The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems

Ryan Lowe [*], Nissan Pow[*], Iulian V. Serban[†] and Joelle Pineau[*]

[*]School of Computer Science, McGill University, Montreal, Canada
[†]Dept Computer Science and Operations Research, Université de Montréal, Montreal, Canada
{ryan.lowe, nissan.pow}@mail.mcgill.ca, julianserban@gmail.com, jpineau@cs.mcgill.ca

Abstract

This paper introduces the Ubuntu Dialogue Corpus, a dataset containing almost 1 million multi-turn dialogues, with a total of over 7 million utterances and 100 million words. This provides a unique resource for research into building dialogue managers based on neural language models that can make use of large amounts of unlabeled data. The dataset has both the multi-turn property of conversations in the Dialog State Tracking Challenge datasets, and the unstructured nature of interactions from microblog services such as Twitter. We also describe two neural learning architectures suitable for analyzing this dataset, and provide benchmark performance on the task of selecting the best next response.

1   Introduction

The ability for a computer to converse in a natural and coherent manner with a human has long been held as one of the primary objectives of artificial intelligence (AI). In this paper we consider the problem of building dialogue agents that have the ability to interact in one-on-one multi-turn conversations on a diverse set of topics. We primarily target *unstructured* dialogues, where 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 a slot-filling representation [10, 27, 32].

We observe that in several subfields of AI—computer vision, speech recognition, machine translation—fundamental break-throughs were achieved in recent years using machine learning

methods, more specifically with neural architectures [1]; however, it is worth noting that many of the most successful approaches, in particular convolutional and recurrent neural networks, were known for many years prior. It is therefore reasonable to attribute this progress to three major factors: 1) the public distribution of very large rich datasets [5], 2) the availability of substantial computing power, and 3) the development of new training methods for neural architectures, in particular leveraging unlabeled data. Similar progress has not yet been observed in the development of dialogue systems. We hypothesize that this is due to the lack of sufficiently large datasets, and aim to overcome this barrier by providing a new large corpus for research in multi-turn conversation.

The new Ubuntu Dialogue Corpus consists of almost one million two-person conversations extracted from the Ubuntu chat logs[1], used to receive technical support for various Ubuntu-related problems. The conversations have an average of 8 turns each, with a minimum of 3 turns. All conversations are carried out in text form (not audio). The dataset is orders of magnitude larger than structured corpuses such as those of the Dialogue State Tracking Challenge [32]. It is on the same scale as recent datasets for solving problems such as question answering and analysis of microblog services, such as Twitter [22, 25, 28, 33], but each conversation in our dataset includes several more turns, as well as longer utterances. Furthermore, because it targets a specific domain, namely technical support, it can be used as a case study for the development of AI agents in targeted applications, in contrast to chatbox agents that often lack a well-defined goal [26].

In addition to the corpus, we present learning architectures suitable for analyzing this dataset, ranging from the simple frequency-inverse docu-

---

The first two authors contributed equally.

[1]These logs are available from 2004 to 2015 at `http://irclogs.ubuntu.com/`

ment frequency (TF-IDF) approach, to more sophisticated neural models including a Recurrent Neural Network (RNN) and a Long Short-Term Memory (LSTM) architecture. We provide benchmark performance of these algorithms, trained with our new corpus, on the task of selecting the best next response, which can be achieved without requiring any human labeling. The dataset is ready for public release[2]. The code developed for the empirical results is also available[3].

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 (due to space constraints), but surveys resources most related to our contribution. A list of datasets discussed is provided in Table 1.

2.1 Dialogue Datasets

The Switchboard dataset [8], and the Dialogue State Tracking Challenge (DSTC) datasets [32] have been used to train and validate dialogue management systems for interactive information retrieval. 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 quite small compared to datasets currently used for training neural architectures.

Recently, a few datasets have been used containing unstructured dialogues extracted from Twitter[4]. Ritter et al. [21] collected 1.3 million conversations; this was extended in [28] to take advantage of longer contexts by using A-B-A triples. Shang et al. [25] used data from a similar Chinese website called Weibo[5]. 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. estimate that only 37% of posts on Twitter are 'conversational in nature', and 69%

of their collected data contained exchanges of only length 2 [21]. We hypothesize that chat-room style messaging is more closely correlated to human-to-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 [30]. 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 [3], however these interactions are much shorter than what we seek to study.

2.2 Learning Architectures

Most dialogue research has historically focused on structured slot-filling tasks [24]. 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. [11], which proposes an RNN structure, initialized with a denoising autoencoder, to tackle the DSTC 3 domain.

Work on unstructured dialogues, recently pioneered by Ritter et al. [22], proposed a response generation model for Twitter data based on ideas from Statistical Machine Translation. This is shown to give superior performance to previous information retrieval (e.g. nearest neighbour) approaches [14]. This idea was further developed by Sordoni et al. [28] to exploit information from a longer context, using a structure similar to the Recurrent Neural Network Encoder-Decoder model [4]. 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. [22]. Their results are also verified with a human-subject study. A similar encoder-decoder framework is presented in [25]. This model 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. This model is also evaluated in a human-subject study, although much smaller in size than in [28]. Overall, these models highlight the potential of neural learning architectures for interactive systems, yet so far they have

---

Dataset	Type	Task	# Dialogues	# Utterances	# Words	Description
Switchboard [8]	Human-human spoken	Various	2,400	—	3,000,000	Telephone conversations on pre-specified topics
DSTC1 [32]	Human-computer spoken	State tracking	15,000	210,000		Bus ride information system
DSTC2 [10]	Human-computer spoken	State tracking	3,000	24,000	—	Restaurant booking system
DSTC3 [9]	Human-computer spoken	State tracking	2,265	15,000	—	Tourist information system
DSTC4[13]	Human-human spoken	State tracking	35	—	—	21 hours of tourist info exchange over Skype
Twitter Corpus [21]	Human-human micro-blog	Next utterance generation	1,300,000	3,000,000	—	Post/ replies extracted from Twitter
Twitter Triple Corpus [28]	Human-human micro-blog	Next utterance generation	29,000,000	87,000,000	—	A-B-A triples from Twitter replies
Sina Weibo [25]	Human-human micro-blog	Next utterance generation	4,435,959	8,871,918	—	Post/ reply pairs extracted from Weibo
Ubuntu Dialogue Corpus	**Human-human chat**	**Next utterance classification**	**930,000**	**7,100,000**	**100,000,000**	**Extracted from Ubuntu Chat Logs**

Table 1: A selection of structured and unstructured large-scale datasets applicable to dialogue systems. Faded datasets are not publicly available. The last entry is our contribution.

been limited to very short conversations.

3 The Ubuntu Dialogue Corpus

We seek a large dataset for research in dialogue systems with the following properties:

- Two-way (or *dyadic*) conversation, as opposed to multi-participant chat, preferably human-human.
- Large number of conversations; $10^5 - 10^6$ is typical of datasets used for neural-network learning in other areas of AI.
- Many conversations with several turns (more than 3).
- Task-specific domain, as opposed to chatbot systems.

All of these requirements are satisfied by the Ubuntu Dialogue Corpus presented in this paper.

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 [29], 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. Figure 4 shows 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 replies and initial questions 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.

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 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 in-

tended 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[6], 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 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 5, 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 5. 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

---

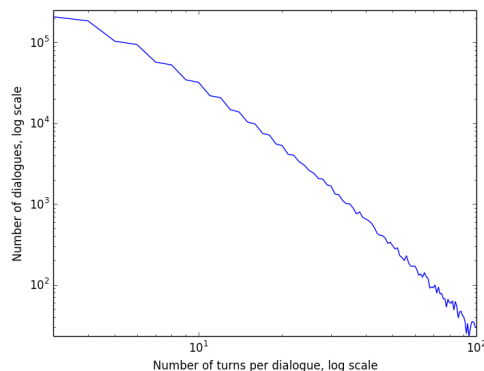[6]We use the GNU Aspell spell checking dictionary.

Figure 1: Plot of number of conversations with a given number of turns. Both axes use a log scale.

# dialogues (human-human)	930,000
# 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

Table 2: Properties of Ubuntu Dialogue Corpus.

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 2 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 1. 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 (randomly selected) to form a test set that can be used for evaluation of response selection algorithms. 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 dialogue prior to the response. We create a pair of triples, where one triple contains the correct response (i.e. the actual next utterance in the dialogue), and the other triple contains a false response, sampled randomly from elsewhere within the test set. The flag is set to 1 in the first case and to 0 in the second case. An example pair is shown in Table 3. 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? __EOS__ ah not like that	I guess I could just get an enclosure and copy via USB	1
well, can I move the drives? __EOS__ ah not like that	you can use "ps ax" and "kill (PID #)"	0

Table 3: Test set example with (context, reply, flag) format. The '__EOS__' tag is used to denote the end of an utterance within the context.

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 using a simple formula:

$$c = \min(t - 1, n - 1),$$

$$\text{where } n = \frac{10C}{\eta} + 2, \eta \sim Unif(C/2, 10C)$$

Here, C denotes the maximum desired context size, which we set to $C = 20$. The last term is

the desired minimum context size, which we set to be 2. Parameter t is the actual length of that dialogue (thus the constraint that $c \leq t - 1$), and n is a random number corresponding to the randomly sampled context length, that is selected to be inversely proportional to C.

In practice, this leads to short test dialogues having short contexts, while longer dialogues are often broken into short or medium-length segments, with the occasional long context of 10 or more turns.

3.5 Evaluation Metric

We consider the task of best response selection. 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 [24].

A family of metrics often used in language tasks is Recall@k (denoted R@1 R@2, R@5 below). Here the agent is asked to select the k most likely responses, and it is correct if the true response is among these k candidates. Only the R@1 metric is relevant in the case of binary classification (as in the Table 3 example).

Although a language model that performs well on response classification is not a gauge of good performance on next 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.

4 Learning Architectures for Unstructured Dialogues

To provide further evidence of the value of our dataset for research into neural architectures for dialogue managers, we provide performance benchmarks for two neural learning algorithms, as well as one naive baseline. The approaches considered are: TF-IDF, 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.

---

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

To train the RNN and LSTM architectures, we process the full training Ubuntu Dialogue Corpus into the same format as the test set described in Section 3.4, extracting *(context, response, flag)* triples from dialogues. For the training set, we do not sample the context length, but instead consider each utterance (starting at the 3rd one) as a potential response, with the previous utterances as its context. So a dialogue of length 10 yields 8 training examples. Since these are overlapping, they are clearly not independent, but we consider this a minor issue given the size of the dataset (we further alleviate the issue by shuffling the training examples). Negative responses are selected at random from the rest of the training data.

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 [20]. 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:

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

where $f(w, d)$ indicates the number of times word w appeared in context d, N is the total number of dialogues, 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

Recurrent neural networks are a variant of neural networks that allows for time-delayed directed cycles between units [17]. This leads to the formation of an internal state of the network, h_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}. W_x and
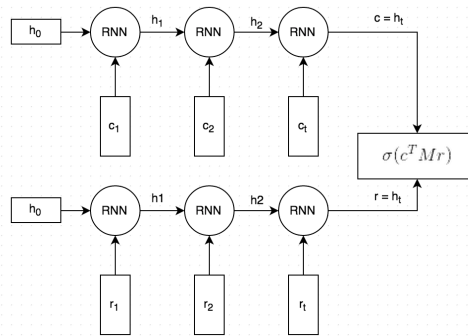

Figure 2: Diagram of our model. The RNNs have tied weights. c, r are the last hidden states from the RNNs. c_i, r_i are word vectors for the context and response, $i < t$. We consider contexts up to a maximum of $t = 160$.

W_h are matrices associated with the input and hidden state.

$$h_t = f(W_h h_{t-1} + W_x x_t).$$

A diagram of an RNN can be seen in Figure 2. RNNs have been the primary building block of many current neural language models [22, 28], which use RNNs for an encoder and decoder. 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. In our work, we are concerned with classification of responses, instead of generation. We build upon the approach in [2], which has also been recently applied to the problem of question answering [33].

We utilize a siamese network consisting of two RNNs with tied weights to produce the embeddings for the context and response. 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 [19]), 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:

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

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 [33]:

$$\mathcal{L} = -\log \prod_n p(\text{flag}_n | c_n, r_n) + \frac{\lambda}{2}||\theta||_2^F$$

where $||\theta||_2^F$ is the Frobenius norm of $\theta = \{M, b\}$. In our experiments, we use $\lambda = 0$ for computational simplicity.

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 50 neurons. The W_h matrix is initialized using orthogonal weights [23], while W_x is initialized using a uniform distribution with values between -0.01 and 0.01. We use Adam as our optimizer [15], with gradients clipped to 10. We found that weight initialization as well as the choice of optimizer were critical for training the RNNs.

4.3 LSTM

In addition to the RNN model, we consider the same architecture but changed the hidden units to long-short term memory (LSTM) units [12]. 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 fed back indefinitely into the gates of the LSTM unit. This helps overcome the vanishing and exploding gradient problems in standard RNNs, where the error gradients would otherwise decrease or increase at an exponential rate. In training, we used 1 hidden layer with 200 neurons. The hyper-parameter configuration (including number of neurons) was optimized independently for RNNs and LSTMs using a validation set extracted from the training data.

5 Empirical Results

The results for the TF-IDF, RNN, and LSTM models are shown in Table 4. The models were evaluated using both 1 (1 in 2) and 9 (1 in 10) false

examples. Of course, the Recall@2 and Recall@5 are not relevant in the binary classification case.

Method	TF-IDF	RNN	LSTM
1 in 2 R@1	65.9%	76.8%	87.8%
1 in 10 R@1	41.0%	40.3%	60.4%
1 in 10 R@2	54.5%	54.7%	74.5%
1 in 10 R@5	70.8%	81.9%	92.6%

Table 4: Results for the three algorithms using various recall measures for binary (1 in 2) and 1 in 10 (1 in 10) next utterance classification %.

We observe that the LSTM outperforms both the RNN 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 Table 5.

We also show, in Figure 3, 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.

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

Table 5: Example showing the ranked responses from the LSTM. Each utterance is shown after preprocessing steps.

6 Discussion

This paper presents the Ubuntu Dialogue Corpus, 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 preliminary results demonstrating use of this dataset to train an RNN and an LSTM for the task of selecting the next best response in a conversation; we obtain significantly better results with the LSTM architecture. There are several interesting directions for future work.
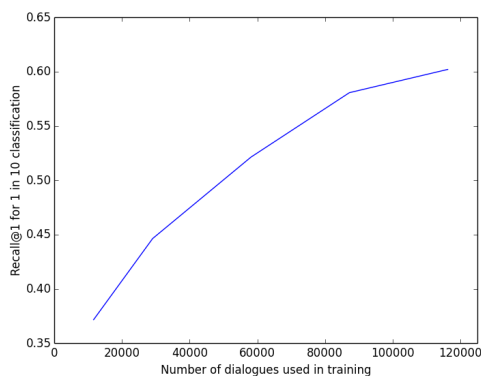
Figure 3: The LSTM (with 200 hidden units), showing Recall@1 for the 1 in 10 classification, with increasing dataset sizes.

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 [6]. 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 the subject of more formal evaluation.

6.2 Altering Test Set Difficulty

One of the interesting properties of the response selection task is the ability 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, we will consider selecting false responses that are similar to the actual response (e.g. as measured by 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.

6.3 State Tracking and Utterance Generation

The work described here focuses on the task of response selection. This can be seen as an intermediate step between slot filling and utterance generation. In slot filling, the set of candidate outputs (*states*) is identified *a priori* through knowledge engineering, and is typically smaller than the set of responses considered in our work. When the set of candidate responses is close to the size of the dataset (e.g. all utterances ever recorded), then we are quite close to the response generation case.

There are several reasons not to proceed directly to response generation. First, it is likely that current algorithms are not yet able to generate good results for this task, and it is preferable to tackle metrics for which we can make progress. Second, we do not yet have a suitable metric for evaluating performance in the response generation case. One option is to use the BLEU [18] or METEOR [16] scores from machine translation. However, using BLEU to evaluate dialogue systems has been shown to give extremely low scores [28], due to the large space of potential sensible responses [7]. Further, since the BLEU score is calculated using N-grams [18], it would provide a very low score for reasonable responses that do not have any words in common with the ground-truth next utterance.

Alternatively, one could measure the difference between the generated utterance and the actual sentence by comparing their representations in some embedding (or *semantic*) space. However, different models inevitably use different embeddings, necessitating a standardized embedding for evaluation purposes. Such a standardized embeddings has yet to be created.

Another possibility is to use human subjects to score automatically generated responses, but time and expense make this a highly impractical option.

In summary, while it is possible that current language models have outgrown the use of slot filling as a metric, we are currently unable to measure their ability in next utterance generation in a standardized, meaningful and inexpensive way. This motivates our choice of response selection as a useful metric for the time being.

Acknowledgments

References

[1] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[2] A. Bordes, J. Weston, and N. Usunier. Open question answering with weakly supervised embedding models. In *MLKDD*, pages 165–180. Springer, 2014.

[3] J. Boyd-Graber, B. Satinoff, H. He, and H. Daume. Besting the quiz master: Crowdsourcing incremental classification games. In *EMNLP*, 2012.

[4] 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. *arXiv preprint arXiv:1406.1078*, 2014.

[5] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[6] M. Elsner and E. Charniak. You talking to me? a corpus and algorithm for conversation disentanglement. In *ACL*, pages 834–842, 2008.

[7] 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.

[8] J.J. Godfrey, E.C. Holliman, and J. Mc-Daniel. Switchboard: Telephone speech corpus for research and development. In *ICASSP*, 1992.

[9] M. Henderson, B. Thomson, and J. Williams. Dialog state tracking challenge 2 & 3, 2014.

[10] M. Henderson, B. Thomson, and J. Williams. The second dialog state tracking challenge. In *SIGDIAL*, page 263, 2014.

[11] M. Henderson, B. Thomson, and S. Young. Word-based dialog state tracking with recurrent neural networks. In *SIGDIAL*, page 292, 2014.

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

[13] Dialog state tracking challenge 4.

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

[15] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[16] A. Lavie and M.J. Denkowski. The METEOR metric for automatic evaluation of Machine Translation. *Machine Translation*, 23(2-3):105–115, 2009.

[17] L.R. Medsker and L.C. Jain. Recurrent neural networks. *Design and Applications*, 2001.

[18] K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[19] J. Pennington, R. Socher, and C.D. Manning. GloVe: Global Vectors for Word Representation. In *EMNLP*, 2014.

[20] J. Ramos. Using tf-idf to determine word relevance in document queries. In *ICML*, 2003.

[21] A. Ritter, C. Cherry, and W. Dolan. Unsupervised modeling of twitter conversations. 2010.

[22] A. Ritter, C. Cherry, and W. Dolan. Data-driven response generation in social media. In *EMNLP*, pages 583–593, 2011.

[23] 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.

[24] J. Schatzmann, K. Georgila, and S. Young. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *SIGDIAL*, 2005.

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

[26] B. A. Shawar and E. Atwell. Chatbots: are they really useful? In *LDV Forum*, volume 22, pages 29–49, 2007.

[27] 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.

[28] 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. 2015.

[29] D.C. Uthus and D.W. Aha. Extending word highlighting in multiparticipant chat. Technical report, DTIC Document, 2013.

[30] D.C. Uthus and D.W Aha. The ubuntu chat corpus for multiparticipant chat analysis. In *AAAI Spring Symposium on Analyzing Microtext*, pages 99–102, 2013.

[31] H. Wang, Z. Lu, H. Li, and E. Chen. A dataset for research on short-text conversations. In *EMNLP*, 2013.

[32] J. Williams, A. Raux, D. Ramachandran, and A. Black. The dialog state tracking challenge. In *SIGDIAL*, pages 404–413, 2013.

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

[34] M.D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Appendix A: Dialogue excerpts

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 4: Example chat room conversation from the #ubuntu channel of the Ubuntu Chat Logs (top), with the disentangled conversations for the Ubuntu Dialogue Corpus (bottom).

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 5: Example of before (top box) and after (bottom box) 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.