Dialogue & Discourse 8(1) (2017) 31–65 doi: 10.5087/dad.2017.102

Training End-to-End Dialogue Systems with the Ubuntu Dialogue Corpus

Ryan Lowe RYAN.LOWE@CS.MCGILL.CA
School of Computer Science, McGill University

Nissan Pow NISSAN.POW@MAIL.MCGILL.COM
School of Computer Science, McGill University

Iulian Vlad Serban JULIANSERBAN@GMAIL.COM
DIRO, Université de Montréal

Laurent Charlin LCHARLIN@GMAIL.COM
School of Computer Science, McGill University

Chia-Wei Liu CHIA-WEI.LIU@MAIL.MCGILL.CA
School of Computer Science, McGill University

Joelle Pineau JPINEAU@CS.MCGILL.CA
School of Computer Science, McGill University

Editor: Amanda Stent

Submitted 04/2016; Accepted 12/2016; Published online 01/2017

Abstract

In this paper, we analyze neural network-based dialogue systems trained in an end-to-end manner using an updated version of the recent Ubuntu Dialogue Corpus, a dataset containing almost 1 million multi-turn dialogues, with a total of over 7 million utterances and 100 million words[1]. This dataset is interesting because of its size, long context lengths, and technical nature; thus, it can be used to train large models directly from data with minimal feature engineering. We provide baselines in two different environments: one where models are trained to select the correct next response from a list of candidate responses, and one where models are trained to maximize the log-likelihood of a generated utterance conditioned on the context of the conversation. These are both evaluated on a recall task that we call next utterance classification (NUC), and using vector-based metrics that capture the topicality of the responses. We observe that current end-to-end models are

1. This work is an extension of a paper appearing in SIGDIAL (Lowe et al., 2015). This paper further includes results on generative dialogue models, more extensive evaluation of the retrieval models using vector-based generative metrics, and a qualitative examination of responses from the generative models and classification errors made by the Dual Encoder model. Experiments are performed on a new version of the corpus, the Ubuntu Dialogue Corpus v2, which is publicly available: `https://github.com/rkadlec/ubuntu-ranking-dataset-creator`. The early dataset has been updated to add features and fix bugs, which are detailed in Section 3.

unable to completely solve these tasks; thus, we provide a qualitative error analysis to determine the primary causes of error for end-to-end models evaluated on NUC, and examine sample utterances from the generative models. As a result of this analysis, we suggest some promising directions for future research on the Ubuntu Dialogue Corpus, which can also be applied to end-to-end dialogue systems in general.

1. Introduction

Deriving statistical models that can naturally and coherently converse with humans is one of the cornerstone problems of artificial intelligence. Until recently, such models required significant hand-engineering of features, and thus could only generate a limited number of responses and be deployed in constrained situations. Recent advances in neural network-based language models have begun to make feasible the idea of learning an entire dialogue model directly from conversational data, with humans only specifying the model hyper-parameters. However, significant work needs to be done before these models can be implemented in practice with high confidence.

In this paper we consider the problem of building dialogue agents in an *end-to-end* manner. We define end-to-end systems, contrary to *modular* systems, as those that are trained directly from conversational data to optimize a single objective function (see Section 1.1). We use the recently released Ubuntu Dialogue Corpus, which consists of almost one million two-person (dyadic) conversations extracted from the Ubuntu chat logs, which provide technical support for various Ubuntu-related problems. Dialogues in the corpus are *multi-turn* and *unstructured*, as there is no *a priori* logical representation for the information exchanged during the conversation. This is in contrast to recent systems which focus on structured dialogue tasks, using slot-filling representations (Williams et al., 2013; Henderson et al., 2014a; Singh et al., 2002).

The creation of such a large, unstructured dialogue dataset was motivated by observations of progress in various sub-fields of AI. In particular, it has been argued that this progress can be attributed to three major factors: 1) the public distribution of very large rich datasets (Deng et al., 2009), 2) the availability of substantial computing power, and 3) the development of new training methods for neural architectures, in particular leveraging unlabeled data.

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

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

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

1.1 Motivation for End-to-End Dialogue Systems

It is important to define specifically what is meant by an 'end-to-end' dialogue system. We begin with the standard architecture for a dialogue system, which incorporates a Speech Recognizer, Language Interpreter, State Tracker, Response Generator, Natural Language Generator, and Speech Synthesizer. In the case of text-based (written) dialogues, the Speech Recognizer and Speech Synthesizer can be left out. Although some previous literature on dialogue systems identifies only the State Tracker and Response Selection components as belonging inside the dialogue manager (Young, 2000), we adopt a broader view where the Language Interpreter and Natural Language Generator are also part of the dialogue manager.

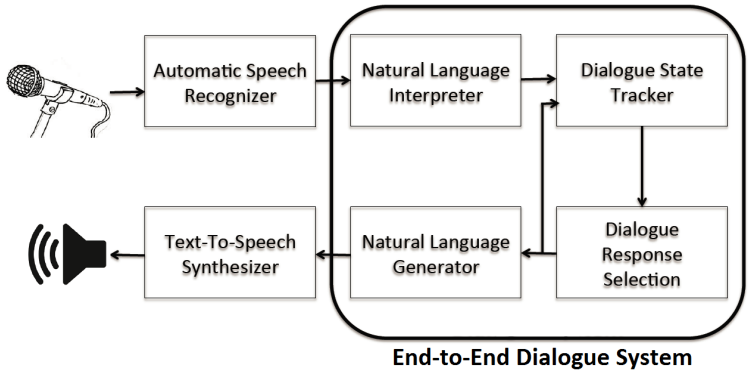

Figure 1: An end-to-end dialogue system replaces the traditional components of a dialogue system with a single statistical model.

When we speak of an 'end-to-end' dialogue system, we mean a single system that can be used to solve each of these four aspects simultaneously (see Figure 1). Typically this is a system that takes as input the history of the conversation and is trained to optimize a single objective, which is a function of the textual output produced by the system and the correct (ground truth) response. This is in contrast to the 'modular' system approach to dialogue systems, where each component of Figure 1 is trained separately, and either takes a more structured input, such as a set of dialogue acts, or is trained to maximize an intermediary objective, such as slot-filling. More formally, we define

a modular dialogue system as a system where two or more elements (sub-components or system parameters) are optimized with respect to two or more different objective functions (e.g. where the State Tracker is trained to minimize the cross-entropy error of predicting the slot-value pairs, and where the Response Generator is trained to maximize the conditional log-likelihood of the correct response given the slot-value pairs). Thus, any machine learning-based dialogue system which is not a modular dialogue system is an end-to-end dialogue system.

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

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

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

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

However, despite these advantages it is not clear whether end-to-end approaches will work for any dialogue domain. For example, in complex negotiation domains where thorough analysis is required and little data is available, end-to-end systems may not be able to learn successful strategies for every combination of preferences and goals. Thus, it is crucial to conduct further research on end-to-end dialogue systems in order to determine the domains in which they are most effective.

1.2 Paper Outline

The paper is structured as follows. In Section 2, we detail some relevant dialogue datasets that are available, and give an overview of existing end-to-end dialogue systems. In Section 3, we describe

the Ubuntu Dialogue Corpus, including corpus statistics, how it was collected, and any changes made in the updated version. We then go on to describe the response ranking models on the Ubuntu Dialogue Corpus in Section 4, and the response generation models in Section 5. We give our results in each of these sections, including our models' performance on next utterance classification and embedding similarity. We also provide a qualitative error analysis of the mistakes made by the classification model in Section 4.6. Finally, we conclude in Section 6, and discuss potential extensions and limitations of the Ubuntu Dialogue Corpus, evaluation metrics, and future directions for end-to-end dialogue systems.

2. Related Work

We briefly review existing dialogue datasets, and some of the more recent learning architectures used for both structured and unstructured dialogues. This is by no means an exhaustive list, but surveys resources most related to our contributions.

2.1 Dialogue Datasets

The Switchboard dataset (Godfrey et al., 1992), and the Dialogue State Tracking Challenge (DSTC) datasets (Williams et al., 2013, 2016) have been used to train and validate dialogue management systems for interactive information retrieval. In the case of the DSTC, the problem is typically formalized as a slot filling task, where agents attempt to predict the goal of a user during the conversation. These datasets have been significant resources for structured dialogues, and have allowed major progress in this field, though they are relatively small compared to datasets currently used for training neural architectures for language-related tasks.

Recently, a few datasets have been used containing unstructured dialogues extracted from Twitter[2]. Ritter et al. (2010) collected 1.3 million conversations; this was extended in Sordoni et al. (2015b) to take advantage of longer contexts by using A-B-A triples. Shang et al. (2015) used data from a similar Chinese website called Weibo[3]. However to our knowledge, these datasets have not been made public, and furthermore, the post-reply format of such microblogging services is perhaps not as representative of natural dialogue between humans as the continuous stream of messages in a chat room. In fact, Ritter et al. (2010) estimate that only 37% of posts on Twitter are 'conversational in nature', and 69% of their collected data contains exchanges of only length 2 (Ritter et al., 2010). We hypothesize that the interaction patterns of chat-room style messaging are more closely correlated to human-human dialogue than micro-blogging websites, or forum-based sites such as Reddit.

Part of the Ubuntu chat logs have previously been aggregated into a dataset, called the Ubuntu Chat Corpus (Uthus and Aha, 2013b). However that resource preserves the multi-participant structure and thus is less amenable to the investigation of more traditional two-party conversations.

Also weakly related to our contribution is the problem of question-answer systems. Several datasets of question-answer pairs are available (Boyd-Graber et al., 2012), however these interactions are much shorter than what we seek to study.

For a comprehensive survey of both available dialogue datasets and prevalent models, see Serban et al. (2015).

2. https://twitter.com/
3. http://www.weibo.com/

2.2 Learning Architectures for End-to-End Dialogue Systems

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

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

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

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

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

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

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

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

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

3. The Ubuntu Dialogue Corpus

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

3.1 Ubuntu Chat Logs

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

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

Despite the nature of the chat room being a constant stream of messages from multiple users, it is through the fairly rigid structure in the messages that we can extract the dialogues between users. Figures 2 and 3 show an example chat room conversation from the #ubuntu channel as well as the extracted dialogues, which illustrates how users usually state the username of the intended message recipient before writing their reply (we refer to all initial questions and replies as 'utterances'). For example, it is clear that users 'Taru' and 'kuja' are engaged in a dialogue, as are users 'Old' and 'bur[n]er', while user '_pm' is asking an initial question, and 'LiveCD' is perhaps elaborating on a previous comment.

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

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

Figure 2: Example chat room conversation from the #ubuntu channel of the Ubuntu Chat Logs (left), with the disentangled conversations for the Ubuntu Dialogue Corpus (right).

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

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

Figure 3: Example of before (left) and after (right) the algorithm adds and concatenates utterances in dialogue extraction. Since `RC` only addresses `dell`, all of his utterances are added, however this is not done for `dell` as he addresses both `RC` and `cucho`.

3.2 Dataset Creation

In order to create the Ubuntu Dialogue Corpus, first a method had to be devised to extract dyadic dialogues from the chat room multi-party conversations. The first step was to separate every message into 4-tuples of *(time, sender, recipient, utterance)*. Given these 4-tuples, it is straightforward

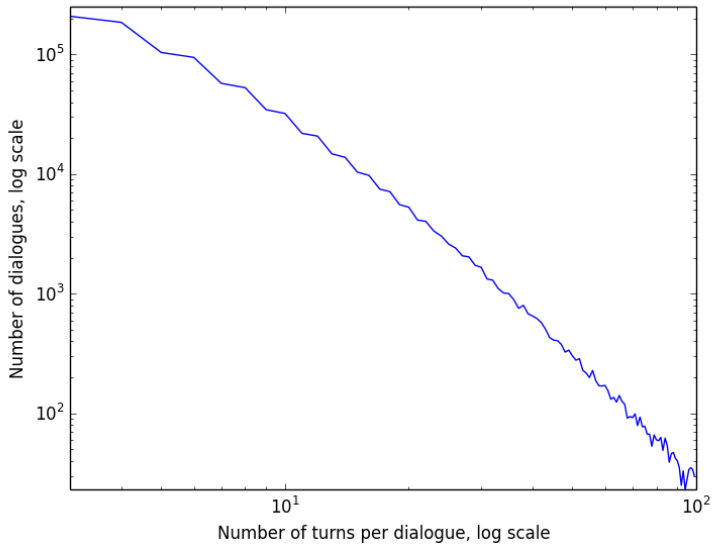

Figure 4: Plot of number of conversations with a given number of turns. Both axes use a log scale.

to group all tuples where there is a matching sender and recipient. Although it is easy to separate the time and the sender from the rest, finding the intended recipient of the message is not always trivial.

3.2.1 RECIPIENT IDENTIFICATION

While in most cases the recipient is the first word of the utterance, it is sometimes located at the end, or not at all in the case of initial questions. Furthermore, some users choose names corresponding to common English words, such as 'the' or 'stop', which could lead to many false positives. In order to solve this issue, we create a dictionary of usernames from the current and previous days, and compare the first word of each utterance to its entries. If a match is found, and the word does not correspond to a very common English word[4], it is assumed that this user was the intended recipient of the message. If no matches are found, it is assumed that the message was an initial question, and the recipient value is left empty.

3.2.2 UTTERANCE CREATION

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

All utterances that do not qualify as a first response or an initial question are discarded; initial questions that do not generate any response are also discarded. We additionally discard conversations longer than five utterances where one user says more than 80% of the utterances, as these are

4. We use the GNU Aspell spell checking dictionary.

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

Table 1: Properties of Ubuntu Dialogue Corpus. Note that any number of training examples can be specified during creation of the training set. Depending on the desired number of examples, multiple passes are made through the dataset, where each pass samples a new context stochastically from each dialogue. Very large training sets are possible, yet they will have overlapping examples.

typically not representative of real chat dialogues. Finally, we consider only extracted dialogues that consist of 3 turns or more to encourage the modeling of longer-term dependencies.

To alleviate the problem of 'holes' in the dialogue, where one user does not address the other explicitly, as in Figure 3, we check whether each user talks to someone else for the duration of their conversation. If not, all non-addressed utterances are added to the dialogue. An example conversation along with the extracted dialogues is shown in Figure 3. Note that we also concatenate all consecutive utterances from a given user.

We do not apply any further pre-processing (e.g. tokenization, stemming) to the data as released in the Ubuntu Dialogue Corpus. However the use of pre-processing is standard for most NLP systems, and was also used in our analysis (see Section 4).

3.2.3 SPECIAL CASES AND LIMITATIONS

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

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

3.3 Dataset Statistics

Table 1 summarizes properties of the Ubuntu Dialogue Corpus. One of the most important features of the Ubuntu chat logs is its size. This is crucial for research into building dialogue managers based

on neural architectures. Another important characteristic is the number of turns in these dialogues. The distribution of the number of turns is shown in Figure 4. It can be seen that the number of dialogues and turns per dialogue follow an approximate power law relationship.

3.4 Test Set Generation

We set aside 2% of the Ubuntu Dialogue Corpus conversations to form a test set that can be used for evaluation of response selection algorithms[5]. Compared to the rest of the corpus, this test set has been further processed to extract a pair of *(context, response, flag)* triples from each dialogue. The *flag* is a Boolean variable indicating whether or not the response was the actual next utterance after the given context. The *response* is a target (output) utterance which we aim to correctly identify. The *context* consists of the sequence of utterances appearing in the conversation prior to the response.

We create a pair of triples, where one triple contains the correct response (i.e. the actual next utterance in the dialogue), and the other triple contains a false response, sampled randomly from elsewhere within the test set. The flag is set to 1 in the first case and to 0 in the second case. An example pair is shown in Table 2. To make the task harder, we can move from pairs of responses (one correct, one incorrect) to a larger set of wrong responses (all with flag=0). In our experiments below, we consider both the case of 1 wrong response and 10 wrong responses.

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

Table 2: Test set example with (context, reply, flag) format. The '__eot__' tag is used to denote the end of a user's turn within the context, and the '__eou__' tag is used to denote the end of a user utterance without a change of turn.

Since we want to learn to predict all parts of a conversation, as opposed to only the closing statement, we consider various portions of context for the conversations in the test set. The context size is determined stochastically by uniform sampling[6]:

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

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

5. Note that, contrary to the original Ubuntu Dialogue Corpus, the updated version separates the training, validation, and test sets by time. That is, the training set consists of conversations that started from 2004 to approximately April 27, 2012; the validation set consists of dialogues starting from April 27 to August 7, 2012; and the test set has dialogues from August 7 to December 1, 2012. This mimics the training of dialogue systems in practice, where we only have access to data in the past, and want to answer user queries in the future.
6. Note that this is a different formula than the original Ubuntu Dialogue Corpus, which sampled from a decreasing distribution. The new formula is simpler and leads to longer sampled contexts, which we consider desirable.

Note that, except for the plot in Figure 7, all experiments, results, and analysis in this paper will refer to the updated Ubuntu Dialogue Corpus v2.

4. Response Classification Architectures

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

We first consider *response classification* architectures, which attempt to distinguish between valid and invalid next responses to the context of a conversation. These are trained on the task of best response selection, which we call next utterance classification (NUC). This can be achieved by processing the data as described in Section 3.4, without requiring any human labels. This classification task is an adaptation of the recall and precision metrics previously applied to dialogue datasets (Schatzmann et al., 2005).

Note that retrieval models trained on the task of NUC are still end-to-end, as the natural language understanding, dialogue planning, and generation modules are combined, and the system is learned with a single supervision signal. These models can be used to 'generate' the next utterance in a conversation by retrieving the most probable next utterance from the entire training set, given the context. Thus, we can also evaluate these models using several generative metrics, that compare the selected response to the ground-truth response. We carry this out in Section 4.5.

We consider one naive model and two neural network-based retrieval models. The approaches considered are: TF-IDF, and models using Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM). Prior to applying each method, we perform standard pre-processing of the data using the NLTK[7] library and Twitter tokenizer[8] to parse each utterance. We use generic tags for various word categories, such as names, locations, organizations, URLs, and system paths.

To train the RNN and LSTM architectures, we process the full training Ubuntu Dialogue Corpus into the same format as the test set described in Section 3.4, extracting *(context, response, flag)* triples from dialogues. For the training set, we sample the responses in the same way described in Section 3.4. One can generate any number of training examples by iterating several times through the training data. Negative responses are selected at random from the rest of the training data. We note that for all models presented in this paper, the entire context of the dialogue that is available (i.e. after the context length sampling procedure in Section 3.4 used to create the dataset) is taken into account, and not just the most recent utterance. For the response classification architectures, this is done by concatenating all context utterances together.

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

7. www.nltk.org/
8. http://www.ark.cs.cmu.edu/TweetNLP/

4.1 TF-IDF

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

$$\text{tfidf}(w, d, D) = f(w, d) \times \log \frac{|D|}{|\{d \in D : w \in d\}|}, \tag{1}$$

where $f(w, d)$ indicates the number of times word w appeared in context d and the denominator represents the number of dialogues in which the word w appears.

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

4.2 RNN Dual Encoder

Recurrent neural networks are a variant of neural networks that allows for time-delayed directed cycles between units (Elman, 1990). This leads to the formation of an internal state of the network, h_t at time step (word index) t, which allows it to model time-dependent data. The internal state is updated at each time step as some function of the observed variables x_t, and the hidden state at the previous time step h_{t-1}, with W^x and W^h matrices associated with the input and hidden state:

$$h_t = f(W^h h_{t-1} + W^x x_t). \tag{2}$$

RNNs have been the primary building block of many current neural models for language-related tasks (Sutskever et al., 2014; Sordoni et al., 2015b), which use RNNs as encoders and decoders; in this case, the first RNN is used to encode the given context, and the second RNN generates a response by using beam-search, where its initial hidden state is biased using the final hidden state from the first RNN. We detail such models in Section 5. However, in this section, we are concerned with classification of responses, and thus using a decoder RNN for generation is not strictly necessary (and thus is not used in the model shown in Figure 5). In this section we build upon the approach in (Bordes et al., 2014), which has also been recently applied to the problem of question answering (Yu et al., 2014), and use RNNs for classification rather then generation.

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

$$p(\text{flag} = 1|c, r, M) = \sigma(c^T M r + b), \tag{3}$$

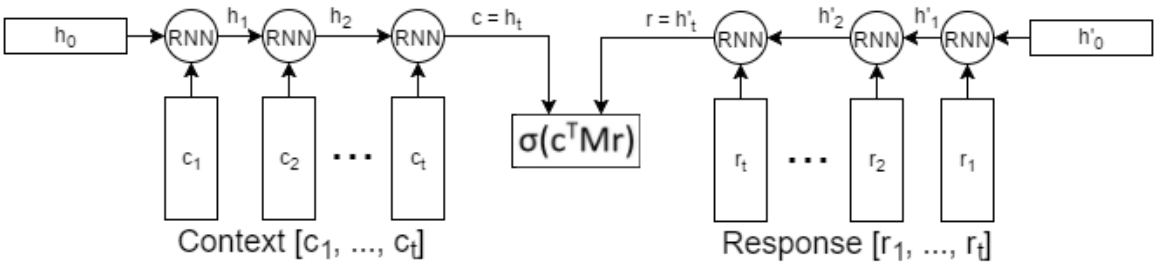

Figure 5: Diagram of the Dual Encoder (DE) model. The RNNs have tied weights. c, r are the last hidden states from the RNNs. c_i, r_i are word vectors for the context and response, $i < t$. We consider contexts up to a maximum of $t = 160$.

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

$$\mathcal{L} = -\sum_{n} \log p(\text{flag}_n | c_n, r_n, M) \tag{4}$$

where $||\theta||_F^2$ is the Frobenius norm of $\theta = \{M, b\}$. A diagram of the DE model can be seen in Figure 5.

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

4.3 LSTM Dual Encoder

In addition to the RNN model, we consider the same architecture but change the hidden units to long-short term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). LSTMs were introduced in order to model longer-term dependencies. This is accomplished using a series of gates that determine whether a new input should be remembered, forgotten (and the old value retained), or used as output. The error signal can now be propagated back much further using the gates of the LSTM unit. This helps overcome the vanishing gradient and exploding gradient problems in standard RNNs, where the error gradients would otherwise decrease or increase at an exponential rate. For this model, we used 1 hidden layer with 200 neurons, a learning rate of 0.001, and a batch size of 256 (optimized over the same values as the RNN). We again use the default Adam settings,

and initialize the forget gate bias of the LSTM to 2.0. The hyper-parameter configuration (including number of neurons) was optimized independently for RNNs and LSTMs using a separate validation set extracted from the training data.

4.4 Evaluation Metrics

We consider two types of evaluation metrics: *retrieval* metrics, and *generative* metrics. These metrics are applicable to both models trained on the task of NUC, detailed in this section, and the generative models introduced in Section 5. In particular, they offer two ways of automatically evaluating dialogue systems trained in an end-to-end manner.

For retrieval, we evaluate using Recall@k (denoted R@1 R@2, R@5 below), which has often been used in language tasks. Here the agent is asked to select the k most likely responses, and it is correct if the true response is among these k candidates. Only the R@1 metric is relevant in the case of binary classification (as in the Table 2 example). Although a language model that performs well on these retrieval metrics is not guaranteed to achieve good performance on utterance generation, we hypothesize that improvements on a model with regards to the classification task will eventually lead to improvements for the generation task. See Section 6 for further discussion of this point.

We also consider generative metrics that compare the generated or retrieved utterance to the ground-truth next utterance. In general, this is a hard open problem (Liu et al., 2016). We use methods based on word embeddings that have recently been proposed for use in evaluating non-task oriented dialogue systems, when no task completion signal is available. These metrics use external word embeddings trained via distributional semantics, such as GloVe, to determine how close the generated utterance is to the ground truth next utterance. We note that these metrics do not necessarily correlate strongly with human judgement (Liu et al., 2016); here, we consider them to be measures of the topicality of the retrieved responses. If the generated response and ground-truth response are semantically similar, then the vector-based metrics should be higher, as word embeddings themselves contain semantic information (Mikolov et al., 2013). It is because of this interpretation that we prefer the vector-based metrics over word-overlap metrics such as BLEU.

The *embedding average score* approximates the compositional embedding of each utterance by taking an average of the word vectors that compose the utterance. The utterance embedding similarities are then compared using cosine similarity. The *greedy matching score*, originally used to analyze semantic similarity between sentences in intelligent tutoring systems (Rus and Lintean, 2012), matches the most similar word in the generated utterance to the actual utterance using cosine similarity of the word embeddings. The *vector extrema score* was proposed by (Forgues et al., 2014) for dialogue systems. Instead of averaging each word embedding, this approach takes the element-wise maximum (or minimum) of each component in the word vectors composing an utterance. This results in utterance embeddings of the same size of each word vector, which can again be compared using cosine similarity. These metrics are able to capture some aspects of dialogue that are not present in BLEU score. We note that, to keep the assumptions of independent and identically distributed (i.i.d.) training and test data examples valid, it is important to have the word embeddings used for the metrics trained on a different corpus than the task corpus, as argued by Liu et al. (2016). This preserves the statistical independence between the task and each performance metric, and alleviates the possibility of spurious and potentially misleading correlations between data examples.

4.5 Experimental Results

We examine the performance of the models using both retrieval and vector based metrics, as shown in Tables 3 and 4. For NUC, the models were evaluated using both 1 (1 in 2) and 9 (1 in 10) false examples.[9]

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

Table 3: Results for the three algorithms using various recall measures for binary (1 in 2) and 1 in 10 (1 in 10) next utterance classification %.

	Generative Metrics		
Method	Embedding Average	Greedy Matching	Vector Extrema
TF-IDF	0.536	0.370	0.342
Dual Encoder w/ LSTM units	**0.650**	**0.413**	**0.376**

Table 4: Results for TF-IDF and the DE model with LSTM units on the embedding average, greedy matching, and vector extrema scores. These scores provide an estimate of the topic consistency of the generated responses.

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

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

Figure 6: Example showing the ranked responses from the LSTM. Each utterance is shown after pre-processing steps.

We observe that the Dual Encoder with LSTM units outperforms both the Dual Encoder with RNN units and TF-IDF on all evaluation metrics. It is interesting to note that TF-IDF actually outperforms the RNN on the Recall@1 case for the 1 in 10 classification. This is most likely due to the limited ability of the RNN to take into account long contexts, which can be overcome by using the LSTM. An example output of the LSTM where the response is correctly classified is shown in Figure 6.

We also show, in Figure 7, the increase in performance of the LSTM as the amount of data used for training increases. This confirms the importance of having a large training set.

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

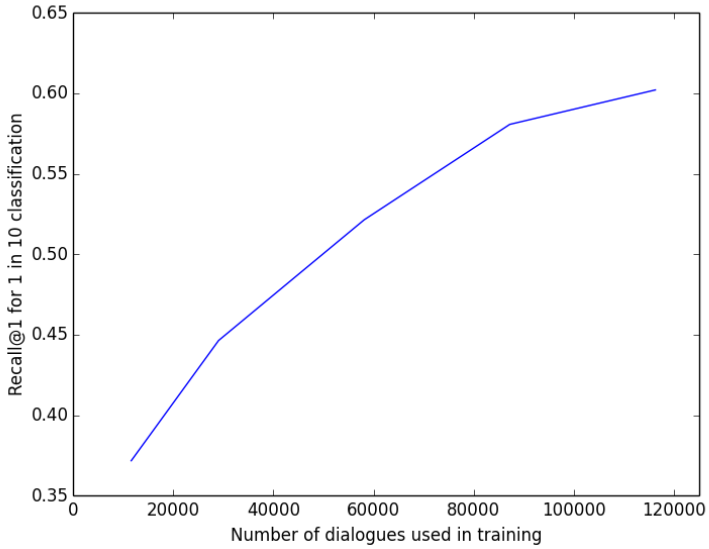

Figure 7: The LSTM (with 200 hidden units), showing Recall@1 for the 1 in 10 classification, with increasing dataset sizes up to 120k dialogues. Note that this was calculated using the old version of the Ubuntu Dialogue Corpus, and thus the Recall@1 values are higher than those in Table 3.

4.6 Qualitative Error Analysis

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

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

We conduct an error analysis with three participants[10] evaluating a total of 100 randomly chosen errors made by the Dual Encoder. For each error made by the model[11], we consider what abilities the model would need to have in order to answer the question correctly. We classify these into several

10. Participants were graduate students in computer science, who had familiarity with both dialogue systems and the Ubuntu domain.
11. We consider an error to be any example where the correct response is not the top 1 response ranked by the model.

categories: using knowledge, understanding tone and style of the responses, a better understanding of the semantic similarity of phrases, and explicitly considering the turn-taking nature of dialogue. Since we are evaluating a classification model, we do not consider problems associated with natural language generation. Note that an error can be classified into multiple categories, if they are each necessary to answer the question correctly.

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

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

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

In the Ubuntu domain, questions where using **external knowledge** would be helpful for the model involve technical terminology. In most cases, the correct response contains the name of a command or process that is related to one stated in the context; however, the model is unable to link the two together. An example of such a question is shown in Figure 8. In this case, the context of the conversation is about file searching in Ubuntu, and the correct response (in italics) mentions the locate command. This response would have been assigned a higher probability if it was able to determine the meaning of the locate command.

There are other examples where the model may be incapable of taking into account the specific **tone or style** of the users in the conversation. For instance, a speaker may use many emoticons, have poor English grammar skills and be prone to misspelling words, use frequent abbreviations, or use a particularly formal tone. Being able to spot these distinct language features could lead the model to improve its performance in terms of selecting the actual next response. An example of this is shown in Figure 9. In the context, Speaker A appends his question with an (unnecessary) smiley face. Thus, it is more likely that the candidate response with multiple smiley faces is the correct response.

One of the most important challenges in natural language is understanding the semantics of phrases. Classification dialogue models can make errors due to an inability to detect **semantic similarities** between sentences, or due to the detection of spurious similarities. This category covers the general case where the topic of the model's response is clearly different than the topic of the context and true response. In this category, we also define two special cases: one where there is a

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

Figure 8: Example where the model would benefit from using an external knowledge source. Correct answer is in italics, and the model's selected answer is in bold. Note that the probabilities do not sum up to 1, as they are binary probabilities – the model considers each candidate response independently.

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

Figure 9: Example where the model would benefit from understanding the tone and style of the speakers. Correct answer is in italics, and the model's selected answer is in bold.

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

Figure 10: Example where the model would benefit from the ability to conduct high-level inference to better understand the semantic similarity between context and correct response. Correct answer is in italics, and the model's selected answer is in bold.

direct **word copying** between the context and true response that the model failed to detect, and one where some **high-level inference** is required to answer correctly. An example of the latter case is shown in Figure 10; Speaker A asks how to install some programs automatically, and the correct response states that they had previously been 'doing it manually'. Thus, if the model was able to infer that a person who has asked to perform an operation automatically could previously have been doing it manually, it would have assigned a higher probability to the correct response.

Finally, we consider errors where the model is unable to account for the **turn-taking structure** of dialogue. For the Ubuntu Dialogue Corpus, interactions between users usually take a certain form, where one user is asking for help and the other user is providing answers. Thus, it is important to consider the role of the current user when selecting the correct response; indeed, there has been preliminary work in this direction (Luan et al., 2016; Li et al., 2016a). We also consider a special case of this error, when the last utterance in the context **asks a question** and the response chosen by the model is not answering any question at all. For example, in Figure 11, the final utterance asks the question 'no MM's?'. The response selected by the model begins with 'thanks', which is clearly not a reasonable response to a question.

An example of the general turn-taking error is shown in Figure 12. This depicts a typical dialogue between two users in the Ubuntu Dialogue Corpus: Speaker A is having trouble with their brightness keys, and Speaker B is trying to help them. The model must predict the next response of Speaker A. In the first response, the user states that they are appreciative of the help being given, which fits with Speaker A's behaviour in the context; thus, it is more likely to be the correct response.

We examined 100 randomly selected errors of the DE model on the Ubuntu Corpus to compute the number of errors in each category; the results are shown in Table 13. We can first note that there is significant progress to be made for classification models on the Ubuntu Dialogue Corpus; over half (60%) of the errors made by the model can be considered feasible for the majority of humans (1-3 on the difficulty rating). However, the number of questions that every human could

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

Figure 11: Example where the model selects an inappropriate response to a question. Correct answer is in italics, and the model's selected answer is in bold.

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

Figure 12: Example where the model does not take into account the roles of the participants in the dialogue. Correct answer is in italics, and the model's selected answer is in bold.

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

Figure 13: Qualitative evaluation of the errors from the DE model. Note that counts for parent categories (semantic similarity and turn-taking structure) include the counts for the child categories. Error categories are not classified for impossible questions and are not mutually exclusive, thus totals may not add up to 100.

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

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

There are many examples where the model could be improved by explicitly accounting for the turn-taking structure of dialogue, as there were often instances where the model selected a response that was not suited to the current speaker. In several cases, the model also needed some form of external knowledge base in order to answer the question correctly. Note that the number of such examples in Table 13 refers to instances where the correct response mentions a Ubuntu term that is related but not identical to the terminology in the context; if this were to be extended to all questions where technical vocabulary is mentioned, the number would be significantly higher. Finally, there is a small number of cases where a better understanding of the tone of the dialogue would help the model, however this does not seem to be the best direction for future research.

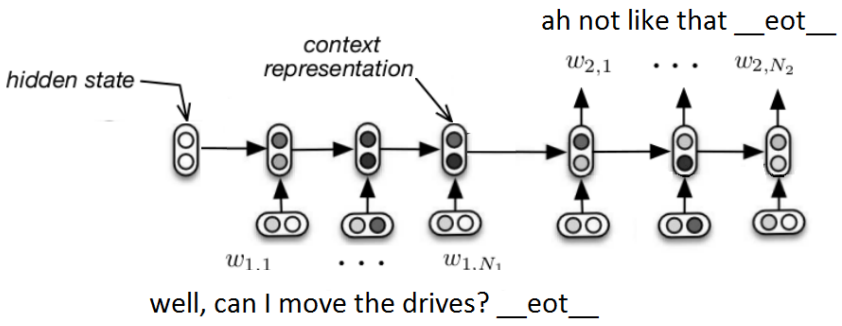

Figure 14: Diagram of the RNN architecture for dialogue modeling. Note that utterances in the context are concatenated together before being fed into the RNN.

5. Generative Response Architectures

In order to aid progress towards the goal of building fully generative conversational models, we present baseline models for generating responses conditioned on the context of the conversation for the Ubuntu Dialogue Corpus. It should be noted that the format of the dataset can easily be altered to support training in this manner: one can simply remove all *(context, response, flag)* triples with $flag = 0$, and be left with only the valid *(context,response)* pairs.

5.1 Generative Recurrent Neural Language Model

We first describe the standard recurrent neural network language model (RNN-LM), as shown in Figure 4.6, a neural network model which is used to predict the next word in a sequence of words (Mikolov et al., 2010). The model observes the dialogue word-by-word and updates its hidden state h_t at time step (word index) t. Given a hidden state h_t the model outputs a probability distribution over all words in the vocabulary. Formally, the model computes the hidden state as described in Equation (2): $h_t = f(W^h h_{t-1} + W^x x_t)$, where, h_{t-1} is the previous hidden state, x_t is the current input word, $f(\cdot)$ is a non-linear activation function such as tanh, and W^x and W^h are model parameters. We will take $\hat{y}_n = \{w_1, ..., w_T\}$ as the target output sequence for the n^{th} training example, given the input sequence $\hat{x}_n = \{x_1, ..., x_T\}$. The conditional distribution over each output symbol is computed in a similar manner, and depends on the current hidden state h_t:

$$p(w_t|w_{t-1}, ..., w_1) = \frac{\exp(W^o_{w_t} \cdot h_t)}{\sum_w \exp([W^o]_w \cdot h_t)}, \tag{5}$$

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

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

$$\max_\theta \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(\hat{y}_n|\hat{x}_n), \tag{6}$$

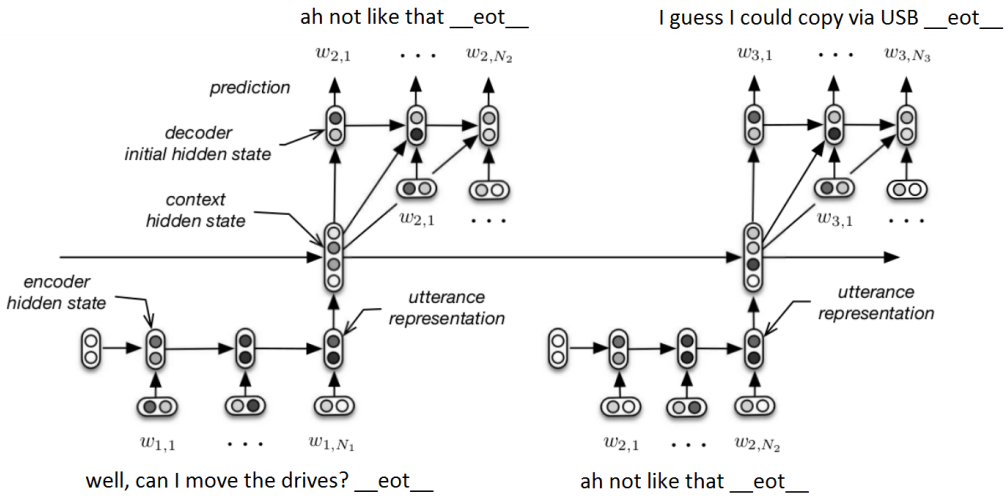

Figure 15: Diagram of the HRED model. Note that each utterance in the context is encoded with a separate 'utterance-level' encoder, which is then fed into a 'context-level' encoder.

where θ are the parameters of the model, including W^x, W^h, W^o, and the corresponding biases.

Thus, the model learns a probability distribution over all output sequences, $p(\hat{y}_1, ..., \hat{y}_T)$.

For dialogue response generation, the model is conditioned on the previous dialogue context and used to generate a response, i.e. the next utterance in the dialogue. Such a model could be used as a full dialogue system, as defined in Section 1.1, to carry out a conversation with a user.

5.2 Hierarchical Recurrent Encoder-Decoder

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

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

model is trained end-to-end to maximize the log-likelihood of the generated utterance. A diagram of the HRED model can be seen in Figure 5.1.

To summarize: the RNN-LM (Figure 4.6) uses a single RNN (or LSTM) to encode the entire history of the dialogue, which consists of all utterances in the context concatenated together. It then uses the same RNN (i.e. an RNN with the same parameters) to decode the prediction utterance. The Encoder-Decoder model (not shown here) augments this with a second RNN with different parameters for the decoder. All context utterances are still concatenated in the encoder, and thus it is difficult to model long-term dependencies for utterances that occur earlier in the dialogue. This problem is alleviated with the HRED model (Figure 5.1), which does not concatenate the context utterances: each is encoded with a separate utterance-level encoder, whose output is fed into an additional context-level encoder. The output of the context-level encoder depends on all the utterances in the context, and is fed into the decoder. Each RNN has separate parameters.

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

Table 5: Results for both the generative and retrieval models on the embedding average, greedy matching, and vector extrema scores. These scores provide an estimate of the topic consistency of the generated responses.

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

Table 6: Results for both the generative and retrieval models using various recall measures for binary (1 in 2) and 1 in 10 (1 in 10) next utterance classification %. We include state-of-the-art results from more recent papers.

5.3 Experimental Results

We now examine the performance of the generative models using the vector-based metrics defined in Section 4.4. The results for the RNN language model with LSTM units (LSTM-LM) and the HRED model can be seen in Figure 5. As expected, the HRED model outperforms the baseline LSTM model across all of the metrics. However, it is interesting to note that in a direct comparison with the Dual Encoder model, the HRED model has a higher score in 2 out of the 3 metrics considered. Thus, it is likely that the HRED model is generating responses that are more semantically similar to the ground-truth response, and is better at staying on topic.

We also present the results for these models on the NUC task. It is possible to apply the generative models to the NUC task, as all that is required is the ability to assign probabilities to sequences of utterances. Again, the HRED model predictably outperforms the LSTM-LM model on all metrics. This coincides with the results of (Serban et al., 2016) on a different dataset, using different metrics. We can also see that the generative models perform much worse than the models explicitly trained to retrieve utterances from a list. This is to be expected, as the retrieval models were trained explicitly on the NUC task, while the generative models were not. Because of the discrepancy in training objectives, we do not recommend the use of NUC for comparing generative models with retrieval models. However, we believe that NUC is very useful for comparing generative models with other generative models, and retrieval models with retrieval models; we justify this further in Section 6.3.

5.4 Examples of Generated Responses

In order to obtain a better understanding of the quality of responses from the generative model, we provide a table of examples from the LSTM-LM and HRED models in Table 7. We chose several representative samples that provide insight into the behaviour, and the limitations, of the generative models.[12]

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

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

There are other instances where the lack of semantic understanding is clear. In example 2), the HRED makes a relevant statement about the command 'sudo apt-get', stating that the user should try to run this command. However, it fails to understand that the last user utterance ('I tried that, it comes the same error'), refers to the user having already tried this exact command. It is difficult for these models to integrate information from multiple utterances (Li et al., 2016b), and to understand the concept of coreference.

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

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

Table 7: Ubuntu model examples. The → token indicates a change of turn, and <unk> is the unknown token.

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

6. Discussion

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

Next, we outline several interesting directions for future work.

6.1 Conversation Disentanglement

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

6.2 Non-Task Oriented Model Evaluation

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

The reasons for this are purely practical; in general, training large, end-to-end goal-driven models is very difficult as it requires the collection of a large amount of task completion data. Annotating data in this way on a large scale is extremely expensive, and is usually only feasible for technical support channels at large corporations, which are rarely released publicly. Indeed, the Ubuntu Dialogue Corpus has no such labelled task completion data, and thus cannot be analyzed in the task-oriented setting for the time being. Obtaining such signals automatically remains an open problem. On the other hand, training non-task oriented dialogue systems such as chatbots only requires conversational data, which can be obtained and shared publicly on a large scale. We believe that significant progress in dialogue systems can be made in this manner, as there remains many unsolved problems as illustrated in Section 4.6.

6.3 Automatic Evaluation of Dialogue Systems

A crucial part of research in building dialogue systems concerns the problem of evaluation. In the goal-oriented setting, when there is a supervised task completion signal available with the data, methods for automatic evaluation are well-established, such as PARADISE (Walker et al., 1997) and MeMo (Möller et al., 2006). An overview of such methods can be found in Jokinen and McTear (2009) and Hastie (2012).

However, in the non-goal oriented setting we consider here, evaluation is more difficult. This is particularly true for end-to-end systems, as there is no way to measure the accuracy of the state tracking module using tasks such as slot filling, since they are not modular systems. Indeed, there

are several reasons for wanting to move away from the slot filling metrics that have become common for modular systems. In slot filling, the set of candidate outputs (*states*) is identified *a priori* through knowledge engineering, and is typically rather small in comparison to the set of responses considered by NUC. Further, it has been speculated that state-of-the-art state-tracking models (Henderson et al., 2014b; Williams, 2014) are achieving close to human-level performance. Thus it is desirable to move beyond this domain into more difficult problems. To do this, it is crucial to investigate ways to evaluate models in the non-goal oriented setting that do not require supervised test data for the internal modules of a system.

Researchers have previously proposed measuring word perplexity and word classification error rate, as these are widely applied in the language modeling and automatic speech recognition community (Serban et al., 2016; Vinyals and Le, 2015; Pietquin and Hastie, 2013). However, these metrics cannot be computed for retrieval models. Researchers have also proposed to use word overlap metrics from machine translation (Galley et al., 2015; Sordoni et al., 2015b). However, such metrics based on word overlaps suffer from severe sparsity issues, since it is unlikely that any sequence of words will be identical in both the generated and reference responses. These have shown to correlate poorly with human judgements when only a single ground-truth response is available (Liu et al., 2016), and they have at best a mediocre correlation when multiple ground-truth responses are available (Galley et al., 2015). Furthermore, it has been argued that such metrics mainly focus on pronouns and punctuation marks when applied to non-task oriented dialogue datasets (Serban et al., 2016). While the word embedding metrics used here do not have a strong correlation with human judgement, they have an additional interpretation of measuring the *semantic similarity* between the generated and reference responses (as argued in Section 4.4), which is why we favour them over word-overlap scores such as BLEU. However, we reiterate that none of these metrics measures the *coherence* of the generated responses, and this remains an important direction for future work.

Another option for evaluating dialogue systems trained in an end-to-end manner is using an alternative task such as next utterance classification. While this does not directly compare the generated response of the system to the ground-truth response, there are several reasons for preferring the recall metric:

1. It is a more difficult task than slot filling, and thus will require further development of more sophisticated dialogue systems in order to solve the task.

2. It does not suffer from the same problems as the word overlap metrics, as it does not have to directly compare the quality of a generated response to the ground-truth response, an inherently noisy process. Instead, it measures the model's capacity to pick out the correct response from a list of responses.

3. Performance using the recall metrics is easily interpretable, and can easily be compared to human performance. Indeed, this has been recently done by Lowe et al. (2016), who show that human performance on this task is above the performance for the Dual Encoder model on the Ubuntu Dialogue Corpus, as well as on movie and Twitter corpora. Thus, there is room for improvement for models on this task.

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

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

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

6.4 Future Research Directions for End-to-End Systems

Given the analysis performed in Section 4.6, we postulate several interesting directions for future research on end-to-end dialogue systems, particularly on the Ubuntu Dialogue Corpus.

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

The largest source of errors from the analysis in Section 4.6 was in the failure to understand the semantic similarity between the context and response. This falls under the more general problem of *natural language understanding*, which arises in many NLP tasks. This will require adjustments in the architecture of end-to-end models to render them more suited to processing language. It is possible that insights can be derived from architectures developed on more targeted language understanding tasks, such as the CNN/ Daily Mail reading comprehension dataset (Hermann et al., 2015), where attention-based models have achieved strong performance.

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

A common problem that has been observed when training generative end-to-end models that maximize the log-likelihood of the conversational response is that these models tend to produce generic responses at test time. This has been observed empirically (Vinyals and Le, 2015; Serban et al., 2016), and was also seen in some of the LSTM and HRED examples presented in Section

5.4. This has been investigated in (Li et al., 2015), where the authors construct an objective function based on mutual information that promotes diversity, however they achieve only modest improvements. This is a large impediment for building end-to-end systems that can have interesting and engaging interactions with users.

Finally, an important direction for future research is building large-scale datasets that allow the training of goal-oriented systems. The Ubuntu domain is particularly suited for training goal-oriented systems, however this is not yet possible on the Ubuntu Dialogue Corpus as there are no supervised task completion signals, as mentioned in Section 6.2. Building models that can approximate such signals is challenging, yet it may be necessary in order to develop systems that can solve users' problems in a meaningful way in a domain as complex as Ubuntu.

Acknowledgments

The authors would like to profusely thank Rudolf Kadlec and Martin Schmid, who produced the script for generating the updated version of the Ubuntu Dialogue Corpus. We gratefully acknowledge financial support for this work by the Samsung Advanced Institute of Technology (SAIT) and the Natural Sciences and Engineering Research Council of Canada (NSERC). We would like to thank Michael Noseworthy and Nicolas Angelard-Gontier for their input into this paper, and the anonymous reviewers for their helpful feedback.

References

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

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

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

J. Boyd-Graber, B. Satinoff, H. He, and H. Daume. Besting the quiz master: Crowdsourcing incremental classification games. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2012.

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

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

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

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

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

G. Forgues, J. Pineau, J.M. Larchevêque, and R. Tremblay. Bootstrapping dialog systems with word embeddings. In *Proceedings of the Workshop on Modern Machine Learning and NLP, NIPS*, 2014.

M. Galley, C. Brockett, A. Sordoni, Y. Ji, M. Auli, C. Quirk, M. Mitchell, J. Gao, and B. Dolan. deltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. *arXiv preprint arXiv:1506.06863*, 2015.

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

Helen Hastie. Metrics and evaluation of spoken dialogue systems. In *Data-Driven Methods for Adaptive Spoken Dialogue Systems*, pages 131–150. Springer, 2012.

M. Henderson, B. Thomson, and J. Williams. The second dialog state tracking challenge. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, page 263, 2014a.

M. Henderson, B. Thomson, and S. Young. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2014b.

K. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2015.

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

Michimasa Inaba and Kenichi Takahashi. Neural utterance ranking model for conversational dialogue systems. In *Proceedings of the Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2016.

S. Jafarpour, C. Burges, and A. Ritter. Filter, rank, and transfer the knowledge: Learning to chat. *Advances in Ranking*, 10, 2010.

K. Jokinen and M. McTear. *Spoken Dialogue Systems*. Morgan Claypool, 2009.

R. Kadlec, M. Schmid, and J. Kleindienst. Improved deep learning baselines for Ubuntu corpus dialogs. In *Proceedings of the Workshop on Spoken Language Understanding, NIPS*, 2015.

D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations*, 2014.

Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication*, 51(5):466–484, 2009.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the Meeting of the North American Chapter of the Association for Computational Linguistics*, 2015.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A persona-based neural conversation model. In *Proceedings of the Association for Computational Linguistics*, 2016a.

J. Li, W. Monroe, A. Ritter, and D. Jurafsky. Deep reinforcement learning for dialogue generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016b.

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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

R. Lowe, N. Pow, I. Serban, and J. Pineau. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2015.

R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. On the evaluation of dialogue systems with next utterance classification. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2016.

Y. Luan, Y. Ji, and M. Ostendorf. LSTM based conversation models. *arXiv preprint arXiv:1603.09457*, 2016.

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*, 2010.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2013.

S. Möller, R. Englert, K. Engelbrecht, V. Hafner, A. Jameson, A. Oulasvirta, A. Raake, and N. Reithinger. Memo: towards automatic usability evaluation of spoken dialogue services by user error simulations. In *Proceedings of INTERSPEECH*, 2006.

Lasguido Nio, Sakriani Sakti, Graham Neubig, Tomoki Toda, Mirna Adriani, and Satoshi Nakamura. Developing non-goal dialog system based on examples of drama television. In *Natural Interaction with Robots, Knowbots and Smartphones*, pages 355–361. Springer, 2014.

J. Pennington, R. Socher, and C.D. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

O. Pietquin and H. Hastie. A survey on metrics for the evaluation of user simulations. *The Knowledge Engineering Review*, 28(01):59–73, 2013.

J. Ramos. Using TF-IDF to determine word relevance in document queries. In *Proceedings of the International Conference on Machine Learning*, 2003.

A. Ritter, C. Cherry, and W. Dolan. Unsupervised modeling of twitter conversations. In *Proceedings of the Meeting of the North American Chapter of the Association for Computational Linguistics*, 2010.

A. Ritter, C. Cherry, and W. Dolan. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.

V. Rus and M. Lintean. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, 2012.

A.M. Saxe, J.L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

J. Schatzmann, K. Georgila, and S. Young. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2005.

I. V. Serban, R. Lowe, L. Charlin, and J. Pineau. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*, 2015.

I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*, 2015.

S. Singh, D. Litman, M. Kearns, and M. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, Jakob Grue S., and J. Y. Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the ACM International on Conference on Information and Knowledge Management*, pages 553–562, 2015a.

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.Y. Nie, J. Gao, and W. Dolan. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the Meeting of the North American Chapter of the Association for Computational Linguistics*, 2015b.

I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2014.

D. Traum, K. Georgila, R. Artstein, and A. Leuski. Evaluating spoken dialogue processing for time-offset interaction. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2015.

D.C. Uthus and D.W. Aha. Extending word highlighting in multiparticipant chat. In *Proceedings of the Florida Artificial Intelligence Research Society Conference*, 2013a.

D.C. Uthus and D.W. Aha. The Ubuntu chat corpus for multiparticipant chat analysis. In *Proceedings of the AAAI Spring Symposium on Analyzing Microtext*, 2013b.

O. Vinyals and Q. Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

M. Walker, D. Litman, C. Kamm, and A. Abella. PARADISE: A framework for evaluating spoken dialogue agents. In *Proceedings of the Meeting of the European Chapter of the Association for Computational Linguistics*, 1997.

H. Wang, Z. Lu, H. Li, and E. Chen. A dataset for research on short-text conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013.

M. Wen, T.and Gasic, N. Mrksic, L. Rojas-Barahona, P. Su, S. Ultes, D. Vandyke, and S. Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.

J. Williams, A. Raux, D. Ramachandran, and A. Black. The dialog state tracking challenge. In *Proceedings of the Meeting of the Special Interest Group on Dialogue and Discourse (SIGDIAL)*, 2013.

J. Williams, A. Raux, and M. Henderson. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33, 2016.

J. D. Williams. Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of the Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014.

Z. Xu, B. Liu, B. Wang, C. Sun, and X. Wang. Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling. *arXiv preprint arXiv:1605.05110*, 2016.

S. J. Young. Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 358(1769): 1389–1402, 2000.

L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.

M.D. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Tiancheng Zhao and Maxine Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of the Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2016.