Hierarchical Recurrent Encoder Decoder: Single Dialogue Response Model

Ankith Gunapal
agunapal@berkeley.edu

Eric Yang
eric.yang@berkeley.edu

Abstract

Our goal is to design a single response dialogue system based on a series of back and forth dialogues (utterances). We intend to show that a hierarchical encoder-decoder neural network framework (HRED)[1], which can retain context at both the utterance level and the conversation level, will have better perplexity and produce more reasonable responses compared to an ngram model. Performance of each model will be evaluated with perplexity scores.

1 Introduction

Chatbots have recently become a popular area of research with the rise of NLP and Deep Learning frameworks. Significant time and money has been spent into developing chatbots with the ultimate goal of creating chatbots that can produce responses and interactions that are indistinguishable from human to human interactions. This goal has yet to be achieved, however research in the area has made great progress over the past few years. The exploration into an HRED model can provide the building block for a more robust chatbot system.

There are many challenges with developing chatbots that can produce coherent responses, but one of the most difficult is keeping track of linguistic and actual context during a conversation. This is made even more complex for open domain generative models which may need to track conversations over many turns and must have the adaptability to respond to unfamiliar words and phrases. Although we will not be implementing a fully functional chatbot, our focus will be to implement the basic idea behind a chatbot with single sentence responses given an open domain context.

2 Background

An end to end dialogue system utilizing a HRED framework was proposed by Sordoni [1]. The HRED model described by Sordoni [1] makes the assumption that a dialogue can be seen as a sequence of utterances which, in turn, are sequences of tokens. The hierarchical model captures context through two levels of RNNs, one at the utterance level and one at the conversation level. This information is encoded to vector representations and passed to a decoder RNN which outputs an utterance based on the probability distribution of tokens to create a response given the context from the provided dialogue. The response ends when the decoder RNN outputs an end of sentence token.

One of the main benefits of this model is that it can provide context awareness that cannot be captured in more traditional count based model such as a n-gram model. At the input level of the HRED, words are parameterized using a co-occurrence matrix which results in a vector representation of all the words in the vocabulary. The word embeddings enable the model to learn over a greater context windows compared to a n-gram model. Using standard n-grams to compute joint probabilities over dialogues, e.g. computing probability tables for each token given the n preceding tokens, suffers from the curse of dimensionality and is intractable for

any realistic vocabulary size [1].

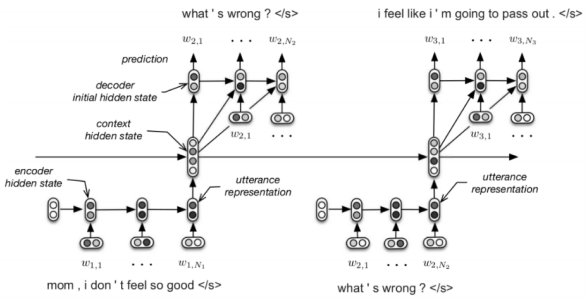
3 Methods

Dataset: We will be using the movie dialogues corpus (Movie-DiC) as our main dataset described by Banchs [3]. This is a corpus scraped from the internet movie script data collection. The corpus contains 132,229 dialogues containing a total of 764,146 turns which have been extracted from 753 movies. The Movie-DiC was created with the intent to provide a wide range of situational, emotional, and stylistic dialogue contexts which should provide a strong training foundation for an open domain conversational agent. The dataset has been preprocessed to reduce sparsity issues. This includes replacing name tokens with ¡person¿ and number tokens with ¡number¿. In addition all words have been made lowercase and all but the 10,000 most common tokens have been replaced with a ¡unk¿ token. Like many corpuses, the distribution of tokens follows a power law distribution where we have a steep drop off for the most common tokens and a extremely long tail.

3.1 Hierarchical Recurrent Encoder-Decoder

We propose using the architecture of a hierarchical recurrent encoder-decoder framework proposed by Sordoni [2] used originally to build a context aware query suggestion system. The first part of the model is a encoder recurrent neural network (RNN). The encoder RNN will take in a vector representation of words (embeddings) from an initial utterance in a dialogue sequence. Embeddings project words that share similarities in meaning and context within sentences or phrases into similar areas of vector space. The encoder RNN maintains the order of embeddings seen in the initial utterance by updating the hidden state and outputs a final hidden state vector with the context retained from the initial utterance. This vector is subsequently passed to the second part of the model, the decoder RNN. The decoder RNN is initialized with the encoding and generates an utterance based on the probability of tokens in a given corpus and the response utterance is generated and completed when the model outputs an end of sentence token.

Figure 1: HRED Diagram (Sordoni [1])



3.2 Gated Recurrent Unit

Within the RNN cell we will make use of a Gated Recurrent Unit (GRU). GRUs will allow us to store context for long term dependencies and have demonstrated to achieve better performance than simpler parameterizations at an affordable computational cost [Sordoni et al [1]]. Figure 2 below shows the equations for a GRU. There are two gates in a GRU, a reset gate r and an update gate z. At a high level the reset gate combines the current context with the previous context and the update gate determines how much of the previous context to retain. h and s_t represent retained or discarded memory in the GRU.

Figure 2: GRU Cell Equations

Reset Gate: $z = \sigma(x_t U^z + s_{t-1} W^z)$

Update Gate: $r = \sigma(x_t U^r + s_{t-1} W^r)$

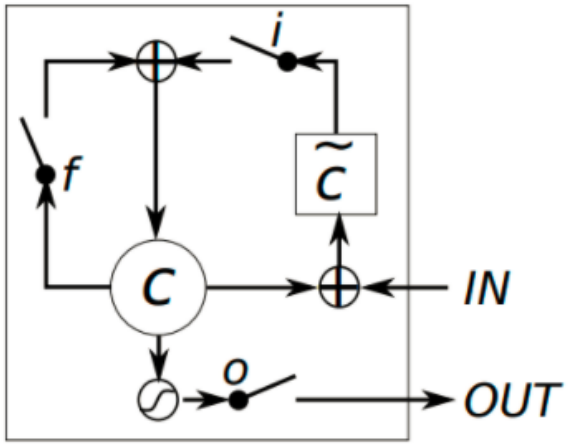
Candidate Update:
$$h = tanh(x_t U^h + (s_{t-1} * r) W^h)$$

Final Update: $s_t = (1 - z) * h + z * s_{t-1}$

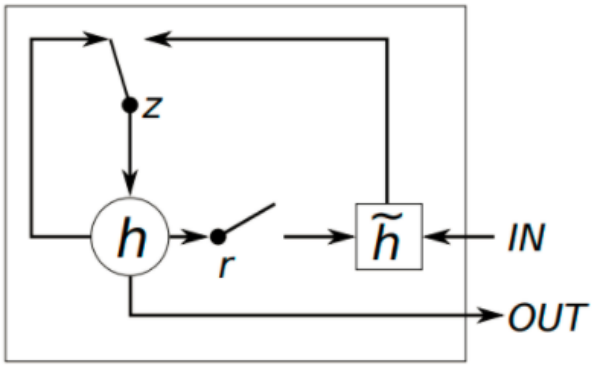
Figure 3 shows diagrams of both an LSTM and GRU cell. Similar to an LSTM cell, a GRU cell has gating units that modulate flow of information inside the cell, however with a

more simplified structure. A GRU cell does not have separate memory cells [4].

Figure 3: LSTM versus GRU cell (Chung [4])



(a) Long Short-Term Memory

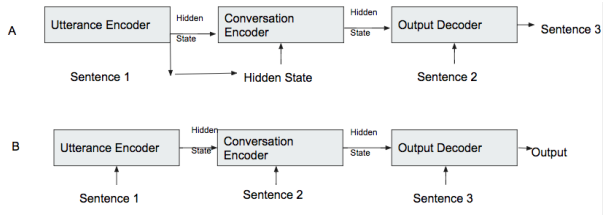


(b) Gated Recurrent Unit

3.3 HRED Implementation Details

We tried two different implementations for the HRED model. These two variations are shown in Figure 4. The implementation shown is used for training the model. In the first implementation we pass in the hidden state from the Utterance Encoder as the input weight vector as well as the initial hidden state of the Conversation Encoder. In the second implementation, we pass the hidden state from the Utterance Encoder to the initial hidden state of the Conversation Encoder. The second sentence is passed as the input weight vector of the Conversation Encoder. For the first implementation we utilized a GRU cell and the second implementation we utilized a LSTM cell.

Figure 4: HRED Implementation



4 Results and Discussion

N-Gram Model: We used an n-gram Model as our baseline model where we tried both Add-k Smoothing and Kneser-Ney Smoothing. For the model we computed the perplexity of the training and test data. The input to the model is a two turn conversation (two utterances) and the model predicts a response, the third utterance in our dataset.

For the the n-gram model, we chose n=4. This is because each sentence ends with a period and an end of sentence token. Thus, at a minimum we need to look at the last 4 words to obtain a better context to predict the next word. If we choose a Trigram model, we will always predict the same sequence of words since the context for every predicted sentence will be the same. We have taken two approaches for predicting the next word, the first is where we output the word with the highest probability of occurrence, the second is utilizing Beam Search.

N-Gram Model Results: With Add-K Smoothing, we see that the sentence is not very meaningful (Table 1). We run into issues with respect to context with the 4-gram model. Once the context is unknown, all the words have an equal probability of being the next word, which results in a random choice of words in our vocabulary. The issues with the Add-k model are reflected in the high perplexity score of 990 achieved on our test data set (Table 3).

The results from a Kneser Ney model are shown in Table 2. It is clear that the

sentence produced by this model is more fluent and the perplextity score of 104 on the test data supports this initial evaluation. As expected, a Kneser Ney model performs better than Add-k because it is able to address some of the context issue we mentioned previously. Specifically we fall back to lower order n-grams to provide more robust context information for the model.

Table 1: Sample Sentence (Add-K)

Input	stop it - he ' s your landlord - you can ' t be picking fights with him . <person> guineas .
Expected	you don ' t give him enough credit chas . <continued_utterance>we better go .
Predicted	sponsored saks wolfi anarchy upon summon mid-life agreeing grab-ass hartford

Table 2: Sample Sentence (KN)

Input	you hear that ? i hear the humming . <person> electricity .
Expected	what are you doing on the reservation ?
Predicted	what ' s the matter with him . <s>

Table 3: Baseline Perplexity

Model	**Perplexity Score**
Add-k w/ Beam Search	990.43
KN w/ Beam Search	103.55

Initially we approached our baseline prediction models with max probability prediction, however Beam Search proved to be more effective. This is because beam search evaluates multiple sequence paths increasing the likelihood we find a more optimal path when compared with max probability. In the case of the baseline calculation we pruned all but the top two results. With max probability, the same sentence has the highest probability of occurring for every input, making it less effective then beam search.

Neural Network Models: To begin our investigation of an HRED model we started with a RNN model using an LSTM cell structure. The RNN was setup with 3 recurrent layers and hidden layers with a size of 200. We used a 3 layer RNN to match the structure of an HRED model as closely as possible. For the HRED model we tested two versions, an LSTM implementation and a GRU implementation. When training all models, training epochs were stopped when we saw less than a 1% improvement in the perplexity score.

Neural Network Model Results: The RNN greatly improved on the perplexity score of both n-gram models. Table 4 shows the perplexity scores of all neural network models for both the first and last training epoch. After the first epoch the RNN already had a greatly reduced perplexity score at 48.22, compared to 103.55 for our best scoring n-gram model. Since we are able to utilize word embeddings with the RNN we expected an improvement in performance since we are capturing longer range context in our prior two utterances. The HRED model improves upon the perplexity scores of the RNN with the best observed scored of 31.98 for the first training epoch of the HRED with GRU. There doesn't appear to be a significant difference between the LSTM and GRU implementations of HRED, however the GRU implementation saw a much improved speed up during training (Table 4).

Although all neural network models saw a great improvement in perplexity scores, we saw a decrease in the fluency of our predicted responses compared to the n-gram models. Table 5 shows the predicted responses from the HRED (LSTM) implementation. The model ends up predicting similar responses/tokens for many of the input sequences. We believe the reason is that the model has learned to only predict the most frequent tokens in the corpus. The lower perplexity suggests that our predicted distribution is closer to our expected distribution when compared to n-gram, however the model is frequently suggesting the most common tokens in the corpus as the most likely response. The corpus is dominated by

punctuation marks and pronouns, since all tokens are weighted equally this may make it difficult for the neural network models to produce diverse responses to the input sequences.

Table 4: Neural Network Perplexity

Model	Perplexity Score
RNN w/ LSTM	48.22
HRED w/ LSTM	31.98
HRED w/ GRU	32.14

Table 5: Sample Sentences (HRED LSTM)

Input	all right , that ' s enough . <person> , we ' re all very impressed with <person> ' s new toy -- toy ?
Expected	t-o-y . toy .
Predicted	what now <person> . s . .

Input	seems <person> ' s a terrible intern , but a talented hacker . thank you for what you ' ve done .
Expected	<person>' t thank me . i only did it for <person> .
Predicted	i ' s . <unk> . .

5 Conclusion

Using perplexity scores to measure language model performance shows that neural network models significantly outperform n-gram models. Specifically the HRED model [1] with either a LSTM or GRU implementation proved to have the best performing perplexity scores. Although perplexity for these models scored relatively well, predicted responses given an input sequence were not fluent and not diverse when compared to the n-gram models. It's likely that although our HRED model benefited from having information over a longer history, this also reduced the diversity of the predicted responses. Since n-grams have limited windows of context, they are more likely to see more diverse contexts, leading to more diverse predictions. In order to alleviate this issue a model would likely need to be able to more accurately identify which context in the conversation is relevant to a predicted response. We might also see an improvement by considering part-of-speech tagging along with the next word probability while generating responses.

References

[1] Iulian V. Serban , Alessandro Sordoni, Yoshua Bengio, Aaron Courville and Joelle Pineau 2016. *Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models.*, volume 1.

[2] Alessandro Sordonif , Yoshua Bengiof , Hossein Vahabig , Christina Liomah , Jakob G. Simonsenh , Jian-Yun Nief 2015. *A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion.*.

[3] R. E. Banchs 2012. *Movie-DiC: A movie dialogue corpus for research and development.*

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio 2015. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*