Do Neural Models for Response Generation Fully Exploit the Input Natural Language Text?

Lingfeng ZHANG† and Tetsuya SAKAI†

† Department of Computer Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
E-mail: †zhang-lingfeng@moegi.waseda.jp, ††tetsuyasakai@acm.org

Abstract The advent of deep neural networks has enabled us to build natural language response generation systems. However, due to the black-box nature of these networks, it is unclear how they utilize different parts of the input text to generate responses. To answer this question, previous work has tried perturbing input by reversing, shuffling and dropping words to explore how models react to the perturbation. In the present study, we extend this approach by replacing original words in the input text with randomly chosen ones. We test four RNN-based seq2seq models and one Transformer-based seq2seq model on two open-domain dialogue datasets, and find that context information can be exploited by models much better than word order. Hence it is useful to differentiate context information and word order when designing perturbations.

Key words Seq2seq Model, Chat Bot, Natural Language Processing

1 Introduction

From RNN-based models to Transformer-based models, deep neural network has become more and more advanced and usable in natural language response generation tasks. To better understand models and evaluate their performance, a lot of researches have been done mainly in two ways. One way is comparing the performance of different models. This can be seen in almost all model design works [4, 5]. The other way is analyzing the interior structure of models to find out whether specific structures or mechanisms help improve model performance [1, 6]. However, little attention has been paid to the information exploited by a model. Few work has investigated what kind and how much of the input information has the model exploited.

To answer the question above, our work(注1) attempts to explore the performance changes on multiple response generation models when input information is modified. The idea follows the work of Sankar et al. [14]. Perturbations including word drop, word shuffle and word reverse are manually introduced to input, and then the performance difference before and after perturbations will be computed as models' sensitivity. The performance difference also shows the models' ability to exploit the information which is influenced by perturbations.

Compared with their work, we have extended the approach of Sankar et al. [14] in three ways to better understand what type and amount of input text information have been exploited by models:

(1) More models

The former work only tested three models with attention mechanism. Our work extends the experiment to five models with attention mechanism and bidirectional structure

(2) Differentiate context information and word order

We notice that all perturbation types tested in the former work actually influence both context information and word order. It is not clear whether the performance difference is caused by context information or by word order. Our work has designed a new perturbation type to separate the influence of context information and word order clearly.

(3) Generalizability

Sankar et al. [14] only tested models' performance on its own dataset. Hence, it is not clear whether their models perform well on a different dataset. In our study, we have tested models' performance not only on the datasets where they are trained but also another irrelevant dataset so that we can investigate whether there is a trade-off between exploits of input text and generalizability for models.

The paper is structured as follows: Section 2 introduces some related work. Section 3 describes our experiment details from three parts: dataset, perturbation type and model. Section 4 shares the experiment results and our findings. Section 5 provides conclusions and discusses future work.

2 Related Work

A recent experiment done by Khandelwal et al. [9] trained an LSTM-based language model on Penn Treebank [11] and Wikitext-2 [12] datasets to make prediction on the next word after one sequence. They tested how model performance changes with a de-

(注1)：Code can be found at https://github.com/Lingfeng-Z/ParlAI creasing context size under various conditions including word function, word order, word drop, etc. Differences in perplexity under specific conditions were recorded with decreasing context size. This evaluates how much information that match specific conditions be learnt by the model. They observed that the LSTM-based language model has an effective context size of about 200 tokens on average and local word order only matters for the most recent 20 tokens. However, their work is mainly about language modeling instead of response generation. Since seq2seq models are in the encoder-decoder structure and both parts of a seq2seq model can influence the final performance, other researchers have conducted experiments on each of them. The decoder part has been investigated by Ford et al. [7]. They divided the response generation process into two steps: First, generating part of the words in responses under manual restrictions. Second, generating the other words to fill the blanks and complete the sentences. The restrictions include POS (part-of-speech), word frequency, etc. A Transformer model has been trained and they found that generating common words and function words in the first step leads to a better model performance. This suggests that Transformer can hardly learn word frequency and syntactic information.

Compared with modifying the encoder, operations on decoder are much more indirect and complicated. Therefore, modifying the encoder part may be more effective and straightforward. The encoder part has been discussed by Sankar et al. [14]. Their work is inspired by the above language model. They trained different models on several dialogue datasets and tested their performance under a lot of different perturbations such as drop, shuffle and reverse. The performance was evaluated by the difference in perplexity before and after perturbation. They found that seq2seq models can exploit quite little information from input text. In addition, they found that attention mechanism can increase the models' sensitivity to long-term context information. Although they conducted the experiments at both utterance level and word level, they have not done analysis in detail about why there is a difference in performance before and after introducing perturbations and what kind of information the models exploited from the input text.

To address the above questions, we extend the approach of Sankar et al. [14] by testing more models with popular structures and more types of well-designed perturbations, and attempt to investigate what types of information in the input text and what amount of them are exploited by the response generation models. In addition, how attention mechanism and birdirectional structure help models exploit input information can be clearly demonstrated.

3 Experiment

Our work follows the experiment setup of Sankar et al. [14] and based on the ParlAI [13] framework. Given an input $\mathbf{X}$ containing utterances $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}$, a response generation model is expected to generate a reasonable utterance of response $\mathbf{y}$. This means that a higher conditional probability $P(y|\mathbf{X})$ of each word y in the response utterance is expected. Thus the conditional probability of the total response can be computed as:

$$P(\mathbf{y}|\mathbf{X}) = \prod_{i=1}^{n} P(y_i|y < i, \mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$$

Higher conditional probability of the whole response means lower perplexity of models. Therefore, perplexity is used as the metric to evaluate the performance of different models in our experiments.

$$Perplexity = P(\mathbf{y}|\mathbf{X})^{-1/i}$$

Manual perturbations are introduced to input and perplexity difference before and after perturbations will be recorded. For example, after shuffling the input, if perplexity increase 1.0, we can assert that the loss of information caused by shuffling lead to the perplexity difference 1.0. In other words, our model can learn that kind of information and we can quantify its sensitivity by using perplexity difference. Note that all models are trained normally and all perturbations are added only in the test processes.

3.1 Dataset

Experiments are conducted on two different chit-chat datasets:

Dailydialog is an open-domain dataset [10] containing chit-chat dialogues with annotations for topic, emotion and utterance acts. Table 1 shows the basic statistics of Dailydialog dataset.

Total Dialogues	13,118
Average Speaker Turns Per Dialogue	7.9
Average Tokens Per Dialogue	114.7
Average Tokens Per Utterance	14.6

Table 1 Basic Statistics of Dailydialog

Persona Chat is a crowdsourced dataset [19] collected via Amazon Mechanical Turk. In each dialogue, paired turkers are given their personas randomly before conversation and are required to better know each other through chatting. The statistics of Persona Chat dataset can be found in Table 2.

Total Dialogues	10,907
Average Speaker Turns Per Dialogue	14.9

Table 2 Basic Statistics of Persona Chat

Selecting these two chit-chat datasets is not only because they are easy to get but we also want to keep a coherence with the former work. With the same datasets, the findings of different researches can be compared and discussed together easily and generalized to many other situations.

3.2 Perturbation Type

Here we list all perturbation types we use in our experiment and briefly introduce the details of perturbations.

(1) Drop

Drop perturbation consists of three parts: Word Drop, Verb Drop and Noun Drop. Drop means that we will omit some part of the input. Specially, we drop 30% of all the words in the input in Word Drop, all verbs in Verb Drop and all nouns in Noun Drop. Drop perturbation makes the input lose part of its words, and it can test: (1) Whether some words hold higher importance than the others. (2) Whether words with some specific POS can be understood by models better than the others.

(2) Shuffle & Reverse

This part of perturbation types includes Word Shuffle and Word Reverse. Word Shuffle randomly changes the positions of all words in the input while Word Reverse feeds the input to models in descent order. These two perturbation types can test if models can learn context information and order information from the input.

(3) Replace

Replace perturbation consists of three variants, which are 10%Replace, 20%Replace and 30%Replace. It means that we randomly replace 10%, 20% and 30% of all the words in the input with other words randomly picked from the dictionary. Being replaced by other words may cause a different influence on context information and word order information. After replacement, the context information of every word in the input has been changed but the word order of most of the words in the input has not been changed. Meanwhile, the percentage of words been replaced will increase 10% every two variants. As a result, we can evaluate how much context information are exploited by the models.

(4) Generalization

Generalization means testing the model performance on other datasets. Here we choose Dialog bAbI dataset [3] and pick one batch of samples to replace the present test cases. In generalization, both input and label are replaced with new pair of ones from Dialog bAbI dataset. With this modification, models' generalizability can be tested. Higher generalizability means that the model can learn more commonly owned information from the tested dataset.

3.3 Model

Experiments have been conducted on different models with some of the popular structures widely used nowadays.

(1) RNN-based Models

We have trained a representative RNN-based model, which is a seq2seq encoder-decoder structure with LSTM units [17]. In addition, the widely-used attention mechanism [2] and bi-directional LSTM structure [15] have also been added into the model. Therefore, we have four models to test and compare the performance: *seq2seq_lstm* (standard seq2seq model), *seq2seq_lstm_att* (seq2seq model with attention mechanism), *seq2seq_bidir* (seq2seq model with a bidirectional LSTM encoder), *seq2seq_bidir_att* (seq2seq model with a bidirectional LSTM encoder and attention mechanism). The encoder and the decoder have two layers of LSTM. The hidden state has been set to 256 and the dropout rate has been set to 0.3 to avoid overfitting.

(2) Transformer-based Models

We train a *transformer* model [18] with an embedding size and hidden state of 300, a batch size of 32, a two-head attention and no positional embedding. Limited by the GPU memory size, the batch size of our transformer model on Dailydialog dataset has been set to 3 and we raise the value of epoch to 40 instead of 20 to reach a closing training effect.

All of our models have not been carefully fine-tuned, as recent research [9] shows that hyperparameters will not influence models' sensitivity to input information. Since we are interested only in the perplexity difference before and after perturbation, we believe that hyperparameter changes will not substantially affect our experimental results.

4 Results and Discussion

The main experiment results are shown in Table 3. They suggest that:

(1) Attention mechanism and bidirectional structure can help improve the model performance. Comparing the Test PPL column of each dataset in a chain of *seq2seq_lstm* → *seq2seq_lstm_att* → *seq2seq_bidir* → *seq2seq_bidir_att*, the models' perplexities without perturbation decrease, meaning better performance on test sets.

(2) Response generation models are still struggling in generalizability. Results in the Generalization column show that all models have a large increase in perplexity after switching to another irrelevant dialogue dataset. This indicates that none of them have good generalizability. Among all models, *transformer* has the best generalizability, which suggests that it learns more general features owned by all dialogue datasets and less unique features of the dataset where it is trained than the other models.

(3) All models learn a rather small amount of input text information, which is the same as the findings of Sankar el al. [14]. From Word Drop column to 30%Replace column, all models have a small difference in perplexity after introducing perturbations. The biggest increase in perplexity occurs on Dailydialog dataset when running *seq2seq_bidir* model and perturbing with Word Reverse. Even in that situation, the increase of perplexity is only 5.12, which is not a large number for perplexity increment.

(4) Attention mechanism and bidirectional structure help improve model's sensitivity to context information. In the columns of all perturbations except Generalization, *seq2seq_bidir_att* has the highest performance in nearly half of our tests. *seq2seq_lstm_att* and *seq2seq_bidir* also performs comparably well in most cases. It may because attention mechanism can give weights to all words in the encoder and bidirectional structure can let a model get in touch with bidirectional context information so models with attention mechanism and bidirectional structure can exploit input information better than other models.

(5) Although traditional attention helps the models learn input information, the same thing does not happen on self-attention. As

Models	Test PPL	Word Drop	Verb Drop	Noun Drop	Word Shuf	Word Rev	10%Replace	20%Replace	30%Replace	Generalization
		Dailydialog(ΔPPL_{avg})								
seq2seq_lstm	33.00	+1.53	+0.73	+0.96	+2.79	+2.68	+0.60	+1.14	+1.71	+514.20
seq2seq_lstm_att	29.48	+1.61	+0.94	+2.07	+2.76	+3.69	+0.94	+1.98	+3.22	+535.82
seq2seq_bidir	28.55	+**2.39**	+2.99	+**2.45**	+**3.80**	+**5.12**	+0.96	+1.97	+2.42	+656.15
seq2seq_bidir_att	**28.36**	+2.36	+**3.69**	+**2.45**	+2.98	+2.76	+1.02	+1.94	+3.11	+605.84
transformer	35.03	+0.56	+0.14	+1.26	+0.00	-0.03	+**3.37**	+**3.83**	**+4.20**	+**297.07**
		Persona Chat(ΔPPL_{avg})								
seq2seq_lstm	43.34	+1.09	+0.69	+0.54	+0.82	+1.25	+0.18	+0.85	+1.31	+392.56
seq2seq_lstm_att	41.37	+2.30	+1.02	+1.24	+1.78	+2.19	+0.37	+**1.17**	+**1.99**	+434.53
seq2seq_bidir	39.68	+1.94	+0.60	+0.17	+1.77	+2.56	+**0.40**	+0.94	+1.73	+435.42
seq2seq_bidir_att	**39.35**	+**2.70**	+**1.28**	+**1.26**	+**2.02**	+**2.81**	+0.31	+1.06	+1.92	+2902.65
transformer	40.04	+0.06	+-0.06	+0.02	-0.09	-0.03	+0.28	+0.50	+0.60	+**171.06**

Table 3 Different models' performance in multiple experiments. Column 2 shows the models' average perplexity without perturbation over 3 times' tests. Other columns behind it show the average difference of perplexity over 3 times' tests after doing the perturbations. The increases in perplexity are marked as positive numbers while decrease ones are marked as negative. The best-performance values are marked as bold. It should be clear that only to Test PPL and Generalization column, lower value is better while to all the other columns higher is better.

the two rows of *transformer* show, *transformer* performs poorly on most of the tests, especially on Word Reverse and Word Shuffle, showing that *transformer* cannot learn word order information at all. It is caused by the structure of self-attention mechanism which is not a sequential structure like RNN. Therefore, positional information can hardly be learnt.

(6) Context information is exploited much more by models than word order. From the results of n%Replace columns, we can notice that the perplexity difference caused by introducing 30%Replace is near the ones caused by Word Shuffle and Word Reverse. Specifically, *transformer* has the best score among all models in the n%Replace test of Dailydialog dataset, showing that:

- The Transformer model can learn context information though hard to learn word order information.
- Context and word order are different kinds of information and should be discussed separately.

To analyze the difference clearly, we compute the ratio of n%Replace's performance to Word Shuffle/Reverse's as Table 4 shows.

The results in 30%Replace column of Table 4 show that to most of the models, 30%Replace can have more than 65% performance of Word Shuffle and Word Reverse. The rest are also very close to the threshold we set. Empirically, perturbations like Word Shuffle and Word Reverse have changed both context information and word order, so it is reasonable that the perplexity difference is caused by both of them. However, the results in Table 4 provides evidence that context information has played a much more important role in the final perplexity difference than we expected, which suggests that context information can be better understood and exploited, and is more important than word order.

Perturbation	Model	10%Replace	20%Replace	30%Replace
	Dailydialog			
Word Shuf	seq2seq_lstm	21.5%	40.9%	61.3%
	seq2seq_lstm_att	34.1%	**71.7%**	**116.7%**
	seq2seq_bidir	25.3%	51.8%	63.7%
	seq2seq_bidir_att	34.2%	65.1%	**104.3%**
	tranformer	N/A	N/A	N/A
Word Rev	seq2seq_lstm	22.4%	42.6%	63.8%
	seq2seq_lstm_att	25.5%	53.7%	**87.3%**
	seq2seq_bidir	18.8%	38.5%	47.3%
	seq2seq_bidir_att	37.0%	**70.3%**	**112.7%**
	tranformer	N/A	N/A	N/A
	Persona Chat			
Word Shuf	seq2seq_lstm	22.0%	**103.7%**	**160.0%**
	seq2seq_lstm_att	20.8%	**65.7%**	**118.0%**
	seq2seq_bidir	22.6%	53.1%	**97.7%**
	seq2seq_bidir_att	15.3%	52.5%	**95.0%**
	tranformer	N/A	N/A	N/A
Word Rev	seq2seq_lstm	14.4%	68.0%	**104.8%**
	seq2seq_lstm_att	16.9%	53.4%	**90.9%**
	seq2seq_bidir	15.6%	36.7%	**67.6%**
	seq2seq_bidir_att	11.0%	37.7%	**68.3%**
	tranformer	N/A	N/A	N/A

Table 4 The ratio of n%Replace's perplexity difference to Word Shuffle/Reverse's. The first column lists the perturbation type compared with n%Replace. The second column lists the model name. Columns 3-5 list the ratios. The first row shows n%Replace's type. All ratios are written in percentage and "N/A" represents an infinity number or a negative number which has no value in this experiment. The ratios larger than 65% have been marked as bold to show that context information is much better understood by models than word order here.

5 Conclusions and Future Work

This research studies what kinds of input information and how much of them have been exploited by generative deep-neural networks. We find that:

- All RNN-based models and the Transformer model cannot exploit the input information well.

- Attention mechanism and bidirectional structure do help improve the model's performance as well as make better use of input information.
- The Transformer model can learn more general information from dialogues, leading to a better generalizability. In addition, there is a trade-off between exploits of information and generalizability to the models.
- Context information and word order information are different. In most cases, context information is better exploited by the models.

Although our work has conducted experiments on several models and perturbation types, further steps in testing more advanced models (BERT [5], hierarchical seq2seq model [16], skip-connection [8], etc.) and designing perturbation types which can more accurately evaluate context, word order or syntactic information still should be explored in future work.

References

[1] Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.

[4] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Tobias Domhan. How much attention do you need? a granular analysis of neural machine translation architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1799–1808, 2018.

[7] Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George E Dahl. The importance of generation order in language modeling. *arXiv preprint arXiv:1808.07910*, 2018.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*, 2018.

[10] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*, 2017.

[11] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[12] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[13] Alexander H Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*, 2017.

[14] Chinnadhurai Sankar, Sandeep Subramanian, Christopher Pal, Sarath Chandar, and Yoshua Bengio. Do neural dialog systems use the conversation history effectively? an empirical study. *arXiv preprint arXiv:1906.01603*, 2019.

[15] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[16] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM, 2015.

[17] I Sutskever, O Vinyals, and QV Le. Sequence to sequence learning with neural networks. *Advances in NIPS*, 2014.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[19] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*, 2018.