anchoring points or common ground established between interlocutors. Unfortunately, there is also no mechanism for introducing external media, such as news articles or songs. Furthermore, even if a user returns to talk to the same socialbot, their conversation history is not available to the system. This makes the competition very difficult, as the socialbots have to both establish a topic and a common ground, in addition to conducting the conversation. This stands in contrast with the Ubuntu and Twitter tasks presented earlier in this thesis, where the generative model only has to generate the next response in a conversation and where the topic and some of the common ground has already been established. On a related point, other researchers have explored anchoring conversations based on Wikipedia articles (Burtsev et al., 2018; Zhou et al., 2018; Dinan et al., 2019) and user profiles (Zhang et al., 2018).

Another important limitation of the experiments is caused by their rigid turn-taking structure. Although the conversations are spoken, it is impossible for the user to interrupt the system and, vice-versa, for the system to interrupt the user. This distorts the conversation and may have an adverse effect on the user's expectations and behaviour. This can be handled by allowing socialbots to utilize incremental dialogue processing and give responses at any point in the conversation (Howes et al., 2011; Dethlefs et al., 2012).

4.14 Directions for Future Research

In this section, we will discuss several avenues for future research.

4.14.1 Rethinking The Non-Goal-Driven Dialogue Task

As discussed earlier, a major limitation of the Amazon Alexa Prize 2017 competition is the underspecified format of the conversations. The conversations have no anchoring points, no given topics, and no common ground established between interlocutors. This substantially increases the difficulty of the task. Furthermore, it makes it difficult to compare systems fairly. For example, consider the following two hypothetical systems. The first system starts the conversation by proposing a news article about political issues and then discusses these intelligently with the user. The second system asks the user to name their favorite movie, then conducts a binary trivia quiz by presenting them related facts, where the user has to say *"yes"* or *"no"* depending on if a fact is true or false, and finishes by telling the user how often they were right and saying goodbye. Both of these two systems appear to be strong contenders for the Amazon Alexa Prize 2017 competition, due to the underspecified format of the conversations. However, from a scientific point of view, what knowledge or insight would we have gained if the experiments with real-world users showed that one of these systems performed better than the other one? It seems likely that, at best, we might conclude that the user population of the experiments preferred one topic over another – regardless of the underlying components of each dialogue system, such as natural language understanding components, natural language generation components, response selection components, knowledge base components and so on. If that was the case, such an experiment would not help the research field progress significantly.

One approach to solving this problem is by anchoring conversations around Wikipedia articles (Burtsev et al., 2018; Zhou et al., 2018; Dinan et al., 2019). For example, in the Conversational Intelligence Challenge 2017 (ConvAI 2017) the interlocutors are shown a snippet of a Wikipedia article and asked to converse about it (Burtsev et al., 2018). This immediately anchors the conversation around a single topic, which makes the task of building an effective dialogue system easier and the comparison between different dialogue systems more fair and interpretable. This approach also allows to easily incorporate external information (e.g. information from other Wikipedia articles or from knowledge bases) and to break down dialogue system performance on a topic-by-topic basis. Another related approach is proposed by Zhang et al. (2018), where each interlocutor is shown a short description of a personality profile and then asked to converse while pretending to be a person with that profile. As such, this approach anchors the conversation around personalities. Unlike the ConvAI 2017, where the conversations tend to be more *fact or knowledge-driven*, in this task the conversations center around personal topics, such as hobbies, family, work, and so on.

4.14.2 Extensions of the Bottleneck Simulator

One of the most novel and promising methods, which we have presented, is the *Bottleneck Simulator*. This is a model-based reinforcement learning algorithm, which learns an approximate environment model by mapping a dialogue history $s_t \in S$ to an abstract, high-level semantic state $z_t \in Z$ (a *bottleneck state*) at every turn of the dialogue.

In the algorithm presented earlier, the set Z is defined by a Cartesian product:

$$Z = Z_{\text{Dialogue act}} \times Z_{\text{User sentiment}} \times Z_{\text{Generic user utterance}}, \tag{117}$$

where $Z_{\text{Dialogue act}}$ represents a set of dialogue acts, $Z_{\text{User sentiment}}$ represents a set of user sentiments and $Z_{\text{Generic user utterance}}$ represents a binary set indicating if the user's last utterance is generic. In total, Z has 60 states (i.e. $|Z| = 60$). Although this approach proved effective across many experiments, at best Z is capturing only a limited amount of high level information about the dialogue.

Therefore, the *Bottleneck Simulator* algorithm can be improved by expanding the set Z to include additional information. For example, the set Z could be redefined as the Cartesian product:

$$Z = Z_{\text{Dialogue act}} \times Z_{\text{User sentiment}} \times Z_{\text{Generic user utterance}} \times Z_{\text{Dialogue topic}} \times Z_{\text{User profile}}, \tag{118}$$

where $Z_{\text{Dialogue topic}}$ is a discrete set of dialogue topics and $Z_{\text{User profile}}$ is a discrete set of user profiles. Here, a model could be constructed to map dialogue histories to a discrete set of topics either by using a topic model, such as the Latent Dirichlet allocation (LDA) model (Blei et al., 2003), or by combining a clustering algorithm with a Word2Vec word embedding model, such as the Skip-Gram model (Mikolov et al., 2013a,b). Alternatively, the set of topics $Z_{\text{Dialogue topic}}$ could be defined as a set of real-valued numbers in n dimensions: $Z_{\text{Dialogue topic}} \in \mathbb{R}^n$ where $n \in \mathbb{N}$. In this case, the topic for a given dialogue $s_t \in S$ might be computed as the average of the Word2Vec word embeddings of the last k utterances in the dialogue. The corresponding learned environment model would require a separate model for predicting the real-valued vector of the next topic conditioned on the state z_t and s_t, system action a_t and label y_t. For the discrete set of user profiles $Z_{\text{User profile}}$, a model could similarly be constructed to map dialogue histories to a set of user profiles. For example, a regression model can be trained to predict the user's personality w.r.t. openness, conscientiousness, extroversion, agreeableness, and neuroticism (Golbeck et al., 2011) based on the user utterances. These personality traits can then be clustered into a set of n personalities.

To incorporate a richer and larger set of bottleneck states, it seems likely that the *Bottleneck Simulator* would require additional training data to estimate the learned environment model. One way to acquire additional training data is by utilizing other, related datasets, such as the Switchboard Corpus (Godfrey et al., 1992) or datasets extracted from online discussion forums, such as `www.reddit.com`. A simple way to utilize a related dataset is to first train the transition model

$P_\psi(z_{t+1} \mid z_t, s_t, a_t, y_t)$, given in eq. (111), on the external dataset and then fine-tune its parameters on the dialogues recorded with real-world users. As discussed earlier, the transition model can be evaluated on a hold-out dataset of dialogues recorded with real-world users.

However, if the external dataset consists of dialogues exclusively between human interlocutors or contains dialogues on highly different topics, then it may be necessary to formulate an approach for mitigating the differences between the external dataset and the dataset of dialogues recorded with real-world users interacting with the dialogue system. For example, one method for doing this is to assign an importance weight to each dialogue example in the external dataset. Dialogues in the external dataset should be assigned a high importance weight (e.g. 1.0) if they are similar to the dialogues of users interacting with the dialogue system. Vice versa, dialogues in the external dataset should be assigned a low importance weight (e.g. 0.0) if they are not similar to the dialogues of users interacting with the dialogue system. For example, the importance weights could be computed automatically by a binary classification model trained to distinguish between dialogues in the external dataset and dialogues with real-world users interacting with the dialogue system. Once the importance weights have been computed, they can be applied inside the training procedure for the transition model to reweight the external dialogue examples.

5 Conclusion

In this thesis, we have presented an investigation of representation learning methods for building dialogue systems and conversational agents, specifically based on deep learning and deep reinforcement learning. This work is motivated by the multitude of real-world applications, ranging from intelligent personal assistants and virtual friends to healthcare assistants and intelligent tutoring systems. As such, the purpose of this thesis is to make a contribution to the research fields of natural language understanding, natural language generation and representation learning, with the goal of building general-purpose natural language dialogue systems.

In the first part of the thesis, we investigated building probabilistic generative dialogue models based on deep learning. These probabilistic generative dialogue models are tasked with generating the next, appropriate response in a text-based dialogue conditioned on the history of the preceding turns. For this task, we proposed three novel models. The first model proposed is the Hierarchical Recurrent Encoder-Decoder (HRED) model. Motivated by the need to better model long-term, discourse-level context, this model incorporates the dialogue turn taking structure into its architecture. The second model proposed is the Multiresolution Recurrent Neural Network (MrRNN) model, which is motivated by the idea of modelling higher-level, abstract semantic information as a stochastic process. The model architecture is a stacked sequence-to-sequence model with an intermediate, stochastic representation (a coarse representation) capturing high-level semantic content. The last model proposed is the Latent Variable Recurrent Encoder-Decoder (VHRED) model. This model is a variant of the HRED model incorporating a continuous, stochastic, latent variables with the purpose of better modelling the ambiguity and uncertainty present in human language communication.

The three proposed models were evaluated on two domains: a goal-driven technical response generation task and a non goal-driven response generation task. The evaluation was conducted through human evaluation studies using on on a crowdsourcing platform and in a laboratory setting. In addition, the models were evaluated through qualitative evaluation and through automated evaluation metrics. The experiment results demonstrate that all models improved upon the baseline models. The HRED model was found to outperform a baseline model on the goal-driven dialogue task, where it generated relevant responses of high quality suggesting it is better capable of incorporating long-term discourse-level context. The MrRNN model was found to perform best on the goal-driven dialogue task, where it outperformed all baseline models as well as the HRED and VHRED models. Specifically, MrRNN was able to generate more relevant and fluent responses compared to the other models. These results demonstrate that MrRNN is a highly promising approach to response generation on goal-driven dialogue tasks. The success of MrRNN underlines the importance of modelling higher-level, abstract semantic information, and suggests that similar

approaches should be explored in future research. At the same time, the VHRED model was found to perform best on the non-goal-driven dialogue task, where it appeared to be better capable of generating long and semantically coherent responses compared to other models. This underlines the importance of modelling high-level latent structure and suggests that future research should investigate new methods for modelling natural language ambiguity and uncertainty. In summary, each of the three proposed models represent one small step forward in the pursuit of building generative dialogue models. Nevertheless, it seems likely that far more research is required in order to make this class of models more applicable to real-world problem settings.

In the second part of the thesis, we investigated combining deep learning and reinforcement learning by building a non-goal-driven deep reinforcement learning dialogue system for the Amazon Alexa Prize 2017 competition, which learns from real-world interactions with human interlocutors. This dialogue system is based on an ensemble of 22 response models, where each response model generates a candidate response conditioned on the dialogue history text. Given a set of candidate responses, the system's response selection policy selects a candidate response to emit to the user with the goal of maximizing the overall user satisfaction.

We parametrized the response selection policy as a neural network and proposed to learn the model parameters by framing the response selection task as a sequential decision making problem. We proposed six different algorithms for learning the model parameters. The first proposed algorithm learns the model parameters using supervised learning on crowdsourced human annotations. Another four proposed algorithms are based on methods from deep reinforcement learning, specifically Q-learning and REINFORCE algorithms. Finally, motivated by the generative dialogue models discussed earlier, we proposed an approach for learning the response selection policy parameters by using Q-learning from rollout simulations in a simulated stochastic environment. The simulated environment contains an estimated transition model of the environment and utilizes an abstract state representing high-level semantic information about the dialogue, in order to exploit structural properties of the environment and increase sample efficiency. This approach is called the Bottleneck Simulator. In addition to these proposed algorithms, we also presented two baseline response selection policies: one based on a set of heuristic, hand-crafted rules, and one based on a state abstraction reinforcement learning method.

We evaluated all proposed response selection policies through several A/B testing experiments with real-world users. The results from these experiments show that the Bottleneck Simulator policy performs better than or on par with all other policies. Furthermore, a quantitative analysis of the conversations shows that the Bottleneck Simulator policy maintains the highest topical coherency and generates the most topic-relevant responses. These conclusions are confirmed by additional experiments on the crowdsourced human annotations and by rollout simulations in the simulated environment. In summary, the results demonstrate the overall importance and utility

of modelling the interactions between the interlocutors in a stochastic framework incorporating high-level semantic information.

If we take a step back, a general pattern emerges of the work presented in this thesis. Throughout the thesis, we have repeatedly applied the framework of probabilistic generative models in order to understand the existing models and ideas in the literature, to understand their underlying assumptions and hypotheses, and to propose new models motivated by new hypotheses. We have used the "language of probabilistic generative models" to understand existing ideas and to formulate new ideas, such as modelling high-level semantic information as a stochastic process, capturing ambiguity and uncertainty in a stochastic, latent variable, and approximating an environment model through an abstract, high-level stochastic variable representing the dialogue state. For each new idea, we have proposed a new structure for a probabilistic generative model by modifying or adding stochastic variables and by making assumptions around these variables, such as their probabilistic dependencies, whether they are observed or latent, the type of information they aim to capture and their corresponding learning procedure. These ideas have proved fruitful and may, in turn, help to move the field a small step forward.

Bibliography

R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

D. Ameixa, L. Coheur, P. Fialho, and P. Quaresma. Luke, I am your father: dealing with out-of-domain requests by using movies subtitles. In *Intelligent Virtual Agents*, 2014.

S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, 2017.

L. E. Asri, J. He, and K. Suleman. A sequence-to-sequence model for user simulation in spoken dialogue systems. In *INTERSPEECH*, 2016.

P. Bachman and D. Precup. Data generation as sequential decision making. In *NIPS*, 2015.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

R. E. Banchs and H. Li. IRIS: a chat-oriented dialogue system based on the vector space model. In *ACL, System Demonstrations*, 2012.

S. Banerjee and A. Lavie. METEOR: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL, Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation And/Or Summarization*, 2005.

M. Baroni, R. Bernardi, and R. Zamparelli. Frege in space: A program of compositional distributional semantics. *LiLT (Linguistic Issues in Language Technology)*, 9, 2014.

J. Bayer and C. Osendorfer. Learning stochastic recurrent networks. In *NIPS, Workshop on Advances in Variational Inference*, 2014.

M. J. Beal. *Variational algorithms for approximate Bayesian inference*. University of London, 2003.

J. C. Bean, J. R. Birge, and R. L. Smith. Aggregation in dynamic programming. *Operations Research*, 35(2):215–220, 1987.

Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*. Springer, 2012.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *JMLR*, 3:1137–1155, 2003.

D. P. Bertsekas and D. A. Castanon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34(6):589–598, 1989.

D. P. Bertsekas and J. N. Tsitsiklis. Neuro-dynamic programming: an overview. In *IEEE Conference on Decision and Control, 1995*, volume 1. IEEE, 1995.

H. R. Bhat and T. A. Lone. Cortana - intelligent personal digital assistant: Review. *International Journal of Advanced Research in Computer Science*, 8, 2017.

S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

C. M. Bishop. Pattern recognition. *Machine Learning*, 2006.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

D. Bohus, A. Raux, T. K. Harris, M. Eskenazi, and A. I. Rudnicky. Olympus: an open-source framework for conversational spoken language interface research. In *ACL, Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, 2007.

A. Bordes, Y.-L. Boureau, and J. Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.

N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML*, 2012.

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

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

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. Retrieved 2019-01-01, https://www.businessinsider.com/stanford-therapy-chatbot-app-depression-anxiety-woebot-2018-1.

M. Burtsev, V. Logacheva, V. Malykh, I. V. Serban, R. Lowe, S. Prabhumoye, A. W. Black, A. Rudnicky, and Y. Bengio. The first conversational intelligence challenge. In *The NIPS'17 Competition: Building Intelligent Systems*. Springer, 2018.

J. Camacho-Collados and T. Pilehvar. From word to sense embeddings: A survey on vector representations of meaning. *JAIR*, 63, 2018.

R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz. Tell me when and why to do it! Run-time planner model updates via natural language instruction. In *ACM/IEEE International Conference on Human-Robot Interaction*, 2012.

K. Cao and S. Clark. Latent variable dialogue models and their diversity. In *EACL*, 2017.

F. Charras, G. D. Duplessis, V. Letard, A.-L. Ligozat, and S. Rosset. Comparing system-response retrieval models for open-domain and casual conversational agent. In *Workshop on Chatbots and Conversational Agent Technologies*, 2016.

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.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In *NIPS*, 2015.

G. Chung. Developing a flexible spoken dialog system using simulation. In *ACL*, 2004.

J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *NIPS*, 2015.

H. H. Clark and S. E. Brennan. Grounding in communication. *Perspectives on socially shared cognition*, 13(1991):127–149, 1991.

K. M. Colby. Modeling a paranoid mind. *Behavioral and Brain Sciences*, 4:515–534, 1981.

H. Cuayáhuitl. Simpleds: A simple deep reinforcement learning dialogue system. In *Dialogues with Social Robots*. Springer, 2017.

H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira. Human-computer dialogue simulation using hidden Markov models. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*. IEEE, 2005.

N. Dahlbäck, A. Jönsson, and L. Ahrenberg. Wizard of oz studies - why and how. *Knowledge-based systems*, 6(4):258–266, 1993.

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

I. P. Debater. IBM Research, Project Debater. *Research, IBM*, 2018. Retrieved 2018-12-24, https://www.research.ibm.com/artificial-intelligence/project-debater/index.html.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391, 1990.

M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, 2011.

N. Dethlefs, H. Hastie, V. Rieser, and O. Lemon. Optimising incremental dialogue decisions using information density for interactive systems. In *EMNLP / CoNLL*, 2012.

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.

T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *JAIR*, 13:227–303, 2000.

R. Dillet. Hugging Face wants to become your artificial BFF. *Tech Crunch*, 2016. Retrieved 2018-12-24, https://techcrunch.com/2017/03/09/hugging-face-wants-to-become-your-artificial-bff/.

E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston. Wizard of wikipedia: Knowledge-powered conversational agents. In *ICLR*, 2019.

J. Dowding, J. M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran. Gemini: A natural language system for spoken-language understanding. In *HLT*, 1993.

D. Eck and J. Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing, 2002*, 2002.

W. Eckert, E. Levin, and R. Pieraccini. User modeling for spoken dialogue system evaluation. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997.

O. Fabius and J. R. van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

M. Fatemi, L. E. Asri, H. Schulz, J. He, and K. Suleman. Policy networks with two-stage training for dialogue systems. In *SIGDIAL*, 2016.

L. Ferrone and F. M. Zanzotto. Symbolic, distributed and distributional representations for natural language processing in the era of deep learning: a survey. *arXiv preprint arXiv:1702.00764*, 2017.

D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. Building Watson: An overview of the DeepQA project. *AI magazine*, 31(3), 2010.

J. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*, 2016.

G. Forgues, J. Pineau, J.-M. Larchevêque, and R. Tremblay. Bootstrapping dialog systems with word embeddings. In *NIPS, Workshop on Modern Machine Learning and Natural Language Processing*, 2014.

V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Number 10 in 1. Springer series in statistics New York, NY, USA:, 2001.

D. Furness. Baidu releases Melody, a medical assistant chatbot to keep physicians humming. *Digital Trends*, 2016. Retrieved 2019-01-01, https://www.digitaltrends.com/health-fitness/baidu-melody-medical-chatbot/.

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. In *ACL*, 2015.

M. Gašić, F. Jurčíček, B. Thomson, K. Yu, and S. Young. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

M. Gasic, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young. On-line policy optimisation of Bayesian spoken dialogue systems via human interaction. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

K. Georgila and D. Traum. Reinforcement learning of argumentation dialogue policies in negotiation. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

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

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2011.

J. J. Godfrey, E. C. Holliman, and J. McDaniel. Switchboard: Telephone speech corpus for research and development. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1. IEEE, 1992.

J. Golbeck, C. Robles, M. Edmondson, and K. Turner. Predicting personality from twitter. In *IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom)*. IEEE, 2011.

I. Goodfellow, A. Courville, and Y. Bengio. *Deep Learning*. MIT Press, 2016. URL http://goodfeli.github.io/dlbook/.

J. T. Goodman. A bit of progress in language modeling extended version. *Machine Learning and Applied Statistics Group Microsoft Research. Technical Report, MSR-TR-2001-72*, 2001.

A. L. Gorin, G. Riccardi, and J. H. Wright. How may I help you? *Speech Communication*, 23(1): 113–127, 1997.

A. Graves. Sequence transduction with recurrent neural networks. In *ICML RLW*, 2012.

A. Graves. Generating sequences with recurrent neural networks. *arXiv:1308.0850*, 2013.

K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2017.

K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *ICLR*, 2015.

S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In *ICML*, 2016.

S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346, 1990.

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

P. A. Heeman. Representing the reinforcement learning state in a negotiation dialogue. In *IEEE Workshop on Automatic Speech Recognition & Understanding*, 2009.

J. Henderson, O. Lemon, and K. Georgila. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34(4):487–511, 2008.

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

C. Howes, M. Purver, P. G. Healey, G. Mills, and E. Gregoromichelaki. On incrementality in dialogue: Evidence from compound contributions. *Dialogue & Discourse*, 2(1):279–311, 2011.

Y. Ji, G. Haffari, and J. Eisenstein. A latent variable recurrent neural network for discourse relation language models. In *NAACL-HLT*, 2016.

N. Jiang, A. Kulesza, and S. Singh. Abstraction selection in model-based reinforcement learning. In *ICML*, 2015.

N. K. Jong and P. Stone. State abstraction discovery from irrelevant state variables. In *IJCAI*, 2005.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *JAIR*, 4: 237–285, 1996.

A. Kamath and R. Das. A survey on semantic parsing. In *Conference on Automated Knowledge Base Construction (AKBC)*, 2019.

C. Kamm. User interfaces for voice applications. *National Academy of Sciences*, 92(22):10031–10037, 1995.

H. Khouzaimi, R. Laroche, and F. Lefevre. Incremental human-machine dialogue simulation. In *Dialogues with Social Robots*. Springer, 2017.

D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *NIPS*, 2015.

P. Koehn and C. Monz. Manual and automatic evaluation of machine translation between european languages. In *Workshop on Statistical Machine Translation, ACL*, 2006.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. *ICML*, 2016.

L. Kuvayev and R. S. Sutton. Model-based reinforcement learning with an approximate, learned model. In *Ninth Yale Workshop on Adaptive and Learning Systems*. Citeseer, 1996.

T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998a.

T. K. Landauer, D. Laham, and P. W. Foltz. Learning human-like knowledge by singular value decomposition: A progress report. In *NIPS*, 1998b.

A. Lazaridou, N. T. Pham, and M. Baroni. Towards multi-agent communication-based language learning. *arXiv preprint arXiv:1605.07133*, 2016.

A. Lazaridou, A. Peysakhovich, and M. Baroni. Multi-agent cooperation and the emergence of (natural) language. In *ICLR*, 2017.

S. Lee and M. Eskenazi. POMDP-based let's go system for spoken dialog challenge. In *Spoken Language Technology Workshop (SLT)*, 2012.

O. Lemon and O. Pietquin. Machine learning for spoken dialogue systems. In *INTERSPEECH*, 2007.

E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23, 2000.

M. Lewis, D. Yarats, Y. N. Dauphin, D. Parikh, and D. Batra. Deal or No Deal? End-to-End Learning for Negotiation Dialogues. In *EMNLP*, 2017.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. In *NAACL*, 2016a.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A persona-based neural conversation model. In *ACL*, 2016b.

J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky. Deep reinforcement learning for dialogue generation. In *EMNLP*, 2016c.

J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky. Adversarial learning for neural dialogue generation. In *EMNLP*, 2017.

Y. Li and T. Yang. Word embedding for understanding natural language: A survey. In *Guide to Big Data Applications*. Springer, 2018.

L.-J. Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

Z. C. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

B. Liu and I. Lane. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2017.

C.-W. Liu, R. Lowe, I. V. 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 *EMNLP*, 2016.

G. López, L. Quesada, and L. A. Guerrero. Alexa vs. siri vs. cortana vs. google assistant: a comparison of speech-based natural user interfaces. In *International Conference on Applied Human Factors and Ergonomics*. Springer, 2017.

R. López-Cózar. Automatic creation of scenarios for evaluating spoken dialogue systems via user-simulation. *Knowledge-Based Systems*, 106:51–73, 2016.

R. Lowe, N. Pow, I. Serban, L. Charlin, and J. Pineau. Incorporating unstructured textual knowledge sources into neural dialogue systems. In *NIPS, Workshop on Machine Learning for Spoken Language Understanding*, 2015a.

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 *SIGDIAL*, 2015b.

R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau. On the evaluation of dialogue systems with next utterance classification. In *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. In *ACL*, 2017a.

R. T. 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), 2017b.

M. T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. In *ACL*, 2015.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *SemEval Workshop, COLING*, 2014.

J. Markoff and P. Mozur. For Sympathetic Ear, More Chinese Turn to Smartphone Program. *New York Times*, 2015.

B. McCann, J. Bradbury, C. Xiong, and R. Socher. Learned in translation: Contextualized word vectors. In *NIPS*, 2017.

H. Mei, M. Bansal, and M. R. Walter. Coherent dialogue with attention-based language models. In *AAAI*, 2017.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013b.

T. Mikolov et al. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*, 2017.

S. Miller, R. Bobrow, R. Ingria, and R. Schwartz. Hidden understanding models of natural language. In *ACL*, 1994.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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, 2015.

S. Mohan and J. Laird. Learning goal-oriented hierarchical tasks from situated interactive instruction. In *AAAI*, 2014.

A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, 1993.

I. Mordatch and P. Abbeel. Emergence of grounded compositional language in multi-agent populations. *AAAI*, 2018.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 1999.

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268*, 2016.

B. D. Nye, A. C. Graesser, and X. Hu. Autotutor and family: A review of 17 years of natural language tutoring. *International Journal of Artificial Intelligence in Education*, 24(4):427–469, 2014.

O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, N. Schneider, and N. A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *NAACL*, 2013.

T. Paek. Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment. In *INTERSPEECH, Dialog-on-Dialog Workshop*, 2006.

I. Papaioannou, A. C. Curry, J. L. Part, I. Shalyminov, X. Xu, Y. Yu, O. Dusek, V. Rieser, and O. Lemon. Alana: Social dialogue using an ensemble model and a ranker trained on user feedback. In *Alexa Prize Proceedings*, 2017.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL*, 2002.

B. H. Partee. *Compositionality in formal semantics: Selected papers*. John Wiley & Sons, 2008.

P. Parthasarathi and J. Pineau. Extending neural generative conversational model using external knowledge sources. In *EMNLP*, 2018.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML*, 28, 2012.

B. Peng, X. Li, J. Gao, J. Liu, and K.-F. Wong. Integrating planning for task-completion dialogue policy learning. In *ACL*, 2018.

J. Peng and R. J. Williams. Efficient learning and planning within the dyna framework. *Adaptive Behavior*, 1(4):437–454, 1993.

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

R. Pieraccini, E. Tzoukermann, Z. Gorelov, J.-L. Gauvain, E. Levin, C.-H. Lee, and J. G. Wilpon. A speech understanding system based on statistical representation of semantics. In *ICASSP*. IEEE, 1992.

R. Pieraccini, D. Suendermann, K. Dayanidhi, and J. Liscombe. Are we there yet? research in commercial spoken dialog systems. In *Text, Speech and Dialogue*, 2009.

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. S. Polydoros and L. Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

D. Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, 2000.

D. Precup, R. S. Sutton, and S. Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, 2001.

D. Prylipko, D. Schnelle-Walka, S. Lord, and A. Wendemuth. Zanzibar openivr: an open-source framework for development of spoken dialog systems. In *International Conference on Text, Speech and Dialogue*, 2011.

W. V. O. Quine. *Word and object*. MIT press, 2013.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*, 2018.

A. Ram, R. Prasad, C. Khatri, A. Venkatesh, R. Gabriel, Q. Li, J. Nunn, B. Hedayatnia, M. Heng, A. Nagar, E. King, K. Bland, A. Wartick, Y. Pan, H. Song, S. Jayadevan, G. Hwang, and A. Pettigrue. Conversational ai: The science behind the alexa prize. In *Alexa Prize Proceedings*, 2017.

A. Raux, D. Bohus, B. Langner, A. W. Black, and M. Eskenazi. Doing research on a deployed spoken dialogue system: One year of let's go! experience. In *Ninth International Conference on Spoken Language Processing*, 2006.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

B. Richard. Adaptive control processes: A guided tour, 1961.

A. Ritter, C. Cherry, and W. B. Dolan. Data-driven response generation in social media. In *EMNLP*, 2011a.

A. Ritter, S. Clark, O. Etzioni, et al. Named entity recognition in tweets: an experimental study. In *EMNLP*, 2011b.

V. Rus and M. Lintean. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *ACL, Workshop on Building Educational Applications Using NLP*, 2012.

J. Schatzmann, K. Georgila, and S. Young. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *SIGDIAL*, 2005.

J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *HLT / ACL*, 2007.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, 2015.

S. Seneff. Tina: A natural language system for spoken language applications. *Computational linguistics*, 18(1):61–86, 1992.

I. Serban, A. Ororbia, J. Pineau, and A. C. Courville. Piecewise latent variables for neural variational text processing. In *EMNLP*, 2017a.

I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, 2016.

I. V. Serban, T. Klinger, G. Tesauro, K. Talamadupula, B. Zhou, Y. Bengio, and A. Courville. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, 2017b.

I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke, et al. A Deep Reinforcement Learning Chatbot. *arXiv preprint arXiv:1709.02349*, 2017c.

I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. *AAAI*, 2017d.

I. V. Serban, R. Lowe, P. Henderson, L. Charlin, and J. Pineau. A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, 9(1):1–49, 2018.

S. Shaikh, T. Strzalkowski, S. Taylor, and N. Webb. VCA: an experiment with a multiparty virtual chat agent. In *ACL, Workshop on Companionable Dialogue Systems*, 2010.

L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. In *Association for Computational Linguistics*, 2015.

B. A. Shawar and E. Atwell. Chatbots: are they really useful? In *LDV Forum*, volume 22, 2007.

P. Simon. *Too Big to Ignore: The Business Case for Big Data*, volume 72. John Wiley & Sons, 2013.

S. Singh, D. Litman, M. Kearns, and M. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *JAIR*, 2002.

S. P. Singh, M. J. Kearns, D. J. Litman, and M. A. Walker. Reinforcement learning for spoken dialogue systems. In *NIPS*, 1999.

N. Slonim. Project debater. *Computational Models of Argument: Proceedings of COMMA 2018*, 305:4, 2018.

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 *International on Conference on Information and Knowledge Management*. ACM, 2015a.

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. In *NAACL-HLT*, 2015b.

A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. V. Ess-Dykema, and M. Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373, 2000.

B. Stone and S. Soper. Amazon Unveils a Listening, Talking, Music-Playing Speaker for Your Home. *Bloomberg L.P*, 2014. Retrieved 2014-11-07.

P.-H. Su, D. Vandyke, M. Gašić, D. Kim, N. Mrkšić, T.-H. Wen, and S. Young. Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. In *INTERSPEECH*, 2015.

P.-H. Su, M. Gasic, N. Mrksic, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*, 2016.

D. Suendermann-Oeft, V. Ramanarayanan, M. Teckenbrock, F. Neutatz, and D. Schmidt. Halef: An open-source standard-compliant telephony-based modular spoken dialog system: A review and an outlook. In *Natural language dialog systems and intelligent assistants*. Springer, 2015.

S. Sukhbaatar, R. Fergus, et al. Learning multiagent communication with backpropagation. In *NIPS*, 2016.

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

R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*, 1990.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Number 1 in 1. MIT Press Cambridge, 1998.

G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3): 58–68, 1995.

D. Traum, S. C. Marsella, J. Gratch, J. Lee, and A. Hartholt. Multi-party, multi-issue, multi-strategy negotiation for multi-modal virtual agents. In *International Workshop on Intelligent Virtual Agents*, 2008.

H. P. Truong, P. Parthasarathi, and J. Pineau. Maca: A modular architecture for conversational agents. In *SIGDIAL*, 2017.

S. Tu and B. Recht. The gap between model-based and model-free methods on the linear quadratic regulator: An asymptotic viewpoint. *arXiv preprint arXiv:1812.03565*, 2018.

L. Vilnis and A. McCallum. Word representations via gaussian embedding. *ICLR*, 2015.

R. S. Wallace. The anatomy of alice. *Parsing the Turing Test*, 2009.

W. Ward and S. Issar. Recent improvements in the CMU spoken language understanding system. In *ACL, Workshop on Human Language Technology*, 1994.

C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

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.

T.-H. Wen, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, D. Vandyke, and S. Young. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, 2017.

M. Werning, W. Hinzen, and E. Machery. *The Oxford handbook of compositionality*. Oxford Handbooks in Linguistic, 2012.

J. Weston, S. Chopra, and A. Bordes. Memory networks. *ICLR*, 2015.

Y. Wilks and D. Fass. The preference semantics family. *Computers & Mathematics with Applications*, 23(2-5):205–221, 1992.

J. Williams, A. Raux, D. Ramachandran, and A. Black. The dialog state tracking challenge. In *SIGDIAL*, 2013.

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, A. Raux, and M. Henderson. Introduction to the special issue on dialogue state tracking. *Dialogue & Discourse*, 7(3):1–3, 2016.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4), 1992.

T. Wit. The Story of Siri, by its founder Adam Cheyer. *Blog Post*, 2014. Retrieved 2018-12-24, https://medium.com/wit-ai/the-story-of-siri-by-its-founder-adam-cheyer-3ca38587cc01.

J. Wortham. Will Google's Personal Assistant Be Creepy or Cool? *The New York Times*, 2012. Retrieved 2013-01-30, http://bits.blogs.nytimes.com/2012/06/28/will-googles-personal-assistant-be-creepy-or-cool/.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

S. Young, M. Gasic, B. Thomson, and J. D. Williams. POMDP-based statistical spoken dialog systems: A review. *IEEE*, 101(5):1160–1179, 2013.

T. Young, E. Cambria, I. Chaturvedi, M. Huang, H. Zhou, and S. Biswas. Augmenting end-to-end dialog systems with commonsense knowledge. *arXiv preprint arXiv:1709.05453*, 2017.

Z. Yu, Z. Xu, A. W. Black, and A. I. Rudnicky. Strategy and policy learning for non-task-oriented conversational systems. In *SIGDIAL*, 2016.

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? In *ACL*, 2018.

T. Zhao and M. Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *SIGDIAL*, 2016.

T. Zhao, K. Lee, and M. Eskenazi. Dialport: Connecting the spoken dialog research community to real user data. In *IEEE, Spoken Language Technology Workshop (SLT)*, 2016.

T. Zhao, R. Zhao, and M. Eskenazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL*, 2017.

K. Zhou, S. Prabhumoye, and A. W. Black. A dataset for document grounded conversations. In *EMNLP*, 2018.

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, 2015.

I Appendix: Coarse Sequence Representations

This appendix describes the course sequence representations utilized by the MrRNN models. The text for this appendix has been adapted from the appendix in Serban et al. (2017b) written by the author of this thesis. The appendix text was never published in any journal, conference or workshop proceedings.

Nouns

The noun-based procedure for extracting coarse tokens aims to exploit high-level structure of natural language discourse. More specifically, it builds on the hypothesis that dialogues in general are topic-driven and that these topics may be characterized by the nouns inside the dialogues. At any point in time, the dialogue is centered around one or several topics. As the dialogue progresses, the underlying topic evolves as well. In addition to the tokenizer required by the previous extraction procedure, this procedure also requires a part-of-speech (POS) tagger to identify the nouns in the dialogue suitable for the language domain.

For extracting the noun-based coarse tokens, we define a set of 795 stop words for Twitter and 84 stop words for Ubuntu containing mainly English pronouns, punctuation marks and prepositions (excluding special placeholder tokens). We then extract the coarse tokens by applying the following procedure to each dialogue:

1. We apply the POS tagger version $0.3.2$ developed by Owoputi and colleagues Owoputi et al. (2013) to extract POS.[42] For Twitter, we use the parser trained on the Twitter corpus developed by Ritter et al. (2011b). For Ubuntu, we use the parser trained on the NPS Chat Corpus developed by Forsyth and Martellwhich was extracted from IRC chat channels similar to the Ubuntu Dialogue Corpus.[43] [44]

2. Given the POS tags, we remove all words which are not tagged as nouns and all words containing non-alphabet characters.[45] We keep all urls and paths.

3. We remove all stop words and all repeated tokens, while maintaining the order of the tokens.

[42] www.cs.cmu.edu/~ark/TweetNLP/

[43] As input to the POS tagger, we replace all unknown tokens with the word "something" and remove all special placeholder tokens (since the POS tagger was trained on a corpus without these words). We further reduce any consecutive sequence of spaces to a single space. For Ubuntu, we also replace all commands and entities with the word "something". For Twitter, we also replace all numbers with the word "some", all urls with the word "somewhere" and all heart *emoticons* with the word "love".

[44] Forsyth, E. N. and Martell, C. H. (2007). Lexical and discourse analysis of online chat dialog. In Semantic Computing, 2007. ICSC 2007. International Conference on, pages 19âĂŞ26. IEEE.

[45] We define nouns as all words with tags containing the prefix "NN" according to the PTB-style tagset.

Table 17: Unigram and bigram models bits per word on noun representations.

Model	**Ubuntu**	**Twitter**
Unigram	10.16	12.38
Bigram	7.26	7.76

4. We add the "no_nouns" token to all utterances, which do not contain any nouns. This ensures that no coarse sequences are empty. It also forces the coarse sub-model to explicitly generate at least one token, even when there are no actual nouns to generate.

5. For each utterance, we use the POS tags to detect three types of time tenses: past, present and future tenses. We append a token indicating which of the 3 tenses are present at the beginning of each utterance.[46] If no tenses are detected, we append the token "no_tenses".

As before, there exists a one-to-many alignment between the extracted coarse sequence tokens and the natural language tokens, since this procedure also maintains the ordering of all special placeholder tokens, with the exception of the "no_nouns" token.

We cut-off the vocabulary at 10000 coarse tokens for both the Twitter and Ubuntu datasets excluding the special placeholder tokens. On average a Twitter dialogue in the training set contains 25 coarse tokens, while a Ubuntu dialogue in the training set contains 36 coarse tokens.

Model statistics for the unigram and bigram language models are presented in Table 17 for the noun representations on the Ubuntu and Twitter training sets.[47] The table shows a substantial difference in bits per words between the unigram and bigram models, which suggests that the nouns are significantly correlated with each other.

Activity-Entity Pairs

The activity-entity-based procedure for extracting coarse tokens attempts to exploit domain specific knowledge for the Ubuntu Dialogue Corpus, in particular in relation to providing technical assistance with problem solving. Our manual inspection of the corpus shows that many dialogues are centered around *activities*. For example, it is very common for users to state a specific problem they want to resolve (e.g. *how do I install program X?* or *My driver X doesn't work, how do I fix it?*). In response to such queries, other users often respond with specific instructions (e.g. *Go to website X to download software Y* or *Try to execute command X*). In addition to the technical entities, the principle message conveyed by each utterance resides in the verbs (e.g. *install*, *work*,

[46] Note that an utterance may contain several sentences. It therefore often happens that an utterance contains several time tenses.

[47] The models were trained using maximum log-likelihood on the noun representations excluding all special tokens.

fix, *go*, *to*, *download*, *execute*). Therefore, it seems clear that a dialogue system must have a strong understanding of both the activities and technical entities if it is to effectively assist users with technical problem solving. It seems likely that this would require a dialogue system able to relate technical entities to each other (e.g. to understand that *firefox* depends on the *GCC* library) and conform to the temporal structure of activities (e.g. to understand that the *download* activity is often followed by *install* activity).

We therefore construct two word lists: one for activities and one for technical entities. We construct the activity list based on manual inspection yielding a list of 192 verbs. For each activity, we further develop a list of synonyms and conjugations of the tenses of all words. We also use Word2Vec word embeddings (Mikolov et al., 2013b), trained on the Ubuntu Dialogue Corpous training set, to identify commonly misspelled variants of each activity. The result is a dictionary, which maps a verb to its corresponding activity (if such exists). For constructing the technical entity list, we scrape publicly available resources, including Ubuntu and Linux-related websites as well as the Debian package manager *APT*. Similar to the activities, we also use the Word2Vec word embeddings to identify misspelled and paraphrased entities. This results in another dictionary, which maps one or two words to the corresponding technical entity. In total there are 3115 technical entities. In addition to this we also compile a list of 230 frequent commands.

Afterwards, we extract the coarse tokens by applying the following procedure to each dialogue:

1. We apply the technical entity dictionary to extract all technical entities.

2. We apply the POS tagger version $0.3.2$ developed by Owoputi and colleagues, trained on the NPS Chat Corpus developed by Forsyth and Martell as before. As input to the POS tagger, we map all technical entities to the token "something". This transformation should improve the POS tagging accuracy, since The corpus the parser was trained on does not contain technical words.

3. Given the POS tags, we extract all verbs which correspond to activities.[48]. If there are no verbs in an entire utterance and the POS tagger identified the first word as a noun, we will assume that the first word is in fact a verb. We do this, because the parser does not work well for tagging technical instructions in imperative form (e.g. *upgrade firefox*). If no activities are detected, we append the token "none_activity" to the coarse sequence. We also keep all urls and paths.

4. We remove all repeated activities and technical entities, while maintaining the order of the tokens.

[48]We define verbs as all words with tags containing the prefix "VB" according to the PTB-style tagset.

5. If a command is found inside an utterance, we append the "cmd" token at the end of the utterance. Otherwise, we append the "no_cmd" token to the end of the utterance. This enables the coarse sub-model to predict whether or not an utterance contains executable commands.

6. As for the noun-based coarse representation, we also append the time tense to the beginning of the sequence.

As before, there exists a one-to-many alignment between the extracted coarse sequence tokens and the natural language tokens, with the exception of the "none_activity" and "no_cmd" tokens.

Since the number of unique tokens are smaller than 10000, we do not need to cut-off the vocabulary. On average a Ubuntu dialogue in the training set contains 43 coarse tokens.

Our manual inspection of the extracted coarse sequences, show that the technical entities are identified with very high accuracy and that the activities capture the main intended action in the majority of utterances. Due to the high quality of the extracted activities and entities, we are confident that they may be used for evaluation purposes as well.

Scripts to generate the noun and activity-entity representations, and to evaluate the dialogue responses w.r.t. activity-entity pairs are available online at: https://github.com/julianser/Ubuntu-Multiresolution-Tools/tree/master/ActEntRepresentation.

Table 18: Twitter Coarse Sequence Examples

Natural Language Tweets	Noun Representation
<first_speaker> at pinkberry with my pink princess enjoying a precious moment <url>	present_tenses pinkberry princess moment
	present_tenses alma emma bif sis hugs
<second_speaker>- they are adorable, alma still speaks about emma bif sis . hugs	
<first_speaker> <at> when you are spray painting, where are you doing it ? outside ? in your apartment ? where ?	present_tenses spray painting apartment
	present_tenses spray stuff bathroom
<second_speaker> <at> mostly spray painting outside but some little stuff in the bathroom .	

Table 19: Ubuntu Coarse Sequence Examples

Natural Language Dialogues	Activity-Entity Coarse Dialogues
if you can get a hold of the logs, there 's stuff from **unknown** about his inability to install amd64	future_tenses get_activity install_activity amd64_entity no_cmd
I'll check fabbione 's log, thanks sounds like he had the same problem I did ew, why ? ...	no_tenses check_activity no_cmd past_present_tenses none_activity no_cmd no_tenses none_activity no_cmd ...
upgrade lsb-base and acpid	no_tenses upgrade_activity lsb_entity acpid_entity no_cmd
i'm up to date	no_tenses none_activity no_cmd
what error do you get ?	present_tenses get_activity no_cmd
i don't find error :/ where do i search from ? acpid works, but i must launch it manually in a root sterm ...	present_tenses discover_activity no_cmd present_future_tenses work_activity acpid_entity root_entity no_cmd ...

Stop Words for Noun-based Coarse Tokens

Ubuntu stop words for noun-based coarse representation:

all another any anybody anyone anything both each each other either everybody everyone everything few he her hers herself him himself his I it its itself many me mine more most much myself neither no one nobody none nothing one one another other others ours ourselves several she some somebody someone something that their theirs them themselves these they this those us we what whatever which whichever who whoever whom whomever whose you your yours yourself yourselves . , ? ’ - – !

Twitter stop words for noun-based coarse representation: [49]

all another any anybody anyone anything both each each other either everybody everyone everything few he her hers herself him himself his I it its itself many me mine more most much myself neither no one nobody none nothing one one another other others ours ourselves several she some somebody someone something that their theirs them themselves these they this those us we what whatever which whichever who whoever whom whomever whose you your yours yourself yourselves . , ? ’ - – !able about above abst accordance according accordingly across act actually added adj adopted affected affecting affects after afterwards again against ah all almost alone along already also although always am among amongst an and announce another any anybody anyhow anymore anyone anything anyway anyways anywhere apparently approximately are aren arent arise around as aside ask asking at auth available away awfully b back bc be became because become becomes becoming been before beforehand begin beginning beginnings begins behind being believe below beside besides between beyond biol bit both brief briefly but by c ca came can cannot can’t cant cause causes certain certainly co com come comes contain containing contains cos could couldnt d date day did didn didn’t different do does doesn doesn’t doing don done don’t dont down downwards due during e each ed edu effect eg eight eighty either else elsewhere end ending enough especially et et-al etc even ever every everybody everyone everything everywhere ex except f far few ff fifth first five fix followed following follows for former formerly forth found four from further furthermore g game gave get gets getting give given gives giving go goes going gone gonna good got gotten great h had happens hardly has hasn hasn’t have haven haven’t having he hed hence her here hereafter hereby herein heres hereupon hers herself hes hey hi hid him himself his hither home how howbeit however hundred i id ie if i’ll im immediate immediately importance important in inc indeed index information instead into invention inward is isn isn’t it itd it’ll its itself i’ve j just k keep keeps kept keys kg km know known knows l ll largely last lately later latter latterly least less lest let lets like liked likely line little ll ’ll lol look looking looks lot ltd m made mate mainly make makes many may maybe me mean means meantime meanwhile merely mg might million miss ml more moreover most mostly mr mrs much mug must my myself n na name namely nay nd near nearly necessarily necessary need needs neither never nevertheless new next nine ninety no nobody non none nonetheless noone nor normally nos not noted nothing now nowhere o obtain obtained obviously of off often oh ok okay old omitted omg on once one ones only onto or ord other others otherwise ought our ours ourselves out outside over overall owing own p page pages part particular particularly past people per perhaps placed please plus poorly possible possibly potentially pp predominantly present previously primarily probably promptly proud provides put q que quickly quite qv r ran rather rd re readily really recent recently ref refs regarding regardless regards related relatively research respectively resulted resulting results right rt run s said same saw say saying says sec section see seeing seem seemed seeming seems seen self selves sent seven several shall she shed she’ll shes should shouldn shouldn’t show showed shown showns shows significant significantly similar similarly since six slightly so some somebody somehow someone somethan something sometime sometimes somewhat somewhere soon sorry specifically specified specify specifying state states still stop strongly sub substantially successfully such sufficiently suggest sup sure t take taken taking tbh tell tends th than thank thanks thanx that that’ll thats that’ve the their theirs them themselves then thence there thereafter thereby thered therefore therein there’ll thereof therere theres thereto thereupon there’ve these they theyd they’ll theyre they’ve thing things think this those thou though thoughh thousand throug through throughout thru thus til time tip to together too took toward towards tried tries truly try trying ts tweet twice two u un under unfortunately unless unlike unlikely until unto up upon ups ur us use used useful usefully usefulness uses using usually v value various ve ’ve very via viz vol vols vs w wanna want wants was wasn wasn’t way we wed welcome well we’ll went were weren weren’t we’ve what whatever what’ll whats when whence whenever where whereafter whereas whereby wherein wheres whereupon wherever whether which while whim whither who whod whoever whole who’ll whom whomever whos whose why widely will willing wish with within without won won’t words world would wouldn wouldn’t www x y yeah yes yet you youd you’ll your youre yours yourself yourselves you’ve z zero

[49] Part of these were extracted from https://github.com/defacto133/twitter-wordcloud-bot/blob/master/assets/stopwords-en.txt.

Activities and Entities for Ubuntu Dialogue Corpus

Ubuntu activities:

accept, activate, add, ask, appoint, attach, backup, boot, check, choose, clean, click, comment, compare, compile, compress, change, affirm, connect, continue, administrate, copies, break, create, cut, debug, decipher, decompress, define, describe, debind, deattach, deactivate, download, adapt, eject, email, conceal, consider, execute, close, expand, expect, export, discover, correct, fold, freeze, get, deliver, go, grab, hash, import, include, install, interrupt, load, block, log, log-in, log-out, demote, build, clock, bind, more, mount, move, navigate, open, arrange, partition, paste, patch, plan, plug, post, practice, produce, pull, purge, push, put, queries, quote, look, reattach, reboot, receive, reject, release, remake, delete, name, replace, request, reset, resize, restart, retry, return, revert, reroute, scroll, send, set, display, shutdown, size, sleep, sort, split, come-up, store, signup, get-ahold-of, say, test, transfer, try, uncomment, de-expand, uninstall, unmount, unplug, unset, sign-out, update, upgrade, upload, use, delay, enter, support, prevent, loose, point, contain, access, share, buy, sell, help, work, mute, restrict, play, call, thank, burn, advice, force, repeat, stream, respond, browse, scan, restore, design, refresh, bundle, implement, programming, compute, touch, overheat, cause, affect, swap, format, rescue, zoomed, detect, dump, simulate, checkout, unblock, document, troubleshoot, convert, allocate, minimize, maximize, redirect, maintain, print, spam, throw, sync, contact, destroy

Ubuntu entities (excerpt):

ubuntu_7.04, dmraid, vnc4server, tasksel, aegis, mirage, system-config-audit, uif2iso, aumix, unrar, dell, hibernate, ucoded, finger, zoneminder, ucfg, macaddress, ia32-libs, synergy, aircrack-ng, pulseaudio, gnome, kid3, bittorrent, systemsettings, cups, finger, xchm, pan, uwidget, vnc-java, linux-source, ucommand.com, epiphany, avanade, onboard, uextended, substance, pmount, lilypond, proftpd, unii, jockey-common, aha, units, xrdp, mp3check, cruft, uemulator, ulivecd, amsn, ubuntu_5.10, acpidump, uadd-on, gpac, ifenslave, pidgin, soundconverter, kdelibs-bin, esmtp, vim, travel, smartdimmer, uactionscript, scrotwm, fbdesk, tulip, beep, nikto, wine, linux-image, azureus, vim, makefile, uuid, whiptail, alex, junior-arcade, libssl-dev, update-inetd, uextended, uaiglx, sudo, dump, lockout, overlay-scrollbar, xubuntu, mdk, mdm, mdf2iso, linux-libc-dev, sms, lm-sensors, dsl, lxde, dsh, smc, sdf, install-info, xsensors, gutenprint, sensors, ubuntu_13.04, atd, ata, fatrat, fglrx, equinix, atp, atx, libjpeg-dbg, umingw, update-inetd, firefox, devede, cd-r, tango, mixxx, uemulator, compiz, libpulse-dev, synaptic, ecryptfs, crawl, ugtk+, tree, perl, tree, ubuntu-docs, libsane, gnomeradio, ufilemaker, dyndns, libfreetype6, daemon, xsensors, vncviewer, vga, indicator-applet, nvidia-173, rsync, members, qemu, mount, rsync, macbook, gsfonts, synaptic, finger, john, cam, lpr, lpr, xsensors, lpr, lpr, screen, inotify, signatures, units, ushareware, ufraw, bonnie, nec, fstab, nano, bless, bibletime, irssi, ujump, foremost, nzbget, ssid, onboard, synaptic, branding, hostname, radio, hotwire, xebia, netcfg, xchat, irq, lazarus, pilot, ucopyleft, java-common, vm, ifplugd, ncmpcpp, irc, uclass, gnome, sram, binfmt-support, vuze, java-common, sauerbraten, adapter, login

Ubuntu commands:

alias, apt-get, aptitude, aspell, awk, basename, bc, bg, break, builtin, bzip2, cal, case, cat, cd, cfdisk, chgrp, chmod, chown, chroot, chkconfig, cksum, cmp, comm, command, continue, cp, cron, crontab, csplit, curl, cut, date, dc, dd, ddrescue, declare, df, diff, diff3, dig, dir, dircolors, dirname, dirs, dmesg, du, echo, egrep, eject, enable, env, eval, exec, exit, expect, expand, export, expr, false, fdformat, fdisk, fg, fgrep, file, find, fmt, fold, for, fsck, ftp, function, fuser, gawk, getopts, grep, groupadd, groupdel, groupmod, groups, gzip, hash, head, history, hostname, htop, iconv, id, if, ifconfig, ifdown, ifup, import, install, ip, jobs, join, kill, killall, less, let, link, ln, local, locate, logname, logout, look, lpc, lpr, lprm, ls, lsof, man, mkdir, mkfifo, mknod, more, most, mount, mtools, mtr, mv, mmv, nc, nl, nohup, notify-send, nslookup, open, op, passwd, paste, ping, pkill, popd, pr, printf, ps, pushd, pv, pwd, quota, quotacheck, quotactl, ram, rar, rcp, read, readonly, rename, return, rev, rm, rmdir, rsync, screen, scp, sdiff, sed, select, seq, set, shift, shopt, shutdown, sleep, slocate, sort, source, split, ssh, stat, strace, su, sudo, sum, suspend, sync, tail, tar, tee, test, time, timeout, times, touch, top, tput, traceroute, tr, true, tsort, tty, type, ulimit, umask, unalias, uname, unexpand, uniq, units, unrar, unset, unshar, until, useradd, userdel, usermod, users, uuencode, uudecode, vi, vmstat, wait, watch, wc, whereis, which, while, who, whoami, write, xargs, xdg-open, xz, yes, zip, admin, purge

II Appendix: Human Evaluation on Amazon Mechanical Turk (Twitter)

This appendix describes the human evaluation of the TF-IDF, RNNLM, HRED and VHRED models on the Twitter dataset described in the first part of the thesis. The text for this appendix has been adapted from the appendix in Serban et al. (2017d) written by the author of this thesis. The appendix text was never published in any journal, conference or workshop proceedings.

Setup

We choose to use crowdsourcing platforms such as AMT rather than carrying out in-lab experiments, even though in-lab experiments usually exhibit less idiosyncratic noise and result in higher agreement between human annotators. We do this because AMT experiments involve a larger and more heterogeneous pool of annotators, which implies less cultural and geographic biases, and because such experiments are easier to replicate, which we believe is important for benchmarking future research on these tasks.

Allowing the AMT human evaluators to not assign preference for either response is important. There are many reasons for why humans may not understand the dialogue context, such as topics they are not familiar with, slang language and non-English language. We refer to such evaluations as “indeterminable".

The evaluation setup resembles the classical Turing Test where human judges have to distinguish between human-human conversations and human-computer conversations. However, unlike the original Turing Test, we only ask human evaluators to consider the next utterance in a given conversation and we do not inform them that any responses were generated by a computer. Apart from minimum context and response lengths we impose no restrictions on the generated responses.

Selection Process

At the beginning of each experiment, we briefly instruct the human evaluator on the task and show them a simple example of a dialogue context and two potential responses. To avoid presentation bias, we shuffle the order of the examples and the order of the potential responses for each example. During each experiment, we also show four trivial “attention check" examples that any human evaluator who has understood the task should be able to answer correctly. We discard responses from human evaluators who fail more than one of these checks.

We select the examples shown to human evaluators at random from the test set. We filter out all non-English conversations and conversations containing offensive content. This is done by

automatically filtering out all conversations with non-ascii characters and conversations with profanities, curse words and otherwise offensive content. This filtering is not perfect, so we manually skim through many conversations and filter out conversations with non-English languages and offensive content. On average, we remove about $1/80$ conversations manually. To ensure that the evaluation process is focused on evaluating conditional dialogue response generation (as opposed to unconditional single sentence generation), we constrain the experiment by filtering out examples with fewer than 3 turns in the context. We also filter out examples where either of the two presented responses contain less than 5 tokens. We remove the special token placeholders and apply regex expressions to detokenize the text.

Execution

We run the experiments in batches. For each pairs of models, we carry out 3-5 human intelligence tests (HITs) on AMT. Each HIT contains 70-90 examples (dialogue context and two model responses) and is evaluated by 3-4 unique humans. In total we collect 5363 preferences in 69 HITs.

The following are screenshots from one actual Amazon Mechanical Turk (AMT) experiment. These screenshots show the introduction (debriefing) of the experiment, an example dialogue and one dialogue context with two candidate responses, which human evaluators were asked to choose between. The experiment was carried out using psiturk, which can be downloaded from www.psiturk.org.

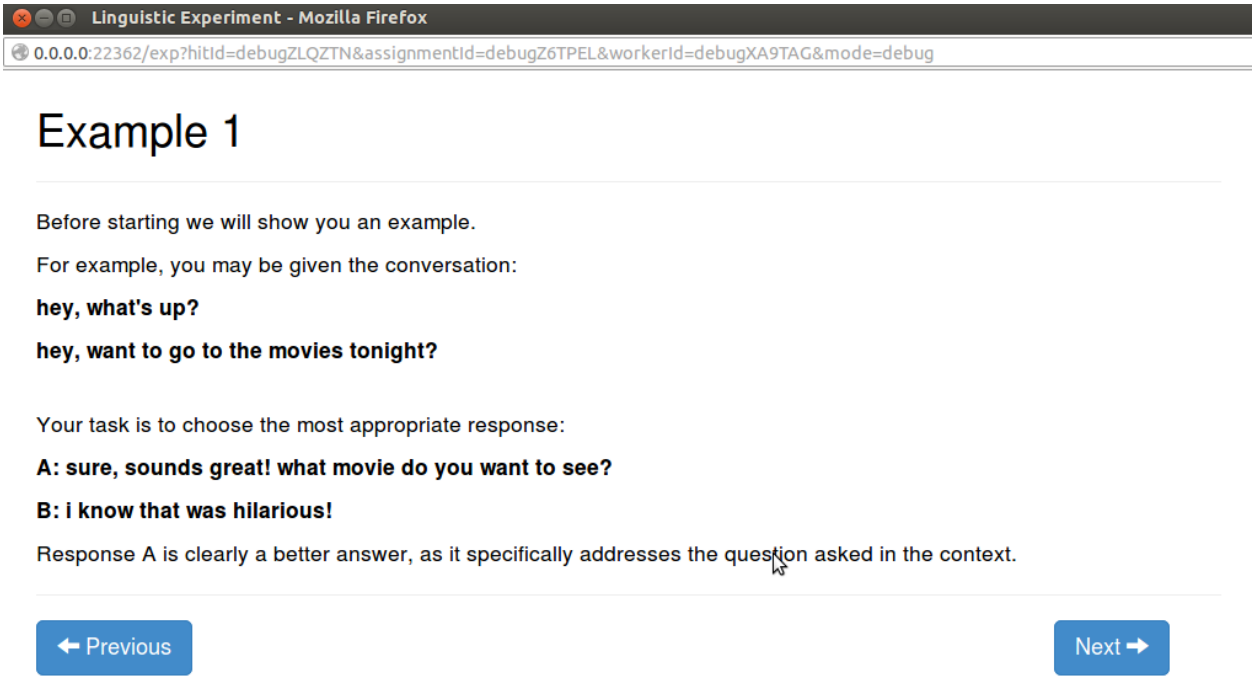

Figure 21: Screenshot of the introduction (debriefing) of the experiment.

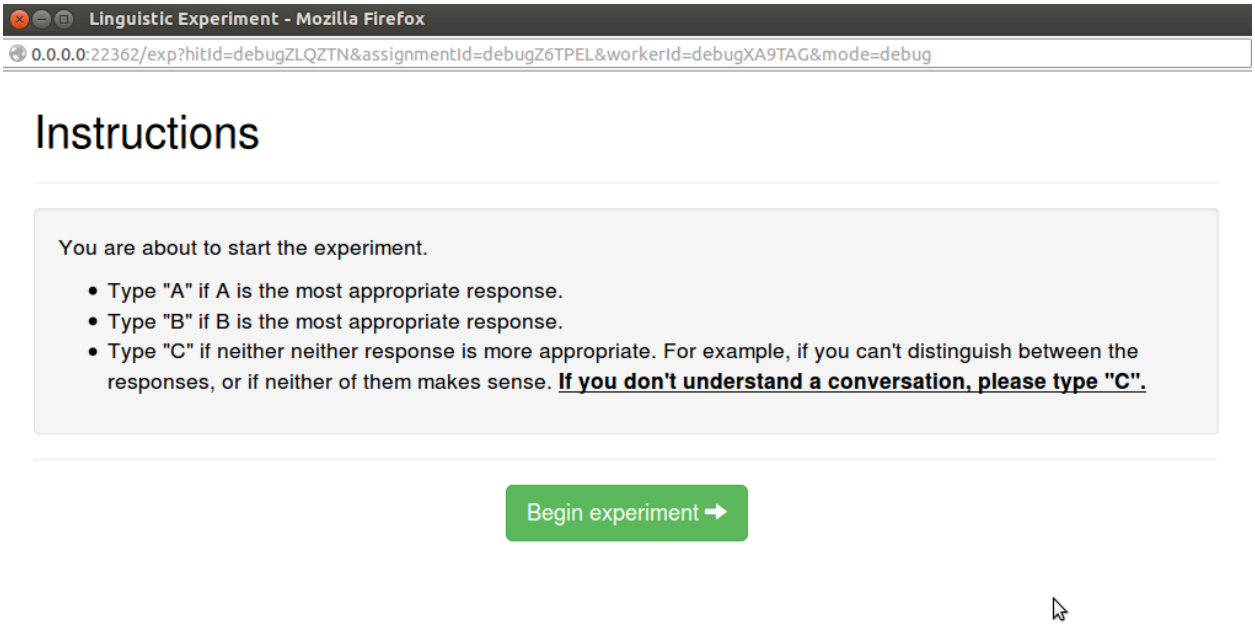

Figure 22: Screenshot of the introductory dialogue example.

III Appendix: Human Evaluation in the Research Lab (Ubuntu)

This appendix describes the human evaluation on the Ubuntu task described in the first part of the thesis. The text for this appendix has been adapted from the appendix in Serban et al. (2017b) written by the author of this thesis. The appendix text was never published in any journal, conference or workshop proceedings.

All human evaluators either studied or worked in an English speaking environment, and indicated that they had some experience using a Linux operating system. To ensure a high quality of the ground truth responses, human evaluators were only asked to evaluate responses, where the ground truth contained at least one technical entity. Before starting, the evaluators were shown one short annotated example with a brief explanation of how to give annotations. In particular, the evaluators were instructed to use the reference in Figure 23.

Fluency	**Relevancy**
0 : Incomprehensible	**0** : None
1 : Disfluent English	**1** : Little relevance
2 : Non-Native English	**2** : Much relevance
3 : Good English	**3** : Most relevant
4 : Flawless English	**4** : All relevant

Figure 23: Fluency and relevancy reference table presented to human evaluators.

The 5 evaluators gave 1069 ratings in total. Table 20 shows the scores by category.

Table 20: Ubuntu human fluency and relevancy scores by rating category

	Fluency (0-4)					**Relevancy (0-4)**				
Rating Level / **Model**	0	1	2	3	4	0	1	2	3	4
HRED	3	11	21	50	49	68	22	19	19	4
HRED + Act.-Ent.	3	17	19	37	57	69	39	18	6	2
MrRNN Noun	1	2	8	52	71	51	45	24	10	4
MrRNN Act.-Ent	0	2	6	52	74	27	53	39	14	1

IV Appendix: Milabot Response Models

This appendix describes the response models for the Milabot ensemble system. The text for this appendix has been adapted from the appendix in the pre-print article Serban et al. (2017c) written by the author of this thesis. The appendix text was never published in any journal, conference or workshop proceedings.

There are 22 response models in the system, including retrieval-based neural networks, generation-based neural networks, knowledge base question answering systems and template-based systems. Examples of candidate model responses are shown in Table 11.

Template-based Models

We first describe the template-based response models in the system.

Alicebot: *Alicebot* uses a set of AIML (artificial intelligence markup language) templates to produce a response given the dialogue history and user utterance (Wallace, 2009; Shawar and Atwell, 2007). We use the freely available Alice kernel available at `www.alicebot.org`. By default all templates generate non-priority responses, so we configure templates related to the socialbot's name, age and location to output priority responses. We modify a few templates further to make them consistent with the challenge (e.g. to avoid obscene language and to encourage the user to discuss certain topics, such as news, politics and movies). The majority of templates remain unchanged.

The *Alicebot* model also outputs a scalar confidence score. Since the AIML templates repeat the user's input utterance, they are not always correct sentences. Therefore, we use a string-based rules to determine if the response constitutes a correct sentence. If the response is correct sentence, it returns a high confidence and otherwise it returns a low confidence score. This process is illustrated in Algorithm 1.

Algorithm 1: Alicebot

```
input: dialogue history
response ← apply AIML templates to dialogue history
if response is correct sentence then
    if response is given priority then
        confidence ← 1.0
    else
        confidence ← 0.5
else
    confidence ← 0.0
output: response, priority, confidence
```

Elizabot Similar to *Alicebot*, the *Elizabot* model performs string matching to select an answer from a set of templates. The model is based on the famous Eliza system, designed to mimic a Rogerian psychotherapist (Weizenbaum, 1966).[50] Therefore, in contrast with *Alicebot*, most of *Elizabot*'s responses are personal questions which are meant to engage the user to continue the conversation.

[50]We use the implementation available at: https://gist.github.com/bebraw/273706.

Here are two example templates:

1. *"I am (.*)" → "Did you come to me because you are ..."*
2. *"What (.*)" → "Why do you ask?"*

The ellipses mark the parts of the response sentence which will be replaced with text from the user's utterance. The model detects the appropriate template and selects the corresponding response (if there are multiple templates, then a template is selected at random). The model then runs the template response through a set of *reflections* to better format the string for a response (e.g. *"I'd" → "you would"*, *"your" → "my"*).

Algorithm 2: Initiatorbot

```
input: dialogue history
if Initiatorbot was triggered in one of last two turns then
    return ""
else if user did not give a greeting then
    return a non-priority response with a random initiator phrase
else
    return a priority response with a random initiator phrase
```

Initiatorbot The *Initiatorbot* model acts as a *conversation starter*: it asks the user an open-ended question to get the conversation started and increase the engagement of the user. We wrote 40 question phrases for the *Initiatorbot*. Examples of phrases include *"What did you do today?"*, *"Do you have pets?"* and *"What kind of news stories interest you the most?"*. As a special case, the model can also start the conversation by stating an interesting fact. In this case, the initiator phrase is *"Did you know that <fact>?"*, where *fact* is replaced by a statement. The set of facts is the same as used by the *BoWFactGenerator* model, described later.

Before returning a response, *Initiatorbot* first checks that it hasn't already been triggered in the last two turns of the conversation. If the user gives a greeting (e.g. *"hi"*), then *Initiatorbot* will return a response with priority. This is important because we observed that greetings often indicate the beginning of a conversation, where the user does not have a particular topic they would like to talk about. By asking a question, the system takes the *initiative* (i.e. control of the dialogue). The procedure is detailed in Algorithm 2.

Storybot The *Storybot* model outputs a short fiction story at the request of the user. We implemented this model as we observed that many users were asking the socialbot to tell stories.[51] *Storybot* determines if the user requested a story by checking if there was both a request

[51]Requests for telling stories is possibly a side-effect of user's interacting with bots from other teams, which often emphasized *non-conversational activities*, such as telling stories and playing quizzes and word games.

word (e.g. *say*, *tell*.) and story-type word in the utterance (e.g. *story*, *tale*). The response states the story's title and author followed by the story body. For example, one set of responses from this model follows the pattern *"Alright, let me tell you the story* <*story_title*> <*story_body*> *by* <*story_author*>*"* where <story_title> is the title of the story, <story_body> is the main text and <story_author> is the name of the story's author. The stories were scraped from the website: www.english-for-students.com.

An example story is:

> **** The Ant and The Grasshopper ****
> The ants worked hard in summer. They sorted food for winter.
> At that time, a grasshopper remained idle. When winter came, the ants had enough to eat.
> But, the grasshopper had nothing to eat. He had to starve.
> He went to the ants and begged for foods. The ants asked in return, "What did you do in summer?"
> He replied, "I idled away my time during summer".
> The ant replied, "Then you must starve in winter." MORAL: Never be idle.

The *Storybot* is the only component in the system performing a *non-conversational activity*. It is triggered only when a user specifically asks for a story, and in that case its response is a priority response. Otherwise, the *Storybot* response model is never triggered. Further, the rest of the system will not encourage the user to request stories.

Knowledge Base-based Question Answering

Evibot The *Evibot* response model forwards the user's utterance to Amazon's question-answering web-service *Evi*: www.evi.com. *Evi* was designed primarily to handle factual questions. Therefore, *Evibot* returns a priority response for direct questions, defined as user utterances containing a wh-word (e.g. *"who"*, *"what"*), and otherwise returns a non-priority or, possibly, an empty response. If the query is a direct question and contains non-stop words, *Evibot* will follow a three step procedure to generate its response. First, *Evibot* forwards a query to www.evi.com containing the whole user utterance, and returns the resulting answer if its valid. If that fails, *Evibot* applies NLTK's named entity processor (Bird et al., 2009) to the query to find subqueries with named entities. For each subphrase that contains a named entity, *Evibot* forwards queries to www.evi.com, and returns the result upon a valid response. Finally, if the previous two steps fail, *Evibot* forwards queries for every subquery without named entities, and returns either a valid response or an empty response. The procedure is detailed in Algorithm 3.

Algorithm 3: Evibot

```
input: dialogue history
query ← last user utterance
has-wh-words ← true if utterance contains a wh-word, otherwise false
has-only-stop-words ← true if utterance only has stop words, otherwise false
if has-only-stop-words and not has-wh-words then
    return ""
evi-response ← send query to www.evi.com
priority ← true if has-wh-words and evi-response is valid, otherwise false
if evi-response is valid then
    return evi-response, priority
else if has-wh-words then
    priority ← has-wh-words
    subentities ← entities extracted from query using NLTK's named entity processor
    subphrases ← list of subphrases with entities
    for subphrase in subphrases do
        evi-response ← send subphrase to www.evi.com
        if evi-response is valid then
            return evi-response, priority
    subphrases ← list of all subphrases
    for subphrase in subphrases do
        evi-response ← send subphrase to www.evi.com
        if evi-response is valid then
            return evi-response, priority
else
    return ""
```

BoWMovies The *BoWMovies* model is a template-based response model, which handles questions in the movie domain. The model has a list of entity names and tags (e.g. *movie plot* and *release year*). The model searches the user's utterance for known entities and tags. Entities are identified by string matching. This is done in a cascading order, by giving first preference to movie title matches, then actor name matches, and finally director name matches. Tags are also identified by string matching. However, if exact string matching fails for tags, then identification is performed by word embedding similarity. If both an entity and a tag are present, the agent will dispatch an API call to one of several data sources to retrieve the data item for the selected query

type. The agent is limited by the data available in the APIs to which it has access. The model's responses follow predefined templates.

Movie titles, actor names, and director names are extracted from the Internet Movie Database (IMDB). Movie descriptions are taken from Google Knowledge GraphâĂŹs API. Other movie title queries are directed to the Open Movie Database (OMDB).[52] For actor and director queries, the Wikiedata API is used. First, a search for actor and director names is done on a Wikidata JSON dump.

As described earlier, the model uses word embeddings to match tags. These word embeddings are trained using Word2Vec on movie plot summaries and actor biographies extracted from the IMDB database (Mikolov et al., 2013b).

[52] See www.omdbapi.com. This should not be confused with IMDB.

Algorithm 4: BoWMovies - ComputeResponse

input: dialogue history
entity ← entity contained both in last user utterance and list of movie titles, actors or directors
if *no entity* **then**
    entity ← entity contained in previous user utterances and movie titles, actors or directors
if *no entity* **then**
    return ""
if *entity is a movie title* **then**
    response ← ComputeEntityResponse(entity, movie title)
else if *entity is an actor name* **then**
    response ← ComputeEntityResponse(entity, actor name)
else if *entity is an director name* **then**
    response ← ComputeEntityResponse(entity, director name)
return response

Algorithm 5: BoWMovies - ComputeEntityResponse

input: entity and entity type
tag ← string matching tag, where tag is valid for entity type (movie title, actor name, director name)
if *no tag* **then**
    tag ← word embedding matching tag, where tag is a single word and valid for the entity type (movie title, actor name, director name)
if *no tag* **then**
    tag ← word embedding matching tag, where tag is multiple words and valid for the entity type (movie title, actor name, director name)
if *no tag* **then**
    return ""
api-response ← call external API with query (entity, tag).
response ← template with api-response inserted
return response

Retrieval-based Neural Networks

VHRED models: The system contains several VHRED models, sequence-to-sequence models with Gaussian latent variables trained as variational auto-encoders (Serban et al., 2017d; Kingma and Welling, 2014; Rezende et al., 2014). The models are trained using the same procedure as Serban et al. (2017d). The trained VHRED models generate candidate responses as follows. First, a set of K model responses are retrieved from a dataset using cosine similarity between the current dialogue history and the dialogue history in the dataset based on bag-of-words TF-IDF Glove word embeddings (Pennington et al., 2014).[53] An approximation of the log-likelihood for each of the 20 responses is computed by VHRED, and the response with the highest log-likelihood is returned. The system has 4 VHRED models based on datasets scraped from Reddit, one VHRED model based on news articles and one VHRED model based on movie subtitles:

- *VHREDRedditPolitics* trained on https://www.reddit.com/r/politics and extracting responses from all Reddit datasets with $K = 10$,
- *VHREDRedditNews* trained on Reddit https://www.reddit.com/r/news and extracting responses from all Reddit datasets with $K = 20$,
- *VHREDRedditSports* trained on Reddit https://www.reddit.com/r/sports and extracting responses from all Reddit datasets with $K = 20$,
- *VHREDRedditMovies* trained on Reddit https://www.reddit.com/r/movies and extracting responses from all Reddit datasets with $K = 20$,
- *VHREDWashingtonPost*[54] trained on Reddit https://www.reddit.com/r/politics and extracting responses from user comments to WashingtonPost news articles, and
- *VHREDSubtitles*[55] using the movie subtitles dataset SubTle (Ameixa et al., 2014) with $K = 10$.

In particular, *VHREDRedditPolitics* and *VHREDWashingtonPost* use a different retrieval procedure. These two models use a logistic regression model to score the responses instead of the approximate log-likelihood. The logistic regression model is trained on a set of 7500 Reddit threads and candidate responses annotated by Amazon Mechanical Turk workers on a Likert-type scale $1 - 5$. The candidate responses are selected from other Reddit threads according to cosine similarity w.r.t. Glove word embeddings. The label collection and training procedure for the logistic regression model are similar to the procedures described in Section 4.7. For each response,

[53] We use the Glove embeddings trained on Wikipedia 2014 + Gigaword 5: https://nlp.stanford.edu/projects/glove/.

[54] For *VHREDWashingtonPost*, the K responses are extracted based on the cosine similarity between the current dialogue and the news article keywords. K varies depending on the number of user comments within a set of news articles above a certain cosine similarity threshold.

[55] For *VHREDSubtitles*, cosine similarity is computed based on one-hot vectors for each word.

the logistic regression model takes as input the VHRED log-likelihood score, as well as several other input features, and outputs a scalar-valued score. Even though the logistic regression model did improve the appropriateness of responses selected for Reddit threads, *VHREDRedditPolitics* is used extremely rarely in the final system (see Section 4.12.1). This suggests that training a model to rerank responses based on labelled Reddit threads and responses cannot help improve performance.

SkipThought Vector Models: The system contains a SkipThought Vector model (Kiros et al., 2015) trained on the BookCorpus dataset (Zhu et al., 2015) and on the SemEval 2014 Task 1 (Marelli et al., 2014). The model was trained using the same procedure as Kiros et al. (2015) and is called *SkipThoughtBooks*.

SkipThoughtBooks ensures that the system complies with the Amazon Alexa Prize competition rules. One rule, introduced early in the competition, is that socialbots were not supposed to state their own opinions related to political or religious topics. If a user wishes to discuss such topics, the socialbots should proceed by asking questions or stating facts. *SkipThoughtBooks* also handles idiosyncratic issues particular to the Alexa platform. For example, many users did not understand the purpose of a socialbot and asked our socialbot to play music. In this case, the system should instruct the user to exit the *socialbot application* and then play music.

SkipThoughtBooks follows a two-step procedure to generate its response. The first step compares the user's last utterance to a set of trigger phrases. If a match is found, the model returns a corresponding priority response.[56] For example, if the user says *"What do you think about Donald trump?"*, the model will return a priority response, such as *"Sometimes, truth is stranger than fiction."*. A match is found if: 1) the SkipThought Vector model's semantic relatedness score between the user's last utterance and a trigger phrase is above a predefined threshold, and 2) the user's last utterance contains keywords relevant to the trigger phrase.[57] In total, there are 315 trigger phrases (most are paraphrases of each other) and 35 response sets.

If the model did not find a match in the first step, it proceeds to the second step. In this step, the model selects its response from among all Reddit dataset responses. As before, a set of K model responses are retrieved using cosine similarity. The model then returns the response with the highest semantic relatedness score.

Dual Encoder Models: The system contains two Dual Encoder retrieval models, as proposed by Lowe et al. (2015b) and Lowe et al. (2017b), called *DualEncoderRedditPolitics* and *DualEncoderRedditNews*. Both models are composed of two sequence encoders ENC_Q and ENC_R with a single LSTM recurrent layer used to encode the dialogue history and a candidate response. The score for a candidate response is computed by a bilinear mapping of the dialogue history em-

[56] Trigger phrases may have multiple responses. In this case, a response is selected at random.

[57] Some trigger phrases do not have keywords. In this case, matching is based only on semantic relatedness.

bedding and the candidate response embedding. The models are trained using the method proposed by Lowe et al. (2015b). In principle, it is also possible to use early stopping based on a separate model trained on a domain similar to our target domain (Lowe et al., 2016). The response with the highest score from a set of $K = 50$ candidate responses are retrieved using TF-IDF cosine similarity based on Glove word embeddings. The model *DualEncoderRedditPolitics* is trained on the Reddit https://www.reddit.com/r/politics dataset and extracts responses from all Reddit datasets. The model *DualEncoderRedditNews* is trained on the Reddit https://www.reddit.com/r/news dataset and extracts responses from all Reddit datasets.

Bag-of-words Retrieval Models: The system contains three bag-of-words retrieval models based on TF-IDF Glove word embeddings (Pennington et al., 2014) and Word2Vec embeddings (Mikolov et al., 2013b).[58] Similar to the VHRED models, these models retrieve the response with the highest cosine similarity. The *BoWWashingtonPost* model retrieves user comments from WashingtonPost news articles using Glove word embeddings. The model *BoWTrump* retrieves responses from a set of Twitter tweets scraped from Donald Trump's profile: https://twitter.com/realDonaldTrump. This model also uses Glove word embeddings and it only returns a response when at least one relevant keyword or phrase is found in the user's utterance (e.g. when the word *"Trump"* is mentioned by the user). The list of trigger keywords and phrases include: *'donald'*, *'trump'*, *'potus'*, *'president of the united states'*, *'president of the us'*, *'hillary'*, *'clinton'*, *'barack'*, and *'obama'*. The model *BoWFactGenerator* retrieves responses from a set of about 2500 *interesting* and *fun* facts, including facts about animals, geography and history. The model uses Word2Vec word embeddings. The model *BoWGameofThrones* retrieves responses from a set of quotes scraped from https://twitter.com/ThroneQuotes using Glove word embeddings. Tweets from this source were manually inspected and cleaned to remove any tweets that were not quotes from the series. As in the *BoWTrump* model, we use a list of trigger phrases to determine if the model's output is relevant to the user's utterance. We populate this list with around 80 popular character names, place names and family names, which are large unique to the domain. We also added a few aliases to try and account for alternative speech transcriptions of these named entities. Some phrases include: *'ned stark'*, *'jon snow'*, *'john snow'*, *'samwell tarly'*, *"hodor"*, *"dothraki"* and so on. [59]

[58] We use the pre-trained Word2Vec embeddings: https://code.google.com/archive/p/word2vec/.

[59] This model was implemented after the competition ended, but is included here for completeness.

Retrieval-based Logistic Regression

BoWEscapePlan: The system contains a response model, called *BoWEscapePlan*, which returns a response from a set of 35 topic-independent, generic pre-defined responses, such as *"Could you repeat that again"*, *"I don't know"* and *"Was that a question?"*. Its main purpose is to maintain user engagement and keep the conversation going, when other models are unable to provide meaningful responses. This model uses a logistic regression classifier to select its response based on a set of higher-level features.

To train the logistic regression classifier, we annotated $12,000$ user utterances and candidate response pairs for appropriateness on a Likert-type scale $1-5$. The user utterances were extracted from interactions between Alexa users and a preliminary version of the system. The candidate responses were sampled at random from *BoWEscapePlan*'s response list. The label collection and training procedure for the logistic regression model are similar to the procedures described in section 4.7. The logistic regression model is trained with log-likelihood on a training set, with early-stopping on a development set, and evaluated on the testing set. However, the trained model's performance was poor. It obtained a Pearson correlation coefficient of 0.05 and a Spearman's rank correlation coefficient of 0.07. This indicates that the logistic regression model is only slightly better at selecting a topic-independent, generic response compared to selecting a response at uniform random. Future work should investigate collecting more labelled data and pre-training the logistic regression model.

Search Engine-based Neural Networks

The system contains a deep classifier model, called *LSTMClassifierMSMarco*, which chooses its response from a set of search engine results. The system searches the web with the last user utterance as query, and retrieves the first 10 search snippets. The retrieved snippets are preprocessed by stripping trailing words, removing unnecessary punctuation and truncating to the last full sentence. The model uses a bidirectional LSTM to separately map the last dialogue utterance and the snippet to their own embedding vectors. The resulting two representations are concatenated and passed through an neural network to predict a scalar-value between $0-1$ indicating how appropriate the snippet is as a response to the utterance.

The model is trained as a binary classification model on the Microsoft Marco dataset with cross-entropy to predict the relevancy of a snippet given a user query (Nguyen et al., 2016). Given a search query and a search snippet, the model must output one when the search snippet is relevant and otherwise zero. Search queries and ground truth search snippets are taken as positive samples, while other search snippets are selected at random as negative samples. On this task, the model is able to reach a prediction accuracy of 72.96% w.r.t. the Microsoft Marco development set.

The system is able to use search APIs from various search engines including Google and Bing. In the current model, we choose Google as the search engine, since qualitative inspection showed that this retrieved the most appropriate responses.

Generation-based Neural Networks

The system contains a generative recurrent neural network language model, called *GRUQuestion-Generator*, which can generate follow-up questions word-by-word, conditioned on the dialogue history. The input to the model consists of three components: a one-hot vector of the current word, a binary question label and a binary speaker label. The model contains two GRU layers (Cho et al., 2014) and softmax output layer. The model is trained on Reddit Politics and Reddit News conversations, wherein posts were labelled as questions by detecting question marks. We use the optimizer Adam (Kingma and Ba, 2015), and perform early stopping by checking the perplexity on the validation set For generation, we first condition the model on a short question template (e.g. *"How about"*, *"What about"*, *"How do you think of"*, *"What is your opinion of"*), and then generate the rest of the question by sampling from the model with the question label clamped to one. The generation procedure stops once a question mark is detected. Further, the length of the question is controlled by tuning the temperature of the softmax layer. Due to speed requirements, only two candidate responses are generated and the best one w.r.t. log-likelihood of the first 10 words is returned.

V Appendix: Milabot Crowdsourced Data Collection

This appendix describes the crowdsourcing data collection conducted for the Milabot ensemble system. The text for this appendix has been ada

https://github.com/julianser/aiducate/pull/554pted from the pre-print article Serban et al. (2017c) written by the author of this thesis. The text was never published in any journal, conference or workshop proceedings.

We use Amazon Mechanical Turk (AMT) to collect data for training the scoring model. We follow a setup similar to Liu et al. (2016). We show human evaluators a dialogue along with 4 candidate responses, and ask them to score how appropriate each candidate response is on a 1-5 Likert-type scale. The score 1 indicates that the response is inappropriate or does not make sense, 3 indicates that the response is acceptable, and 5 indicates that the response is excellent and highly appropriate.

Our setup only asks human evaluators to rate the overall appropriateness of the candidate responses. In principle, we could choose to evaluate other aspects of the candidate responses. For example, we could evaluate fluency. However, fluency ratings would not be very useful since most of our models retrieve their responses from existing corpora, which contain mainly fluent and grammatically correct responses. As another example, we could evaluate topical relevancy. However, we choose not to evaluate such criteria since it is known to be difficult to reach high inter-annotator agreement on them (Liu et al., 2016). In fact, it is well known that even asking for a single overall rating tends to produce only a fair agreement between human evaluators (Charras et al., 2016); disagreement between annotators tends to arise either when the dialogue context is short and ambiguous, or when the candidate response is only partially relevant and acceptable.

The dialogues are extracted from interactions between Alexa users and preliminary versions of our system. Only dialogues where the system does not have a priority response were extracted (when there is a priority response, the dialogue manager must always return the priority response). About $3/4$ of these dialogues were sampled at random, and the remaining $1/4$ dialogues were sampled at random excluding identical dialogues.[60] For each dialogue, the corresponding candidate responses are created by generating candidate responses from the response models.

We preprocess the dialogues and candidate responses by masking out profanities and swear words with stars (e.g. we map *"fuck"* to *"****"*).[61] Furthermore, we anonymize the dialogues and candidate responses by replacing first names with randomly selected gender-neutral names (for

[60] Sampling at random is advantageous for our goal, because it ensures that candidate responses to frequent user statements and questions tend to be annotated by more turkers. This increases the average annotation accuracy for such utterances, which in turn increases the scoring model's accuracy for such utterances.

[61] The masking is not perfect. Therefore, we also instruct turkers that the task may contain profane and obscene language. Further, it should also be noted that Amazon Mechanical Turk only employs adults.

example, *"Hi John"* could be mapped to *"Hello Casey"*). Finally, the dialogues are truncated to the last 4 utterances and last 500 words. This reduces the cognitive load of the annotators. Examples from the crowdsourcing task are shown in Figure 24, Figure 25 and Figure 26. The dialogue example shown in Figure 26 is a fictitious example.

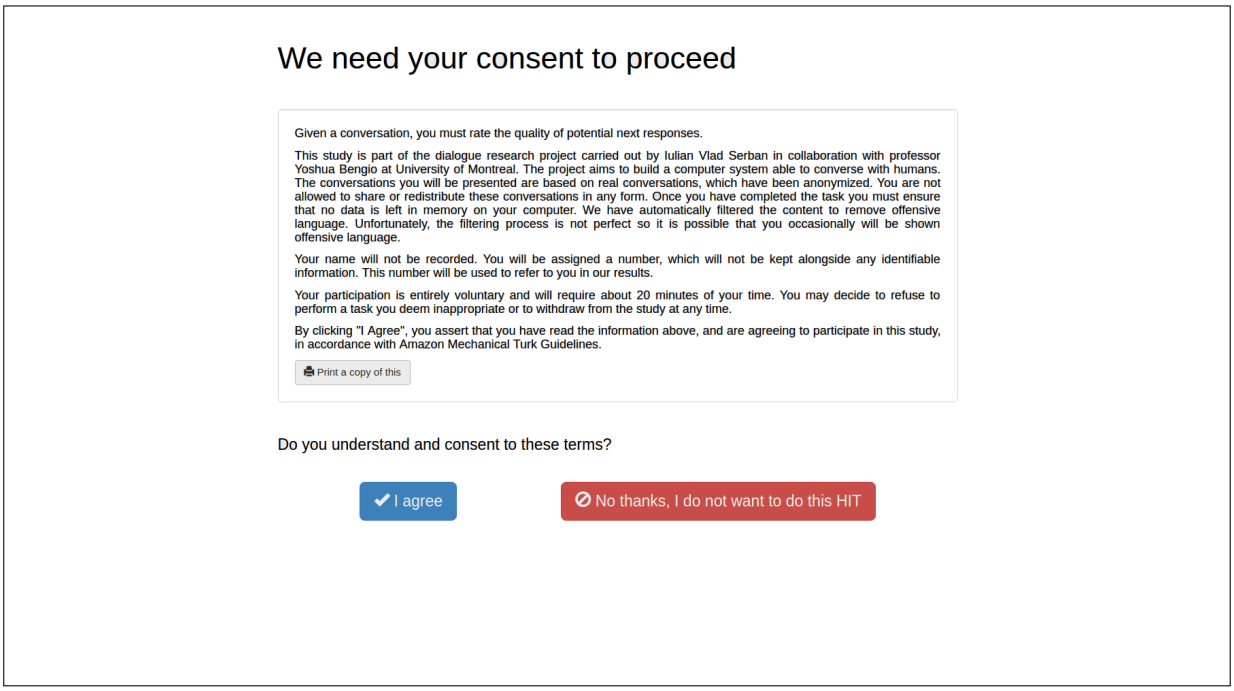

Figure 24: Consent screen for Amazon Mechanical Turk human intelligence tasks (HITs).

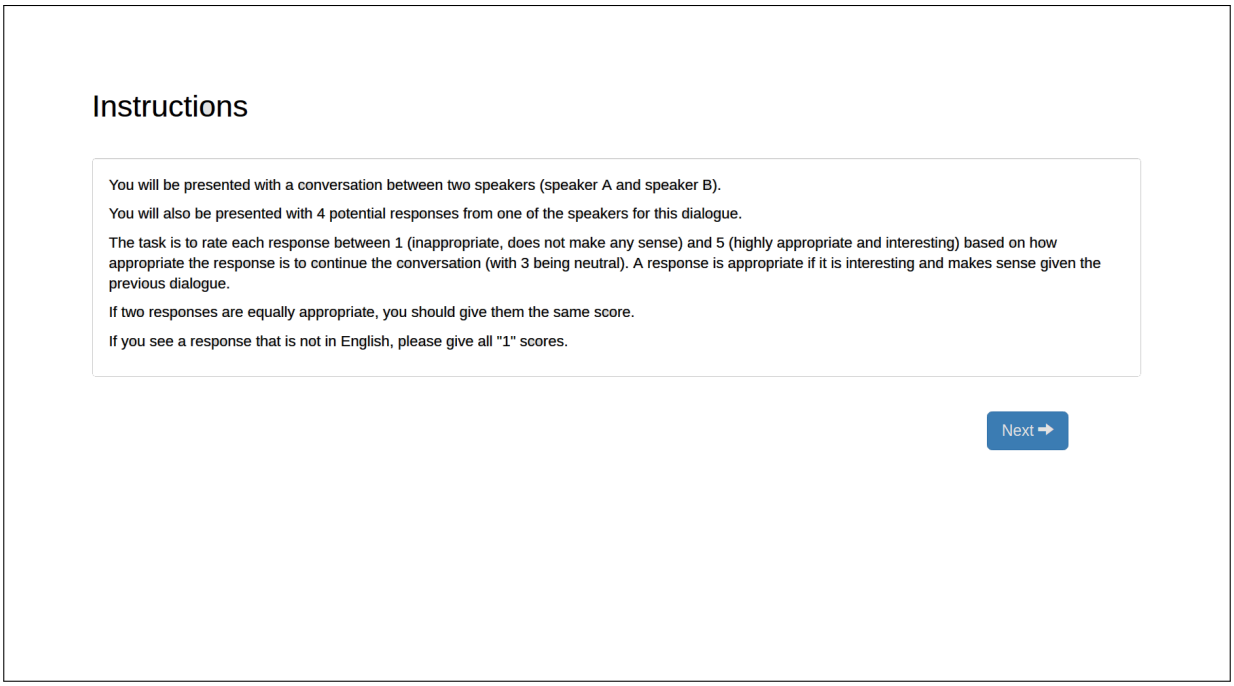

Figure 25: Instructions screen for Amazon Mechanical Turk human intelligence tasks (HITs).

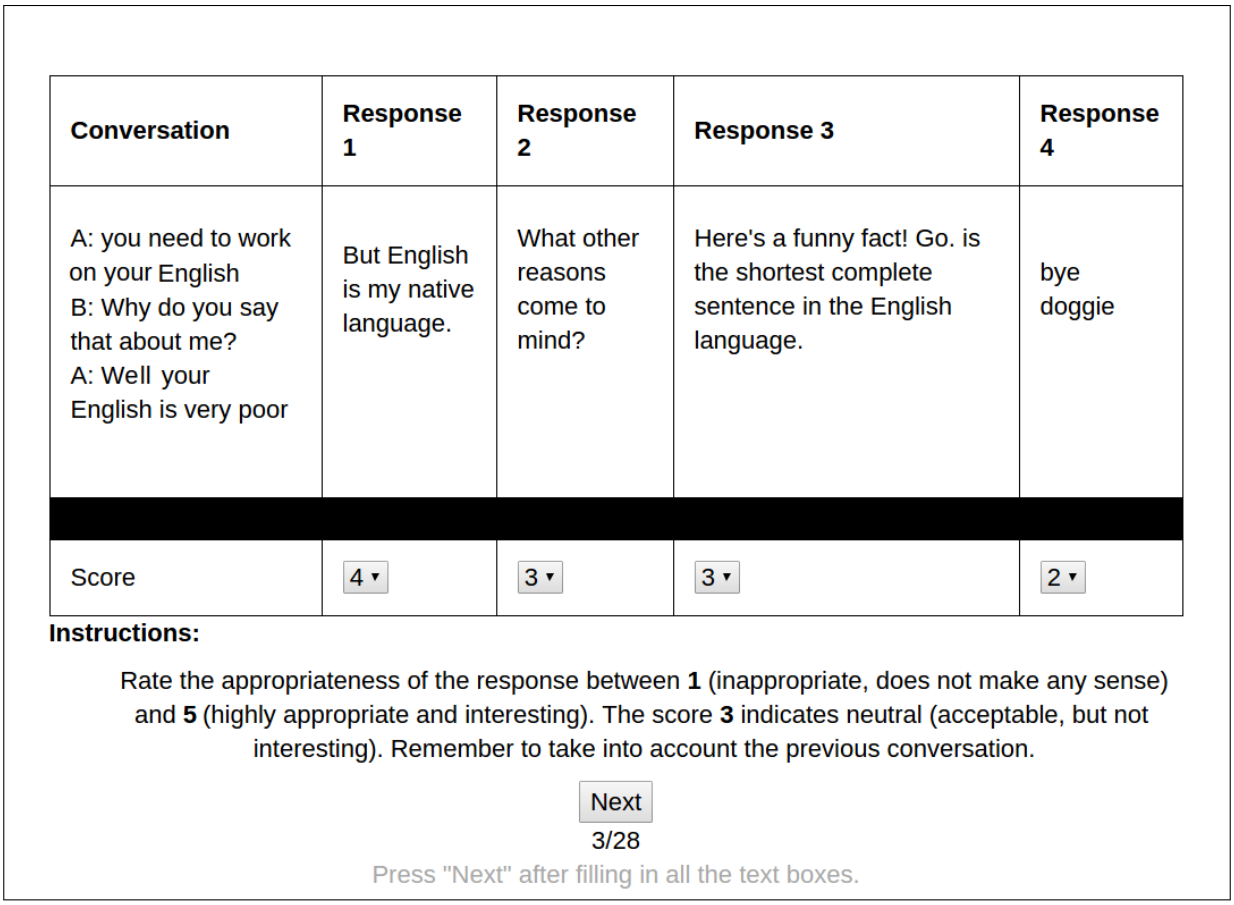

Figure 26: Annotation screen for Amazon Mechanical Turk human intelligence tasks (HITs). The dialogue text is a fictitious example.

We inspected the annotations manually. We observed that annotators tended to frequently overrate topic-independent, generic responses. Such responses may be considered acceptable for a single turn in a conversation, but are likely to be detrimental when repeated over and over again. In particular, annotators tended to overrate responses generated by the response models *Alicebot*, *Elizabot*, *VHREDSubtitles* and *BoWEscapePlan*. Responses generated by these models are often acceptable or good, but the majority of them are topic-independent, generic sentences. Therefore, for these response models, we mapped all labels 5 (*"excellent"*) to 4 (*"good"*). Furthermore, for responses consisting of only stop-words, we decreased the labels by one level (e.g. 4 is mapped to 3). Finally, the *BoWMovies* response model suffered from a bug during the label collection period. Therefore, we decreased all labels given to *BoWMovies* responses to be at most 2 (*"poor"*).

In total, we collected $199,678$ labels. We split this into training (train), development (dev) and testing (test) datasets consisting of respectively 137,549, 23,298 and 38,831 labels each.