Multi-Context Dependent Natural Language Generation For More Robust NPC Dialogue

Kevin Dai

Harvard University
CS91r
Professor Stuart Shieber

December 10, 2019

Abstract

This report studies methods to generate video game non-player dialogue that encapsulates three categories of contextual data: personality, events, and interaction settings. An LSTM is proposed to model the language provided by the *Animal Crossing: New Leaf* script data set. To sample from the model, top-k and top-p (nucleus) sampling are used. Quantitative analysis was done using BLEU score, comparing a baseline with top-k, top-p, and both top-k and top-p sampling method performances. It was found that for this model, none of these sampling methods reached reasonable performances in both quantitative or qualitative analyses; baseline scores reached a score of 80% while experimental scores were around 43%, and while mostly semantically and syntactically correct, the generated utterances lacked meaning. This indicates a lack of robustness in the model itself or a lack of quality in the training data.

1 Introduction

Non-player characters, or NPCs, are incredibly important to many video games' story and setting-immersion. These algorithmically controlled interactable people, animals, or objects typically serve to both further the plot and/or provide elements of world-building. However, the player's immersion within the game's world is somewhat broken when repeatedly speaking to NPCs; most NPCs are only programmed to speak from a prewritten script by the developers. Even if some games, such as Nintendo's *Animal Crossing* series, have a dialogue base of more words than all of the Harry Potter books combined in a single installment, players have reported receiving exact responses multiple times.

A natural way to prevent this repetition problem is to use natural language generation to create new utterances every time the player speaks with an NPC. Natural language generation has already been shown to be useful in many other applications, including dialog systems [1], text summarization [2], and machine translation [3]. Most of these applications depend on an input text corpus that is processed and responded to in some way, but there

ZK ApD Fortune <53>Hmm. <2289>I wonder why I'm in such a random, present-giving mood in the first place.<8> <41><42>?

(a) Training data utterance.

ZK ApD Fortune <173>Oh, right!<6> I didn't mean to go over the shopping stars of my town of the town is over there!

(b) Generated utterance

Figure 1: Examples of utterances from the training data **(a)** and generated from the language model **(b)**. The angle bracketed tags are reverse engineered sequences to be read by the game, indicating actions performed by the NPCs such as pausing, nodding, or mentioning the player's name.

have been work done on encoder-decoder models for text generation from contextual data [4]. However, generating NPC dialog can be significantly more complex because of the broad content diversity and multiple categories of contexts applied to the situation. Therefore, this work attempts to capture multiple contextual clues and generate utterances that would potentially fit into these constraints in written dialogue. The goal of the generative model is to not only be able to create coherent dialogue, but also to capture the essence of certain situations, namely personality, events, and specific interaction settings.

This work will identify one main approach to the problem of multi-context dependant natural language generation, using recurrent neural networks (RNNs). Due to the training methods and data that is used, the model used for the task is a long-short term memory (LSTM) model, with which the outputted distribution is sampled using top-k and top-p/nucleus [5] sampling. However, the results, while mostly syntactically and semantically correct, are oftentimes quite nonsensical and can be somewhat repetitive, as can be seen in Figure 1. Evaluations for the generative samplings are done using BLEU scores, which are compared to the BLEU scores calculated by sampling from the training data itself.

2 Methods

2.1 Recurrent Neural Network

The network used for the language model is an RNN. The reasoning behind choosing this model is the sequential nature and variable length of textual data. A RNN is able to handle this type of data due to its recurrent hidden state that is dependent on previous time steps.

Formally, given input text sequences $\boldsymbol{w} = \{w_1, w_2, \ldots, w_T\}$, the input layer at each time step $x(t)$ is initialized as:

$$x(t) = w(t) + h(t-1) \tag{1}$$

where $h(t)$ is the hidden state at time t, calculated as:

$$h(t) = \boldsymbol{\sigma}(Wx(t) + Uh(t-1)) \tag{2}$$

where $\boldsymbol{\sigma}$ is a sigmoid activation function, W is the input weight vector, and U is the recurrent connection weight vector. The output layer consists of a softmax function:

$$y(t) = g(Uh(t)) \tag{3}$$

$$g(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{4}$$

The output of the RNN is according to the probability model of the character sequence given the previous characters:

$$p(w_1, \ldots, w_T) = p(w_1)\dot{p}(w_2|w_1) \ldots p(w_T|w_1, \ldots, w_{T-1}) \tag{5}$$

2.2 Long-Short Term Memory

The architecture of the language model used is a character-based LSTM with 2 layers and hidden dimension of 512. An LSTM is a fitting model because the feedback connections help out with the vanishing gradient problems of classic RNNs [6]. This is done by maintaining a memory variable $c^k(t)$ at the kth LSTM unit at time t. With memory introduced, the hidden state for each LSTM unit becomes:

$$h^k(t) = o^k(t) \tanh(c^k(t)) \tag{6}$$

where o is an output gate that weights the memory c. This output gate is calculated as follows:

$$o^k(t) = \sigma(W_o x(t) + U_o h(t) + C(t))^k \tag{7}$$

where σ is a sigmoid function and C is the diagonalized matrix of c.

The memory variable c is updated via a forget gate $f(t)$, an input gate $i(t)$ and a new memory variable $c'(t)$:

$$c(t) = f(t)c(t-1) + i(t)c'(t) \tag{8}$$

$$c'(t) = \tanh(W_c x(t) + U_c h(t-1)) \tag{9}$$

$$f(t) = \sigma(W_f x(t) + U_f h(t-1) + C(t-1)) \tag{10}$$

$$i(t) = \sigma(W_i x(t) + U_i h(t-1) + C(t-1)) \tag{11}$$

The weight matrices W and U are learned during training.

2.3 Top-k/Top-p Sampling

While there are many ways to sample from the distribution of a language model to generate the text, the model used mainly focuses on top-k sampling [7] and top-p sampling [5].

Top-k sampling is a process by which at each time step, a top number k possible characters are chosen to follow the previous sequence, from which the actual next character is selected. Formally, a top-k vocabulary $V^{(k)} \subset V$ maximizes its generative probability sum $s = \sum_{w \in V^{(k)}} p(w_T|w_1, \ldots, w_{T-1})$, where $p(w_T|w_1, \ldots, w_{T-1})$ is an outputted distribution from a language model. The original distribution can be rescaled for final sampling:

$$p'(w_T|w_1, \ldots, w_{T-1}) = \begin{cases} \frac{p(w_T|w_1,\ldots,w_{T-1})}{s}, & x \in V^{(k)} \\ 0, & \text{else} \end{cases} \tag{12}$$

As a more optimized form of top-k sampling, top-p sampling selects the smallest vocabulary $V^{(p)} \subset V$ such that the sum of the generated probabilities is greater than some threshold p^*:

$$\sum_{w \in V^{(p)}} p(w_T|w_1, \ldots, w_{T-1}) \geq p^* \tag{13}$$

While top-k sampling performs nicely in most cases, it underperforms when there is widely-varied contexts. Alternatively, top-p sampling has been shown to have an accurate correlation between the rescaled model distribution and threshold parameter p^* [4], allowing for little tradeoff between fluency and diversity, even when presented with varied context.

2.4 BLEU Score Evaluation

Using these sampling methods, the model is tested using BLEU scores. While BLEU score is not the ideal method to test the performance of a language model, it is an objective way to compare results. All scores are calculated by taking the average of BLEU scores from five different generated utterances. This BLEU score is calculated as follows:

$$\text{BLEU} = \{\prod_{i=1}^{N} P(i)\}^{\frac{1}{N}} \tag{14}$$

$P(i)$ is the percentage of i-gram tuples in the generated utterances that also occur in the training references:

$$P(i) = \frac{M(i)}{H(i)} \tag{15}$$

where $H(i)$ is the number of i-gram tuples in the generated utterance and $M(i)$ is calculated as:

$$M(i) = \sum_{t_i} \min(C_h(t_i), \max_j(C_{hj}(t_i))) \tag{16}$$

C_h is the number of times i-gram tuple occurs in an utterance. The final BLEU score is the maximum score between all the individual reference utterances and the generated utterance.

The baseline was created by doing this calculated average from five random samples of each context. This work compares this baseline against the performance of top-k sampling, top-p sampling, and top-k and top-p sampling. 4-gram BLEU score was used.

3 Experiments

The data used to train the LSTM model was mined from the *Animal Crossing: New Leaf* script. The data, once converted to a readable XML format, was formatted in groups that contained three context columns: personality, special or everyday events, and specific interaction types or settings. To allow for the inclusion of the context during training, a three word string is prepended to every training point (a one-hot encoded sequence of characters for every utterance). Before training, the data order is shuffled to prevent the test set from being homogeneous. The final cross-entropy loss achieved after 20 epochs (10600 steps) of training was 0.97, with a validation loss of 0.92.

Text generation was done using top-k sampling, top-p sampling, and a combination of both. A top-k value of 5 and top-p value of 0.9 with a temperature of 0.7 were used to sample from the final distribution output of the LSTM. To evaluate the language model generation quality, BLEU scores were calculated for the baseline, top-k, top-p, and top-k + top-p sampling methods. The results of the scoring can be found in Table 1 and a comparison between the baseline and all generated text BLEU scores can be found in Figure 3

	BLEU Score
Baseline	0.976
Top-k	0.455
Top-p	0.440
Top-k + Top-p	0.430

Table 1: Max BLEU scores calculated for the baseline, top-k, top-p, and top-k + top-p sample methods

It is clear that the BLEU scores from the generated text do not even come close to the baseline scores. Although it is not apparent from Table 1, a large portion of the scores from the generated data were very close to zero and most did not reach past 0.3. The amount of data points per context did not have much of a correlation with the scores, whether it was for the baseline or the generated text. Furthermore, the baseline scores also seemed to have very little correlation with the BLEU score performance of any individual sampling for generated text, although there was an overall trend of higher generated text scores for higher baseline scores.

Baseline

BO Ev Xmas <40>My place is all cozy and toasty<6> and ready for some present-opening action!

Top-k

BO Ev Xmas we're considering make me feel like I'm such a little price!<4><26>Where was that is all that?! <60>You know, I can't believe it...<6> <154>It's nice to have a candy,<2> <16>!

Top-p

BO Ev Xmas he does!<4><48>I just don't want to find out where I feel a little bit as well! <5>And I'm going to get a lot of pretty bugs for a nice town.<6> <10>Do you want to move to this town?

Top-k + Top-p

BO Ev Xmas <174><72>.<4><15>I think I won't think of it, but the card is the biggest time to decide what I want to see anything.

Figure 2: Sample utterances generated via the different sampling methods. Baseline included for comparison at the top.

4 Conclusion & Further Directions

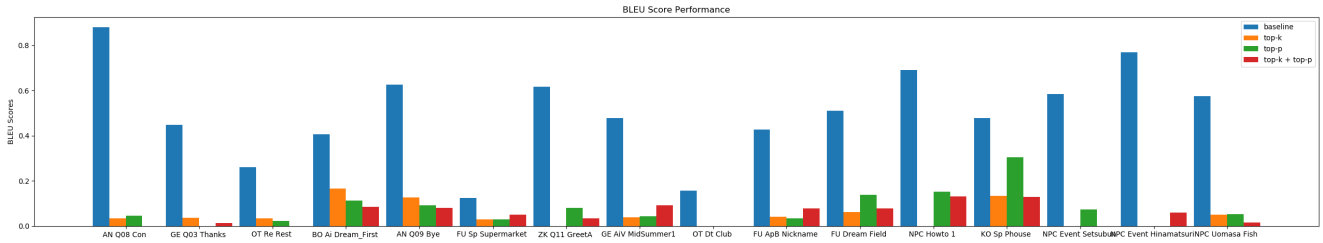
Approaching the problem of multi-context dependent text generation led to the LSTM trained to generate utterances according to the dataset mined from *Animal Crossing: New Leaf*. Although the meaning behind the generated utterances were oftentimes nonsensical, it is clear that they were mostly syntactically and semantically intact. This mostly points to the over-simplicity of the model architecture as well as potentially the training process. Another issue is possibly the structure of the training data itself. The reverse engineered indication tags most likely make the character model too difficult to properly learn distributions. Lastly, the data set could be too small to train a robust model.

While the current results from the model were not satisfactory either quantitatively or qualitatively, there are a number of improvements that can be experimented with in future works.

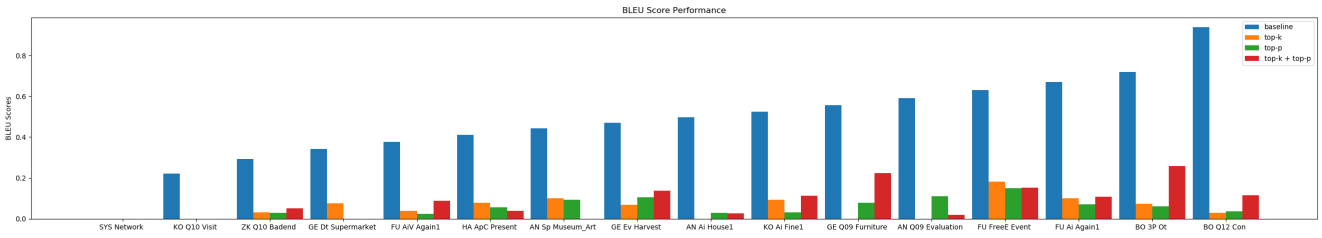
4.1 Data Modifications

As seen in Figure 1, the raw mined training data had many numerical character entities in the form of hexadecimal substrings. These were unparsible by any encodings that were tried; therefore, the sequences were parsed out and replaced with a more legible angle bracket notation. This potentially causes trouble for the language model when learning to generalize due to the wide diversity of these tags. An improvement can be to only keep the tags that pertain to parts of the text, such as the player name or any locations.

Additionally, a character model may not be powerful enough to fully encapture the context with the vocabulary. Further experiments can include a word-based model (unlikely



(a) BLEU scores sorted by number of data points per context.



(b) BLEU scores sorted by baseline score.

Figure 3: BLEU scores for baseline (blue), top-k (orange), top-p (green), and top-k + top-p (red) sampling methods from language model. **(a)** BLEU score does not to seem to correlate with training data point length per context category. **(b)** BLEU score for generated text does not seem to be correlated with baseline BLEU score.

due to size of the data set), or a subword encoding such as `fastText` [8] or byte-pair encoding [9]

4.2 Model Changes

An LSTM by itself as a generative model is perhaps too simplistic to model the language. Therefore, future experiments can include encoder-decoder architecture [4], self-attention [3] and pseudo-self attention [10], and generative-adversarial networks [11].

4.3 Evaluation Methods

Although BLEU score provides a fast, automatic way of evaluating the generative model performance, it oftentimes does not result in a metric that is particularly reflective of the actual quality being generated because it only considers vocabulary and not more subjective measures such as meaning or sentence structure/flow. Thus a better way to evaluate may to use a mechanism like Locally Optimal Learning to Search (LOLS) [12] or simply gathering data for a survey which compare training and generated text, answered by humans to provide a truly qualitative analysis.

References

[1] Owen Rambow, Srinivas Bangalore, and Marilyn Walker. Natural language generation in dialog systems. 2001.

[2] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685, 2015.

[3] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

[4] Jian Tang, Yifan Yang, Samuel Carton, Ming Zhang, and Qiaozhu Mei. Context-aware natural language generation with recurrent neural networks. *CoRR*, abs/1611.09900, 2016.

[5] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[7] Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators. *CoRR*, abs/1805.06087, 2018.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[9] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[10] Zachary M. Ziegler, Luke Melas-Kyriazi, Sebastian Gehrmann, and Alexander M. Rush. Encoder-Agnostic Adaptation for Conditional Language Generation. *arXiv e-prints*, page arXiv:1908.06938, Aug 2019.

[11] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. pages 4006–4015, 2017.

[12] Gerasimos Lampouras and Andreas Vlachos. Imitation learning for language generation from unaligned data. pages 1101–1112, December 2016.