Hierarchical Neural Network Generative Models for Movie Dialogues

Iulian V. Serban[1], Alessandro Sordoni[1], Yoshua Bengio[1,3], Aaron Courville[1] and Joelle Pineau[2]

[1]Department of Computer Science and Operations Research, Université de Montréal, Montreal, Canada
[2]School of Computer Science, McGill University, Montreal, Canada
[3]CIFAR Senior Fellow

Abstract

We consider the task of generative dialogue modeling for movie scripts. To this end, we extend the recently proposed hierarchical recurrent encoder decoder neural network and demonstrate that this model is competitive with state-of-the-art neural language models and backoff n-gram models. We show that its performance can be improved considerably by bootstrapping the learning from a larger question-answer pair corpus and from pretrained word embeddings.

1 Introduction

Dialogue systems, also known as interactive conversational agents, virtual agents and sometimes chatterbots, are used in a wide set of applications ranging from technical support services to language learning tools and entertainment (Young et al., 2013; Shawar and Atwell, 2007). Dialogue systems can be divided into goal driven systems, such as technical support services, and non-goal driven systems, such as language learning tools or computer game characters.

Perhaps the most successful approach to goal driven systems has been to view the dialogue problem as a partially observable Markov decision process (POMDP) (Young et al., 2013; Pieraccini et al., 2009). Unfortunately, most deployed dialogue systems use hand-crafted features for the state and action space representations, and require either a large annotated task-specific corpus or a horde of human subjects willing to interact with the unfinished system. This not only makes it expensive and time-consuming to deploy a real dialogue system, but also limits its usage to a narrow domain. Recent work has tried to push goal driven systems towards learning the observed features themselves with neural network models (Henderson et al., 2013; Henderson et al., 2014), yet such approaches still require large corpora of annotated task-specific simulated conversations.

On the other end of the spectrum are the non-goal driven systems (Ritter et al., 2011; Banchs and Li, 2012; Ameixa et al., 2014; Nio et al., 2014). Most recently Sordoni et al. (2015b) and Shang et al. (2015) have drawn inspiration from the use of neural networks in natural language modeling and machine translation tasks (Cho et al., 2014b; Sutskever et al., 2014). There are two motivations for developing non-goal driven systems. Firstly, they may be deployed directly for tasks which do not naturally exhibit or require a quantifiable goal (e.g. language learning) or simply for entertainment. Secondly, if they are trained on corpora related to the task of a goal-driven dialogue system (e.g. corpora which cover conversations on similar topics) then these models can be used to train a user simulator, which can then train the POMDP models discussed earlier (Young et al., 2013; Pietquin and Hastie, 2013; Levin et al., 2000). This would alleviate the expensive and time-consuming task of constructing a large-scale task-specific dialogue corpus. In addition to this, the features extracted from the non-goal driven systems may be used to expand the state space representation of POMDP models (Singh et al., 2002). This will help generalization to dialogues outside the annotated task-specific corpora.

Our contribution is in the direction of non-goal driven systems and generative probabilistic models that do not require hand-crafted features. We define the generative dialogue problem as modeling the utterances and interactive structure of the dialogue, including turn taking and pauses. Without loss of generality, as a stepping stone and to be comparable to related work, we restrict our experiments to triples, i.e. three consecutive utterances in a dialogue. We focus on models, which scale to long conversations.

We experiment with the well-established recurrent neural networks (RNN) and n-gram models. In particular, we adopt the hierarchical recurrent encoder decoder (HRED) proposed by Sordoni et al. (2015a) and demonstrate that it is competitive with all other models in the literature. We extend the model with architectural changes to better suit the dialogue task and show that this improves its ability to predict semantic and topical content. We show that performance can be improved significantly by bootstrapping from pretrained word embeddings and from pretraining the model on a larger question-answer pair (Q-A) corpus.

To carry out experiments, we introduce the *MovieTriples* dataset based on movie scripts. Movie scripts span a wide range of topics and contain long dialogues with few participants, making them ideal for researching open domain, long interaction dialogue systems. They are close to human spoken language (Forchini, 2009), which makes them suitable for bootstrapping goal-driven dialogue systems.

2 Models

We consider a dialogue as a sequence of M utterances $D = \{U_1, \ldots, U_M\}$ involving two interlocutors. Each U_m contains a sequence of N_m tokens, i.e. $U_m = \{w_{m,1}, \ldots, w_{m,N_m}\}$, where $w_{m,n}$ is a random variable taking values in the vocabulary V and representing the token at position n. The tokens represent both words and *dialogue acts*, e.g. end of a turn and pause tokens. A generative model of dialogue parameterizes a probability distribution P - governed by parameters θ - over the set of all possible dialogues of arbitrary lengths. Under P_θ, the probability of a dialogue D can be written as:

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

where $U_{<m} = \{U_1, \ldots, U_{m-1}\}$ and $w_{m,<n} = \{w_{m,1}, \ldots, w_{m,n-1}\}$, i.e. the tokens preceding n in the utterance U_m. Computing joint probabilities over dialogues, for any realistic vocabulary size, suffers from the curse of dimensionality, i.e. it is extremely unlikely that a new dialogue will be identical to a dialogue in the training set. It is also intractable, since sampling naively must consider an exponential number of combinations.

The task is analogous to language modeling. Here, the classical approach based on n-gram contexts do not share statistical weights and disregard possible semantic commonalities. Bengio et al. (2003) first proposed to tackle this problem by using a distributed (dense) vector representation of words, also called *embeddings*. By means of such distributed representations, the recurrent neural network (RNN) based language model (Mikolov et al., 2010) has pushed state-of-the-art performance by learning long n-gram contexts while avoiding data sparsity issues. Overall, RNNs have performed well on a variety of NLP tasks such as machine translation (Cho et al., 2014b; Sutskever et al., 2014; Bahdanau et al., 2015) and information retrieval (Sordoni et al., 2015a).

2.1 Recurrent Neural Network

A recurrent neural network (RNN) models an input sequence of tokens $\{w_1, \ldots, w_N\}$ by computing the following recurrence:

$$h_n = f(h_{n-1}, w_n), \tag{2}$$

where $h_n \in \mathbb{R}^{d_h}$ is called a recurrent, or *hidden*, state and acts as a compact summary of tokens, and their order, seen up to position n. After running through the sequence, the recurrent states $h_1, \ldots, h_N$ can be used in various ways. The last state h_N may be viewed as an order-sensitive compact summary of the tokens. In language modeling tasks, the context information encoded in h_n is used to predict the next token in the sentence. Formally:

$$P_\theta(w_{n+1} = v | w_{\leq n}) = \frac{\exp\left(g(h_n, v)\right)}{\sum_{v'} \exp\left(g(h_n, v')\right)}.$$

The functions f and g are typically defined as:

$$f(h_{n-1}, w_n) = \tanh\left(H h_{n-1} + I_{w_n}\right), \tag{3}$$

$$g(h_n, v) = O_{w_n}^T h_n, \tag{4}$$

The matrix $I \in \mathbb{R}^{d_h \times |V|}$ contains the input *word embeddings*, i.e. each column I_j is a vector corresponding to token j in the vocabulary V. Due to the size of the model vocabulary V, it is common to approximate the I matrix with a low-rank decomposition, i.e. $I = XE$, where $X \in \mathbb{R}^{d_h \times d_e}$ and $E \in \mathbb{R}^{d_e \times |V|}$, and $d_e < d_h$. This approach has also the advantage that the embedding matrix E may separately be bootstrapped (e.g. learned) from larger corpora and used as a shared parameter in complex neural models composed of several RNNs submodules.

Analogously, the matrix $O \in \mathbb{R}^{d_h \times |V|}$ represents the output word embeddings, where each possible next token is projected into another dense vector and compared to the hidden state h_n. The probability of seeing token v at position $n+1$ increases if its corresponding embedding vector O_v is "near" the context vector h_n.

RNN language models are commonly trained by maximizing the log-likelihood of the parameters on a training set using stochastic gradient descent methods. However, for long sequences, this can be problematic due to either vanishing or exploding gradients (Bengio et al., 1994). To tackle these issues, *gated* variants of the transition function f have been proposed, including the LSTM unit (Hochreiter and Schmidhuber, 1997) and GRU unit (Cho et al., 2014b). GRUs are competitive with LSTMs in performance, but are computationally cheaper (Greff et al., 2015). Formally, a GRU parametrizes f as:

$$\begin{aligned} g_r &= \sigma(I_{r,w_n} + H_r h_{n-1}) \\ g_u &= \sigma(I_{u,w_n} + H_u h_{n-1}) \\ \tilde{h} &= \tanh(I_{w_n} + H(g_r \cdot h_{n-1})) \\ f(h_{n-1}, w_n) &= g_u \cdot h_{n-1} + (1 - g_u) \cdot \tilde{h}, \end{aligned} \tag{5}$$

where σ is the sigmoid, $\sigma(x) \in [0,1]$, $\cdot$ is the element-wise multiplication and $I_r, I_u, I \in \mathbb{R}^{d_h \times |V|}$, $H_r, H_u, H \in \mathbb{R}^{d_h \times d_h}$ are the parameters of the GRU. The vector g_r is called *reset gate*, g_u the *update gate* and $\tilde{h}$ the *candidate activation*. By adjusting g_r and g_u appropriately, the model is able to create linear *skip-connections* between distant hidden states, which in turn makes the credit assignment problem easier and makes the gradient signal stronger to earlier hidden states.

Generative dialogue modeling with RNNs can be achieved by taking the concatenated utterances as a single sequence of tokens and predicting each token conditioned on previous tokens. However, this naïve approach introduces a large number of long-range dependencies which are difficult to capture, even with complex transition functions such as GRUs (Cho et al., 2014a; Bahdanau et al., 2015). This motivates the next section.

2.2 Hierarchical Recurrent Encoder Decoder

Our work extends on the hierarchical recurrent encoder decoder architecture (HRED) proposed by Sordoni et al. (2015a) for web query suggestion, which itself builds on the encoder decoder architecture proposed by Cho et al. (2014b) and is closely related to the architectures proposed by El et al. (1995) and Koutnik et al. (2014).

In their framework, HRED predicts the next query given the queries already submitted by the user. The history of past submitted queries is considered as a sequence at two levels: a sequence of words for each web query and a sequence of queries. HRED models this hierarchy of sequences with two RNNs: one at the word level and one at the query level. A similar situation holds for dialogue: a dialogue can be seen as a sequence of utterances which, in turn, are sequences of tokens. A pictorial representation of HRED for dialogue is presented in Figure 1.

In dialogue, the *encoder* RNN maps each utterance to an utterance vector. The utterance vector is the hidden state obtained after the last token of the utterance has been processed. The higher-level *context* RNN keeps track of past utterances by processing iteratively each utterance vector. After processing the utterance U_m, the hidden state of the context RNN represents a summary of the dialogue up to turn m which is used to predict the next utterance U_{m+1}. The next utterance prediction is performed by means of a *decoder* RNN, which takes the hidden state of the context RNN and produces a probability distribution over the tokens in the next utterance. The *decoder* RNN is similar to the RNN language model, but with the important difference that the prediction is conditioned on the hidden state of the context RNN.

The encoder, context and decoder RNNs all make use of the GRU hidden unit along their temporal dimensions. Everywhere else we use the hyperbolic tangent as activation function. To help generalization, it is also possible to use the maxout activation function between the hidden state and the projected word embeddings of the decoder RNN (Goodfellow et al., 2013).

The HRED architecture allows information to flow more easily over very long sequences compared to a standard RNN. Consider the example in Figure 1. For the RNN trained on concatenated dialogue utterances, the distance between the token *pass* in the third utterance and *feel* in the first

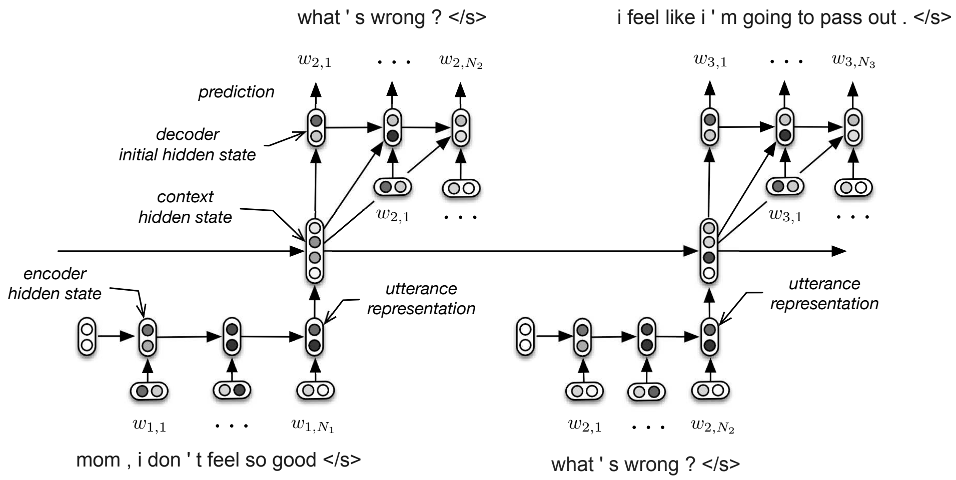

Figure 1: The HRED architecture for a dialogue composed of three turns. Each utterance is encoded into a dense vector and then mapped into the dialogue context, which is used to decode (generate) the tokens in the next utterance. The encoder RNN keeps track of the order of tokens appearing within the utterance, and the context RNN models the temporal structure of the utterances appearing so far in the dialogue, allowing information and gradients to flow over longer time spans. The decoder predicts one token at a time using a RNN. Adapted with permission from Sordoni et al. (2015a).

utterance would be $d = 18$ tokens. The gradient would have to flow d step backwards in order to capture the dependency between the two terms. In HRED, the distance is reduced to $d = 7$, as the context RNN is updated only once per utterance. In HRED, the shortest path between two tokens occurring at positions n_1 and $n_2 > n_1$ is on the order of $(n_2 - n_1)/U$, where U is the average length of an utterance, while in a regular RNN the shortest path is given by $n_2 - n_1$. We believe reducing this distance is a crucial property for neural models to scale to long dialogues.

2.3 Bidirectional HRED

In HRED, the utterance representation is given by the last hidden state of the encoder RNN. This architecture worked well for web queries, but may be insufficient for dialogue utterances, which are longer and contain more syntactic articulations than web queries. For long utterances, the last state of the encoder RNN may not reflect important information seen at the beginning of the utterance. Thus, we propose to extend the HRED architecture with additional representational capacity on the encoder component. We choose to model the utterance encoder with a *bidirectional* RNN, which proved useful to Bahdanau et al. (2015) for machine translation. Bidirectional RNNs run two chains: one forward through the utterance tokens and another backward, i.e. reversing the tokens in the utterance. Hence, the forward hidden state at position n summarizes tokens preceding position n and the backwards hidden state summarizes tokens following position n[1]. To obtain a fixed-length representation for the utterance, we summarize the information in the hidden states by: 1) applying L_2 pooling over the temporal dimension of each chain, and taking the concatenation of the two pooled states as input to the context RNN, or 2) taking the concatenation of the last state of each RNN as input to the context RNN. The RNN running in reverse will effectively also introduce additional short term dependencies, which has proven useful in similar architectures (Sutskever et al., 2014). We refer to this variant as HRED-Bidirectional.

2.4 Bootstrapping From Word Embeddings

The commonsense knowledge that the dialogue interlocutors share may be difficult to infer if the dataset is not sufficiently large. Therefore, our models may be significantly improved by learning word embeddings from larger corpora. This has been beneficial for classification of user intents (Forgues et al., 2014). We choose to initialize our word embeddings E with Word2Vec[2] (Mikolov et al., 2013) trained on the Google News dataset con-

[1]Note that the bidirectional RNN is always one utterance behind the decoder RNN.

[2]`http://code.google.com/p/word2vec/`

taining about 100 billion words. The sheer size of the dataset ensures that the embeddings contain rich semantic information about each word.

2.5 Bootstrapping From Subtitles Q-A

Bootstrapping word embeddings will not affect the other model parameters, which will still rely on the original dialogue corpus for training. To learn a good initialization point for these other parameters, we may pretrain the model on a large non-dialogue corpus, which covers similar topics and types of interactions between interlocutors. One such corpus is the Q-A *SubTle* corpus containing about 5.5M Q-A pairs constructed from movie subtitles. Now, we construct an artificial dialogue dataset by taking each $\{Q, A\}$ pair as a two-turn dialogue $D = \{U_1 = Q, U_2 = A\}$. Since there are two utterances in each example, all the model parameters will be updated during training. However, because these examples are short, the higher-level context RNN may not be initialized to a very useful point for the HRED models.

3 Related Work

Modeling conversations on micro-blogging websites with generative probabilistic models was first proposed by Ritter et al. (2011). They view the response generation problem as a translation problem, where a post needs to be translated into a response. Generating responses was considerably more difficult than translating between languages, which was attributed to the wide range of plausible responses and the lack of alignment on words and phrases between the post and the response. In particular, they found that the statistical machine translation approach was superior to the information retrieval approach. In the same vein, Shang et al. (2015) proposed to use the neural network encoder-decoder framework for generating responses on the micro-blogging website *Weibo*. They also formulated the problem as conditional generation, where given a post, the model generates a response. Unfortunately, this architecture scales linearly with the number of dialogue turns.

A way to consider the conversation context was proposed by Sordoni et al. (2015b) to generate responses for posts on *Twitter*. They concatenated three consecutive Twitter messages, representing a short conversation between two users, and defined the problem as predicting each word in the conversation given all preceding words. They encoded a bag-of-words context representation with a multilayer neural network and generated a response with a standard RNN. They then combined their generative model with a machine translation system, and showed that the hybrid system outperformed the machine translation system proposed by Ritter et al. (2011).

To the best of our knowledge, Banchs et al. (2012) were the first to suggest using movie scripts to build dialogue systems. They constructed an information retrieval system based on the vector space model. Conditioned on one or more utterances, their model searches a database of movie scripts and retrieves an appropriate response. Using another information retrieval system, Ameixa et al. (2014) used movie subtitles to train a dialogue system. They showed that an existing dialogue system could successfully be augmented with the subtitles, such that, when its response confidence is low, it will search an appropriate answer from the subtitle corpus. This helped answer out-of-domain questions.

4 Dataset

The *MovieTriples* dataset has been developed by expanding and preprocessing the *Movie-DiC* dataset by Banchs et al. (2012) to make it fit the generative dialogue modeling framework[3]. Based on a literature review, we found that the *Movie-DiC* was the largest dataset available containing all consecutive utterances from movies. Other datasets in the literature include the corpora by Walker et al. (2012a), Roy et al. (2014), and the unpublished Cornell Movie Dialogue Corpus[4].

Compared to similar-sized domain-specific datasets (Uthus and Aha, 2013; Walker et al., 2012b), movie scripts span a wide range of topics, which makes them ideal for investigating semantic understanding of dialogue models. Contrary to micro-blogging websites, such as Twitter (Ritter et al., 2010), movie scripts contain long dialogues with few participants. This makes them well-suited for modeling long-term interactions. They also contain relatively few spelling mistakes and acronyms, which previously made research on micro-blogging websites difficult. Employing movie scripts also makes it possible to enrich dialogue systems with additional contextual information, such as action descriptions, summaries and genre labels. Movie scripts are close in nature to

[3] The dataset is made available upon request.

[4] `http://www.mpi-sws.org/~cristian/Cornell_Movie-Dialogs_Corpus.html`

	Training	Validation	Test
Movies	484	65	66
Triples	196,308	24,717	24,271
Avg. tokens/triple	53	53	55
Avg. unk/triple	0.97	1.22	1.19

Table 1: Statistics of the *MovieTriples* dataset.

human spoken conversations (Forchini, 2009). As noted by Forchini (2009): *"movie language can be regarded as a potential source for teaching and learning spoken language features"*. Hence, we argue that bootstrapping a goal-driven spoken dialogue system based on movie scripts can improve performance.

4.1 Extraction And Preprocessing

We expanded the dataset to include meta-information for each movie, extracted through the online API service OMDBAPI[5]. We then processed the dataset to remove duplicate manuscripts. Afterwards, a spelling corrector based on Wikipedia's most common English spelling mistakes was applied[6]. We then implemented a set of simple regular expressions to remove double punctuation marks and spacings. We used the python-based natural language toolkit NLTK (Bird et al., 2009) to perform tokenization and named-entity recognition[7]. All names and numbers were replaced with the <*person*> and <*name*> tokens respectively. Numbers were replaced with the <*number*> token. The use of placeholders allows to measure performance w.r.t. the abstract semantic and syntactic structure of dialogues, as opposed to recalling exact names and numbers. Similar preprocessing has been applied in previous work (Ritter et al., 2010; Nio et al., 2014). To reduce data sparsity further, all tokens were finally transformed to lowercase letters, and all but the 10,000 most frequent tokens were replaced with the <*unk*> token representing unknown or out-of-vocabulary words.

4.2 Triples Construction

The atomic entry of the *MovieTriples* is a "triple" $\{U_1, U_2, U_3\}$, i.e. a dialogue of three turns occurring between two interlocutors A and B for which A emits the first utterance U_1, B responds by U_2 and A finally responds with the last utterance U_3. This is similar to previous work (Sordoni et al., 2015b). Unlike conversations extracted from internet relayed chat (IRC) (Elsner and Charniak, 2008), the majority of movie scenes only contain a single dialogue thread, which means that nearly all extracted triples constitute a continuous dialogue segment between the active speakers.

To capture the interactive dialogue structure, a special end-of-utterance token is appended to all utterances. If the same speaker makes a break in an utterance and then continues again, we add a special continued-utterance token. All models must learn to predict these dialogue act tokens.

To avoid co-dependencies between triples coming from the same movie, we first split the movies into training, validation and test set, and then construct the triples. This will ensure that our results generalize to new domains. The dataset contains about 13M words in total and about 10M words in the training set. Statistics are reported in Table 1.

5 Experiments

5.1 Baselines

We test our models against state-of-the-art neural network and non-neural network baselines. First, we compare our models to well-established n-gram models (Goodman, 2001). To compare to a neural network baseline, we train a RNN on the concatenation of the utterances in each triple. We also report results obtained by the context-sensitive model (DCGM-I) recently proposed by Sordoni et al. (2015b).

5.2 Evaluation Metrics

Accurate evaluation of a non-goal driven dialogue system is an open problem (Galley et al., 2015; Pietquin and Hastie, 2013; Schatzmann et al., 2005). There is no well-established method for automatic evaluation, and human-based evaluation is expensive. Nevertheless, for probabilistic language models word perplexity is a well-established performance metric (Bengio et al., 2003; Mikolov et al., 2010), and has been suggested for generative dialogue models previously (Pietquin and Hastie, 2013):

$$\exp\left(-\frac{1}{N_W}\sum_{n=1}^{N}\log P_\theta(U_1^n, U_2^n, U_3^n)\right), \quad (6)$$

[5] http://www.omdbapi.com

[6] Retrieved on February 20th, 2015: http://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings

[7] NLTK uses a maximum entropy chunker trained on the ACE corpus: http://catalog.ldc.upenn.edu/LDC2005T09

for a model with parameters θ, dataset with N triples $\{U_1^n, U_2^n, U_3^n\}_{n=1}^N$, and N_W the number of tokens in the entire dataset. The lower the perplexity, the better the model is assumed to be. Unlike linguistic performance metrics, word perplexity explicitly measures the model's ability to account for the syntactic structure of the dialogue (e.g. turn-taking) and the syntactic structure of each utterance (e.g. punctuation marks). In dialogue, the distribution over the words in the next utterance is highly multi-modal, e.g. there are many possible answers, which makes perplexity particularly appropriate because it will always measure the probability of regenerating the exact reference utterance.

Although perplexity is an established measure for generative models, in the dialogue setting, utterances may be overwhelmed by many common words especially arising from colloquial or informal exchanges. To focus the perplexity metric on deeper semantic content (e.g. the dialogue topic), we propose to also use a reweighed perplexity metric. That is, the perplexity metric applied to all words in the dataset except for a small set of stop words, which instead is assumed to have been predicted correctly and excluded from the denominator in the first fraction of eq. (6). The set of stop words contains 77 English pronouns[8], all punctuation marks, the unknown word token and the end-of-utterance token, which constitute 48.37% of the training set.

5.3 Training Procedure

To train the neural network models, we optimized the log-likelihood of the triples using the recently proposed Adam optimizer (Kingma and Ba, 2014). Our implementation relies on the open-source Theano library (Bastien et al., 2012). The best hyperparameters of the models were chosen by early stopping with patience on the validation set perplexity (Bengio, 2012). For the baseline RNN, we tested hidden state spaces $d_h = 200$, 300 and 400, and found that 400 yielded best performance. For HRED we experimented with encoder and decoder hidden state spaces of size 200, 300 and 400. Increasing these two state space improved performance consistently, but due to GPU memory limitations we limited ourselves to size 300 when not bootstrapping or bootstrapping from Word2Vec, and to 400 when bootstrapping from *SubTle*. Preliminary experiments showed that the context RNN state space at and above 300 performed similarly, so we fixed it at 300 when not bootstrapping or bootstrapping from Word2Vec, and to 1200 when bootstrapping from *SubTle*. To help generalization, we used the maxout activation function when not bootstrapping and when bootstrapping from Word2Vec.

[8] http://www.esldesk.com/vocabulary/pronouns

Bootstrapping Word Embeddings Our embedding matrix E is initialized using the publicly available 300 dim. Word2Vec embeddings trained on the Google News corpus. Certain words in the movie scripts vocabulary could not be directly matched to the Word2Vec embeddings. These words (0.15% of the training set tokens), along with dialogue act and placeholder tokens, were initialized randomly. All dimensions were rescaled to have mean zero and standard deviation 0.01. The training procedure is unfolded into two stages. In the first stage, we trained each neural model with fixed Word2Vec embeddings. During this stage, we also trained the dialogue act and placeholder tokens, together with tokens not covered by the original Word2Vec embeddings. Training the dialogue act tokens from the dialogue corpus allows the model to learn the interaction structure of the dialogue. In the second stage, we trained all parameters of each neural model until convergence. We used L_2 pooling for the HRED models, since it appeared to perform slightly better under this setup.

Bootstrapping SubTle We processed the *SubTle* corpus following the same procedure used for *MovieTriples*, but now treating the last utterance U_3 as empty. The final *SubTle* corpus contained 5,503,741 Q-A pairs, and a total of 93,320,500 tokens. Although *SubTle* was extracted from subtitles, and *MovieTriples* from movie scripts, we found no significant utterance overlap. Manual inspection showed that overlapping utterances consisted mainly of very common short phrases, e.g. *are you okay ?* or *so what ?*. When bootstrapping from the *SubTle* corpus, we found that all models performed slightly better when randomly initializing and learning the word embeddings from *SubTle* compared to fixing the word embeddings to those given by Word2Vec. We did not use L_2 pooling when bootstrapping from *SubTle*, since it appeared to perform slightly worse. We speculate that its regularization effect is unnecessary here.

Model	Perplexity	Perplexity@U_3	Error-Rate	Error-Rate@U_3
Backoff N-Gram	64.89	65.05	-	-
Modified Kneser-Ney	60.11	54.75	-	-
Absolute Discounting N-Gram	56.98	57.06	-	-
Witten-Bell Discounting N-Gram	53.30	53.34	-	-
RNN	35.63 ± 0.16	35.30 ± 0.22	$66.34\% \pm 0.06$	$66.32\% \pm 0.08$
DCGM-I	36.10 ± 0.17	36.14 ± 0.26	$66.44\% \pm 0.06$	$66.57\% \pm 0.10$
HRED	36.59 ± 0.19	36.26 ± 0.29	$66.32\% \pm 0.06$	$66.32\% \pm 0.11$
HRED + Word2Vec	33.95 ± 0.16	33.62 ± 0.25	$66.06\% \pm 0.06$	$66.05\% \pm 0.09$
HRED + SubTle	27.14 ± 0.12	26.60 ± 0.19	$64.10\% \pm 0.06$	$64.03\% \pm 0.10$
HRED-Bi. + SubTle	$\mathbf{26.81 \pm 0.11}$	$\mathbf{26.31 \pm 0.19}$	$\mathbf{63.93\% \pm 0.06}$	$\mathbf{63.91\% \pm 0.09}$

Table 2: Test set word perplexity results computed on $\{U_1, U_2, U_3\}$ and solely on $\{U_3\}$ conditioned on $\{U_1, U_2\}$. Standard deviations are shown for all neural models. Best performances are marked in bold.

	All Tokens				Excluding Stop Words	
Model	Perplexity	Perplex.@U_3	Error-Rate	Error-Rate@U_3	Perplexity	Perplex.@U_3
RNN	27.09 ± 0.13	26.67 ± 0.19	$64.10\% \pm 0.06$	$64.07\% \pm 0.10$	$\mathbf{75.34 \pm 0.47}$	73.24 ± 0.76
HRED	27.14 ± 0.12	26.60 ± 0.19	$64.10\% \pm 0.06$	$64.03\% \pm 0.10$	77.17 ± 0.42	74.41 ± 0.66
HRED-Bi.	$\mathbf{26.81 \pm 0.11}$	$\mathbf{26.31 \pm 0.19}$	$\mathbf{63.93\% \pm 0.06}$	$\mathbf{63.91\% \pm 0.09}$	75.71 ± 0.41	$\mathbf{73.24 \pm 0.64}$

Table 3: Test set word perplexity and classification error on $\{U_1, U_2, U_3\}$ and $\{U_3\}$ when bootstrapping from *SubTle* corpus. We also report word perplexities with stop words removed. Standard deviations are shown for all metrics. Best performances (before rounding) are marked in bold.

The HRED models were pretrained for approximatively four epochs on the *SubTle* dataset. Longer training did not appear to improve performance for any of the models. Then, we fine-tuned the pretrained models on the *MovieTriples* dataset holding the word embeddings fixed, since we found non-significant difference also fine-tuning these.

5.4 Empirical Results

Our results are summarized in Table 2. All neural models beat state-of-the-art n-grams models w.r.t. both word perplexity and word classification error (comparing the most likely predicted word with the actual one). Without bootstrapping, the RNN model performs similarly to the more complex DCGM-I and HRED models. This can be explained by the size of the dataset, which makes it easy for the HRED and DCGM-I model to overfit. The last three lines of Table 2 show that bootstrapping the model parameters from a large non-dialogue corpus achieves significant gains in both measures. Bootstrapping from *SubTle* is particularly useful since it allows a gain of nearly 10 perplexity points compared to the HRED model without bootstrapping. We believe that this is because it trains all model parameters, unlike bootstrapping from Word2Vec.

In Table 3, we report the results of the standard RNN and HRED models when bootstrapped from *SubTle* corpus. The gains due to architectural choice are naturally smaller than those obtained by bootstrapping, because we are in a regime of relatively little training data compared to other natural language processing tasks, such as machine translation, and hence we would expect the differences to grow with more training data and longer dialogues. The largest gains are obtained by the proposed HRED-Bidirectional architecture, which on five out of the six metrics outperform both the standard HRED and RNN model. The perplexity metrics computed excluding stop words demonstrate that the HRED-Bidirectional model outperforms the standard HRED model in capturing semantic and topic-specific information. The bidirectional structure appears to capture and retain information from the U_1 and U_2 utterances better than either the RNN and the original HRED model. This confirms our earlier hypothesis, and demonstrates the potential of HRED as a solution for modeling long dialogues.

5.5 MAP Outputs

We evaluate the use of beam-search for RNNs (Graves, 2012) to approximate the most probable (MAP) utterance U_3, given the first two utterances, U_1 and U_2. MAP outputs are shown in Table 5.5 for HRED-Bidirectional bootstrapped from *SubTle* corpus. As shown in the table, the model often produces sensible

Reference ($\mathbf{U_1}$, $\mathbf{U_2}$)	**MAP**	**Target** ($\mathbf{U_3}$)
U_1: yeah , okay . U_2: well , i guess i ’ ll be going now .	i ’ ll see you tomorrow .	yeah .
U_1: oh . <continued_utterance> oh . U_2: what ’ s the matter , honey ?	i don ’ t know .	oh .
U_1: it ’ s the cheapest . U_2: then it ’ s the worst kind ?	no , it ’ s not .	they ’ re all good , sir .
U_1: <person> ! what are you doing ? U_2: shut up ! c ’ mon .	what are you doing here ?	what are you that crazy ?

Table 4: MAP outputs for HRED-Bidirectional bootstrapped from *SubTle* corpus.

answers. However, the majority of the predictions are generic, such as *I don’t know* or *I’m sorry*[9]. We observed the same phenomenon for the RNN model. This appears to be a recurring observation in the literature (Sordoni et al., 2015b; Vinyals and Le, 2015)[10]. However, to the best of our knowledge, we are the first to emphasize and discuss it in details.

There are several possible explanations for this behavior. Firstly, due to data scarcity, the model may only learn to predict the most frequent utterances. Since dialogue is inherently ambiguous and multi-modal, predicting dialogues accurately would require more data than other natural language processing tasks. Secondly, the majority of dialogue tokens consist of punctuation marks and pronouns. Since every token is weighted equally during training, the gradient signal of the neural network will be dominated by these punctuation and pronoun tokens. This makes it hard for the neural network to learn topic-specific embeddings and even harder to predict diverse utterances. This suggest exploring neural architectures which explicitly separate semantic structure from syntactic structure. Finally, the context of a triple may be too short. In that case, the models should benefit from longer contexts and by conditioning on other information sources, such as semantic and visual information.

An important implication of this observation is that metrics based on MAP outputs (e.g. cosine similarity, BLEU, Levenshtein distance) will primarily favour models that output the same number of punctuation marks and pronouns as are in the test utterances, as opposed to matching semantic content (e.g. nouns and verbs). This would be systematically biased and not necessarily in any way correlate with the objective of producing appropriate responses.

[9]This behavior did not occur when we generated stochastic samples. In fact, these samples contained a large variety of topic-specific words and often appeared to maintain the topic of the conversation.

[10]Our work was carried out independently from that of Vinyals et al. (2015).

6 Conclusion and Future work

The main contributions of this paper are the following. We have demonstrated that a hierarchical recurrent network generative model can outperform both n-gram based models and baseline neural network models on the task of predicting the next utterance and dialogue acts in a dialogue. To this end, we introduced a novel dataset called *MovieTriples* based on movie scripts, which is suitable for modeling long, open domain dialogues close to human spoken language. In addition to the recurrent hierarchical architecture, we found two crucial ingredients: the use of a large external monologue corpus to initialize the word embeddings, and the use of a large related, but non-dialogue, corpus in order to *pretrain* the recurrent net. This points to the need for larger dialogue datasets.

Future work should study full length dialogues, as opposed to triples, and model other dialogue acts, such as interlocutors entering or leaving the dialogue and executing actions. It should focus on bootstrapping from other, large non-dialogue corpora, as well as expand *MovieTriples* to include other movie script corpora. Finally, our analysis of the model MAP outputs suggest that it would be beneficial to include longer and additional context, including other modalities such as video, and that MAP based evaluation metrics are inappropriate when the outputs are generic in nature.

Acknowledgements: The authors acknowledge NSERC, Canada Research Chairs, CIFAR and Compute Canada for funding. The authors thank Rafael Banchs and Luisa Coheur for providing the *Movie-DiC* and *SubTle* corpora, as well as Nissan Pow, Ryan Lowe and Laurent Charlin for helpful feedback.

References

David Ameixa, Luisa Coheur, Pedro Fialho, and Paulo Quaresma. 2014. Luke, i am your father: dealing with out-of-domain requests by using movies subtitles. In *Intelligent Virtual Agents*, pages 13–21. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR 2015)*.

Rafael E Banchs and Haizhou Li. 2012. Iris: a chat-oriented dialogue system based on the vector space model. In *Proceedings of the Association for Computational Linguistics (ACL 2012), System Demonstrations*, pages 37–42. Association for Computational Linguistics.

Rafael E. Banchs. 2012. Movie-dic: A movie dialogue corpus for research and development. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, Association for Computational Linguistics (ACL 2012), pages 203–207, Stroudsburg, PA, USA. Association for Computational Linguistics.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning, Advances in Neural Information Processing Systems (NIPS) 2012 Workshop.

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

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio, 2014a. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, chapter On the Properties of Neural Machine Translation: Encoder–Decoder Approaches, pages 103–111. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*.

Salah El Hihi and Yoshua Bengio. 1995. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems (NIPS)*, pages 493–499. Citeseer.

Micha Elsner and Eugene Charniak. 2008. You talking to me? a corpus and algorithm for conversation disentanglement. In *Association for Computational Linguistics (ACL 2008)*, pages 834–842.

Pierfranca Forchini. 2009. Spontaneity reloaded: American face-to-face and movie conversation compared. In *Corpus Linguistics 2009. Abstracts of the 5th Corpus Linguistics Conference*, page 118.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. *Workshop on Modern Machine Learning and Natural Language Processing, Advances in Neural Information Processing Systems (NIPS)*.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltableu: A discriminative metric for generation tasks with intrinsically diverse targets. *CoRR*, abs/1506.06863.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *Proceedings of The 30th International Conference on Machine Learning (ICML 2013)*, pages 1319–1327.

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

Alex Graves. 2012. Sequence transduction with recurrent neural networks. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012), Representation Learning Workshop*.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2015. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2013. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2014)*, page 292.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jan Koutník, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. 2014. A clockwork rnn. *In Proceedings of the International Conference on Machine Learning (ICML 2014)*.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *Speech and Audio Processing, IEEE Transactions on*, 8(1):11–23.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048.

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

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

Roberto Pieraccini, David Suendermann, Krishna Dayanidhi, and Jackson Liscombe. 2009. Are we there yet? research in commercial spoken dialog systems. In *12th International Conference on Text, Speech and Dialogue*, pages 3–13. Springer.

Olivier Pietquin and Helen Hastie. 2013. A survey on metrics for the evaluation of user simulations. *The knowledge engineering review*, 28(01):59–73.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 172–180, Stroudsburg, PA, USA. Association for Computational Linguistics (ACL 2010).

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Empiricial Methods in Natural Language (EMNLP 2011)*, pages 583–593. Association for Computational Linguistics.

Anindya Roy, Camille Guinaudeau, Hervé Bredin, and Claude Barras. 2014. Tvd: a reproducible and multiply aligned tv series dataset. In *LREC*.

Jost Schatzmann, Kallirroi Georgila, and Steve Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *6th SIGDIAL Workshop on DISCOURSE and DIALOGUE*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Association for Computational Linguistics (ACL-IJCNLP 2015)*. In press.

Bayan Abu Shawar and Eric Atwell. 2007. Chatbots: are they really useful? In *LDV Forum*, volume 22, pages 29–49.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, pages 105–133.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015a. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM 2015)*. In press.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015b. A neural network approach to context-sensitive generation of conversational responses. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2015)*. In press.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances In Neural Information Processing Systems (NIPS 2014)*, pages 3104–3112.

David C Uthus and David W Aha. 2013. The ubuntu chat corpus for multiparticipant chat analysis. In *AAAI Spring Symposium: Analyzing Microtext*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *International Conference on Machine Learning (ICML 2015), Deep Learning Workshop*.

Marilyn A Walker, Grace I Lin, and Jennifer Sawyer. 2012a. An annotated corpus of film dialogue for learning and characterizing character style. In *LREC*, pages 1373–1378.

Marilyn A Walker, Jean E Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012b. A corpus for research on deliberation and debate. In *LREC*, pages 812–817.

Steve Young, Milica Gasic, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *IEEE*, 101(5):1160–1179.