An Empirical Comparison on Imitation Learning and Reinforcement Learning for Paraphrase Generation

Wanyu Du
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
`wd5jq@virginia.edu`

Yangfeng Ji
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
`yangfeng@virginia.edu`

Abstract

Generating paraphrases from given sentences involves decoding words step by step from a large vocabulary. To learn a decoder, supervised learning which maximizes the likelihood of tokens always suffers from the exposure bias. Although both reinforcement learning (RL) and imitation learning (IL) have been widely used to alleviate the bias, the lack of direct comparison leads to only a partial image on their benefits. In this work, we present an empirical study on how RL and IL can help boost the performance of generating paraphrases, with the pointer-generator as a base model [1]. Experiments on the benchmark datasets show that (1) imitation learning is constantly better than reinforcement learning; and (2) the pointer-generator models with imitation learning outperform the state-of-the-art methods with a large margin.

1 Introduction

Generating paraphrases is a fundamental research problem that could benefit many other NLP tasks, such as machine translation (Bahdanau et al., 2014), text generation (Radford et al., 2019), document summarization (Chopra et al., 2016), and question answering (McCann et al., 2018). Although various methods have been developed (Zhao et al., 2009; Quirk et al., 2004; Barzilay and Lee, 2003), the recent progress on paraphrase generation is mainly from neural network modeling (Prakash et al., 2016). Particularly, the encoder-decoder framework is widely adopted (Cho et al., 2015), where the encoder takes source sentences as inputs and the decoder generates the corresponding paraphrase for each input sentence.

In supervised learning, a well-known challenge of generating paraphrases is the exposure bias: the current prediction is conditioned on the ground truth during training but on previous predictions during decoding, which may accumulate and propagate the error when generating the text. To address this challenge, prior work (Li et al., 2018) suggests to utilize the exploration strategy in reinforcement learning (RL). However, training with the RL algorithms is not trivial and often hardly works in practice (Dayan and Niv, 2008). A typical way of using RL in practice is to train the model with supervised learning (Ranzato et al., 2015; Shen et al., 2015; Bahdanau et al., 2016), which leverages the supervision information from training data and alleviate the exposure bias to some extent. In the middle ground between RL and supervised learning, a well-known category is imitation learning (IL) (Daumé et al., 2009; Ross et al., 2011), which has been used in structured prediction (Bagnell et al., 2007) and other sequential prediction tasks (Bengio et al., 2015).[2]

In this work, we conduct an empirical comparison between RL and IL to demonstrate the pros and cons of using them for paraphrase generation. We first propose a unified framework to include some popular learning algorithms as special cases, such as the REINFORCE algorithm (Williams, 1992) in RL and the DAGGER algorithm (Ross et al., 2011) in IL. To better understand the value of different learning techniques, we further offer several variant learning algorithms based on the RL framework. Experiments on the benchmark datasets show: (1) the DAGGER algorithm is better than the REINFORCE algorithm and its variants on paraphrase generation, (2) the DAGGER algorithm with a certain setting gives the best

---

[1] The data and code for this work can be obtained from: `https://github.com/ddddwy/Reinforce-Paraphrase-Generation`

[2] In this work, we view *scheduled sampling* (Bengio et al., 2015) as an imitation learning algorithm similar to DAGGER (Ross et al., 2011).

results, which outperform the previous state-of-the-art with about 13% on the average evaluation score. We expect this work will shed light on how to choose between RL and IL, and alleviate the exposure bias for other text generation tasks.

2 Method

Given an input sentence $\boldsymbol{x} = (x_1, x_2, \cdots, x_S)$ with length S, a paraphrase generation model outputs a new sentence $\boldsymbol{y} = (y_1, y_2, \cdots, y_T)$ with length T that shares the same meaning with $\boldsymbol{x}$. The widely adopted framework on paraphrase generation is the encoder-decoder framework (Cho et al., 2014). The encoder reads sentence $\boldsymbol{x}$ and represents it as a single numeric vector or a set of numeric vectors. The decoder defines a probability function $p(y_t \mid \boldsymbol{y}_{\leq t-1}, \boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{y}_{\leq t-1} = (y_1, y_2, \ldots, y_{t-1})$ and $\boldsymbol{\theta}$ is the collection of model parameters,

$$p(y_t \mid \boldsymbol{y}_{\leq t-1}, \boldsymbol{x}; \boldsymbol{\theta}) = \text{softmax}(\mathbf{W}\boldsymbol{h}_t) \quad (1)$$

with $\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{h}_{t-1}, y_{t-1}, \boldsymbol{x})$, where $\boldsymbol{f}$ as a nonlinear transition function and $\mathbf{W} \in \boldsymbol{\theta}$ as a parameter matrix. We use the *pointer-generator model* (See et al., 2017) as the base model, which is state-of-the-art model on paragraph generation (Li et al., 2018). We skip the detail explanation of this model and please refer to (See et al., 2017) for further information.

2.1 Basic Learning Algorithms

To facilitate the comparison between RL and IL, we propose a unified framework with the following objective function. Given a training example $(\boldsymbol{x}, \boldsymbol{y})$, the objective function is defined as

$$L(\boldsymbol{\theta}) = \Big\{ \sum_{t=1}^{T} \log \pi_\theta(\tilde{y}_t \mid \boldsymbol{h}_t) \Big\} \cdot r(\tilde{\boldsymbol{y}}, \boldsymbol{y}), \quad (2)$$

Following the terminology in RL and IL, we rename $P(\tilde{y}_t \mid \boldsymbol{y}_{\leq t-1}, \boldsymbol{x}; \boldsymbol{\theta})$ as the the policy function $\pi_\theta(\tilde{y}_t \mid \boldsymbol{h}_t)$. That implies taking an action based on the current observation, where the action is *picking* a word $\tilde{y}_t$ from the vocabulary $\mathcal{V}$. $r(\tilde{\boldsymbol{y}}, \boldsymbol{y})$ is a reward function with $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = 1$ if $\tilde{\boldsymbol{y}} = \boldsymbol{y}$. In our experiments, We use the ROUGE-2 score (Lin, 2004) as the reward function. Algorithm 1 presents how to optimize $L(\boldsymbol{\theta})$ in the online learning fashion. As shown in the pseudocode, the **schedule rates** (α, β) and the **decoding function** Decode($\cdot$) are the keys to understand the special cases of this unified framework.

Algorithm 1 Online learning

1: **Input**: A training example $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$, current schedule rates $\alpha^{(i)}, \beta^{(i)} \in [0, 1]$, learning rate η
2: Initialize $L(\boldsymbol{\theta}) \leftarrow 0$
3: **for** $t = 1, \ldots, T$ **do**
4: $\quad p_1, p_2 \sim \text{Uniform}(0, 1)$
5: $\quad \tilde{y}_{t-1} \leftarrow y_{t-1}$ **if** $(p_1 < \alpha^{(i)})$ **else** $\hat{y}_{t-1}$
6: $\quad \boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{h}_{t-1}, \tilde{y}_{t-1}, \boldsymbol{x})$
7: $\quad \hat{y}_t \leftarrow \text{Decode}(\pi(y \mid \boldsymbol{h}_t))$
8: $\quad \tilde{y}_t \leftarrow y_t$ **if** $(p_2 < \beta^{(i)})$ **else** $\hat{y}_t$
9: $\quad L(\boldsymbol{\theta}) \leftarrow L(\boldsymbol{\theta}) + \log \pi(\tilde{y}_t \mid \boldsymbol{h}_t)$
10: **end for**
11: $\delta\boldsymbol{\theta} \leftarrow \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \cdot r(\tilde{\boldsymbol{y}}^{(i)}, \boldsymbol{y}^{(i)})$
12: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \cdot \delta\boldsymbol{\theta}$

The REINFORCE Algorithm. When $\alpha = 0$, $\beta = 0$, and Decode($\pi(y \mid \boldsymbol{h}_t)$) is defined as as:

$$\begin{aligned} &\text{Decode}(\pi_\theta(y \mid \boldsymbol{h}_{t-1})) \\ &\quad = \text{Random\_Sampling}(\pi_\theta(y \mid \boldsymbol{h}_{t-1})), \end{aligned} \quad (3)$$

Specifically, when $\alpha = \beta = 0$, both $\tilde{y}_{t-1}$ and $\tilde{y}_t$ will choose the sampled values from the Decode function with policy π_θ. It essentially samples a trajectory from the decoder $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T)$ as in the REINFORCE algorithm. The reward is $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = r(\hat{\boldsymbol{y}}, \boldsymbol{y})$ once it has the entire trajectory $\hat{\boldsymbol{y}}$.

The DAGGER Algorithm. When $0 < \alpha < 1$, $\beta = 1$, and Decode($\pi(y \mid \boldsymbol{h}_t)$) is defined as as:

$$\text{Decode}(\pi_\theta(y \mid \boldsymbol{h}_{t-1})) = \underset{y'}{\text{argmax}} \, \pi_\theta(y' \mid \boldsymbol{h}_{t-1}). \quad (4)$$

Depending the value of α, $\tilde{y}_{t-1}$ will choose between the ground truth y_{t-1} and decoded value $\hat{y}_{t-1}$ with the function defined in Equation 4. On the other hand, $\tilde{y}_t$ will always choose the ground truth y_t as $\beta = 1$. Since $\tilde{\boldsymbol{y}} = \boldsymbol{y}$, we have $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = 1$ and the reward can be ignored from Equation 2. In imitation learning, ground truth sequence $\boldsymbol{y}$ is called expert actions. The DAGGER algorithm (Ross et al., 2011) is also called scheduled sampling (Bengio et al., 2015) in recent deep learning literature. To be accurate, in the DAGGER and the scheduled sampling, the α is dynamically changed during training. Typically, it starts from 1 and gradually decays to a certain value along with iterations. As shown in our experiments, the selection of decay scheme has a big impact on model performance.

The MLE Algorithm. Besides, there is a trivial case when $\alpha = 1, \beta = 1$. In this case, $\tilde{y}_{t-1}$ and $\tilde{y}_t$ are equal to y_{t-1} and y_t respectively, and $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = 1$. Optimizing the objective function in Equation 2 is reduced to the maximum likelihood estimation (MLE).

2.2 Other Variant Algorithms

Inspired by the previous three special cases, we offer other algorithm variants with different combinations of (α, β), while the decoding function $\mathrm{Decode}(\pi(y \mid \boldsymbol{h}_t))$ in the same as Equation 3 in all following variants.

- **REINFORCE-GTI** (REINFORCE with Ground Truth Input): $\alpha = 1$, $\beta = 0$. Unlike the REINFORCE algorithm, REINFORCE-GTI restricts the input to the decoder can only be ground truth words, which means $\tilde{y}_{t-1} = y_{t-1}$. This is a popular implementation in the deep reinforcement learning for Seq2Seq models (Keneshloo et al., 2018).

- **REINFORCE-SO** (REINFORCE with Sampled Output): $\alpha = 1$, $0 < \beta < 1$. In terms of choosing the value of $\tilde{y}_t$ as output from the decoder, REINFORCE-SO allows $\tilde{y}_t$ to select the ground truth y_t with probability β.

- **REINFORCE-SIO** (REINFORCE with Sampled Input and Output): $0 < \alpha < 1$, $0 < \beta < 1$. Instead of always taking the ground truth y_{t-1} as input, REINFORCE-SIO further relaxes the constraint in REINFORCE-SO and allows $\tilde{y}_{t-1}$ to be the decoded value $\hat{y}_{t-1}$ with probability α.

Unless specified explicitly, an additional requirement when $0 < \alpha, \beta < 1$ is that its value decays to a certain value during training, which by default is 0.

3 Experiments

Dataset and Evaluation Metrics. We evaluate our models on the Quora Question Pair Dataset [3], and the Twitter URL Paraphrasing Dataset (Lan et al., 2017) [4]. Both datasets contain positive and negative examples of paraphrases, and we only keep the positive examples for our experiments as in prior work of paraphrase generation (Li et al.,

2018; Patro et al., 2018). For the Quora dataset, we follows the configuration of (Li et al., 2018) and split the data into 100K training pairs, 30K testing pairs and 3K validation pairs. For the Twitter dataset, since our model cannot deal with the negative examples as (Li et al., 2018) do, we just obtain the 1-year 2,869,657 candidate pairs from https://languagenet.github.io, and filter out all negative examples. Finally, we divided the remaining dataset into 110K training pairs, 3K testing pairs and 1K validation pairs.

We use the following evaluation metrics to compare our models with other state-of-art neural networks: ROUGE-1 and ROUGE-2 (Lin, 2004), BLEU with up to bi-grams (Papineni et al., 2002). For the convenience of comparison, we also calculate the average of the scores.

Competitive Systems. We compare our results with four competitive systems on paraphrase generation: the sequence-to-sequence model (Bahdanau et al., 2014, Seq2seq), the Reinforced by Matching framework (Li et al., 2018, RbM), the Residual LSTM (Prakash et al., 2016, Res-LSTM), and the Discriminator LSTM model (Patro et al., 2018, Dis-LSTM). Among these competitive systems, the RbM (Li et al., 2018) is more closely related to our work, since we both use the pointer-generator as the base model and apply some reinforcement learning algorithms for policy learning.

Experimental Setup. We first pre-train the pointer-generator model with MLE, then fine-tune the models with various algorithms proposed in section 2. Pre-training is critical to make the RE-INFORCE algorithm and some variants to work. More implementation details are provided in Appendix A.

Result Analysis. Table 1 shows the model performances on the Quora test set, and Table 2 shows the model performances on the Twitter test set. For the Quora dataset, all our models outperform the competitive systems with a large margin. We suspect the reason is because we ran the development set during training on-the-fly, which is not the experimental setup used in (Li et al., 2018).

For both datasets, we find that DAGGER with a fixed (α, β) gives the best performance among all the algorithm variants. The difference between DAGGER and DAGGER* is that, in DAGGER, we use the decay function on α at each iteration,

	Models	SCHEDULE RATE		EVALUATION METRICS			
		α	β	ROUGE-1	ROUGE-2	BLEU	Avg-Score
1	Seq2Seq	-	-	58.77	31.47	36.55	42.26
2	Res-LSTM	-	-	59.21	32.43	37.38	43.00
3	RbM	-	-	64.39	38.11	43.54	48.68
4	Dis-LSTM	-	-	-	44.90	45.70	45.30
5	PRE-TRAINED MLE	$\alpha=1$	$\beta=1$	66.72	47.70	54.01	56.14
6	REINFORCE	$\alpha=0$	$\beta=0$	67.00	47.91	54.06	56.32
7	REINFORCE-GTI	$\alpha=1$	$\beta=0$	67.03	48.10	54.23	56.45
8	REINFORCE-SO	$\alpha=1$	$\beta\to0$	66.88	47.95	54.16	56.33
9	REINFORCE-SIO	$\alpha\to0$	$\beta\to0$	67.62	48.99	55.19	57.26
10	DAGGER	$\alpha\to0$	$\beta=1$	67.64	48.96	55.06	57.22
11	DAGGER*	$\alpha=0.5$	$\beta=1$	**68.34**	**49.99**	**55.75**	**58.02**

Table 1: Performance on Quora dataset. The results of competitive systems are reprinted from prior work: line 1 – 3 are obtained from (Li et al., 2018), line 4 is obtained from (Patro et al., 2018). The average score in the last column is for the convenience of comparison.

	Models	SCHEDULE RATE		EVALUATION METRICS			
		α	β	ROUGE-1	ROUGE-2	BLEU	Avg-Score
1	PRE-TRAINED MLE	$\alpha=1$	$\beta=1$	58.49	43.84	38.45	46.92
2	REINFORCE	$\alpha=0$	$\beta=0$	58.67	44.06	38.46	47.06
3	REINFORCE-GTI	$\alpha=1$	$\beta=0$	58.58	43.89	38.42	46.96
4	REINFORCE-SO	$\alpha=1$	$\beta\to0$	58.58	43.89	38.41	46.96
5	REINFORCE-SIO	$\alpha\to0$	$\beta\to0$	58.82	44.10	38.85	47.25
6	DAGGER	$\alpha\to0$	$\beta=1$	58.84	44.24	38.95	47.34
7	DAGGER*	$\alpha=0.2$	$\beta=1$	**58.95**	**44.34**	**39.04**	**47.44**

Table 2: Performance on Twitter dataset. Since the dataset we obtained is different from (Li et al., 2018), we do not directly compare the results with the prior works.

$\alpha \leftarrow k \cdot \alpha$ with $k = 0.9999$. In our experiments, we also try different decaying rates, and present the best results we obtained (more details are provided in Table B). The selection of α depends on the specific task: for the Quora dataset, we find $\alpha = 0.5$ gives us the optimal policy; for the Twitter dataset, we find $\alpha = 0.2$ gives us the optimal policy.

As shown in line 6 – 11 from Table 1, the additional training with whichever variant algorithms can certainly enhance the generation performance over the pre-trained model (line 5). This observation is consistent with many previous works of using RL/IL in NLP. However, we also notice that the improvement of using the REINFORCE algorithm (line 6) is very small, only 0.18 on the average score.

As shown in line 2 – 7 from Table 2, the additional training with variant algorithms also

shows improved performance over the pre-trained model (line 1). However, for the pointer-generator model, it is more difficult to do paraphrase generation on the Twitter dataset. Since in the Twitter dataset, one source sentence shares several different paraphrases, while in the Quora dataset, one source sentence only corresponds to one paraphrase. This explains why the average improvement in the Twitter dataset is not as significant as in the Quora dataset. Besides, from Table 2, we also find that IL (line 6 – 7) outperforms RL (line 2 – 3), which is consist with the experimental results in Table 1.

Overall, in this particular setting of paraphrase generation, we found that DAGGER is much easier to use than the REINFORCE algorithm, as it always takes ground truth (expert actions) as its outputs. Although, picking a good decay function α can be really tricky. On the other hand, the REINFORCE

algorithm (together with its variants) could only outperform the pre-trained baseline with a small margin.

4 Related Work

Paraphrase generation has the potential of being used in many other NLP research topics, such as machine translation (Madnani et al., 2007) and question answering (Buck et al., 2017; Dong et al., 2017). Early work mainly focuses on extracting paraphrases from parallel monolingual texts (Barzilay and McKeown, 2001; Ibrahim et al., 2003; Pang et al., 2003). Later, Quirk et al. (2004) propose to use statistical machine translation for generating paraphrases directly. Despite the particular MT system used in their work, the idea is very similar to the recent work of using encoder-decoder frameworks for paraphrase generation (Prakash et al., 2016; Mallinson et al., 2017). In addition, Prakash et al. (2016) extend the encoder-decoder framework with a stacked residual LSTM for paraphrase generation. Li et al. (2018) propose to use the pointer-generator model (See et al., 2017) and train it with an actor-critic RL algorithm. In this work, we also adopt the pointer-generator framework as the base model, but the learning algorithms are developed by uncovering the connection between RL and IL.

Besides paraphrase generation, many other NLP problems have used some RL or IL algorithms for improving performance. For example, structured prediction has more than a decade history of using imitation learning (Daumé et al., 2009; Chang et al., 2015; Vlachos, 2013; Liu et al., 2018). In addition, scheduled sampling (as another form of DAGGER) has been used in sequence prediction ever since it was proposed in (Bengio et al., 2015). Similar to IL, reinforcement learning, particularly with neural network models, has been widely used in many different domains, such as coreference resolution (Yin et al., 2018), document summarization (Chen and Bansal, 2018), and machine translation (Wu et al., 2018).

5 Conclusion

In this paper, we performed an empirical study on some reinforcement learning and imitation learning algorithms for paraphrase generation. We proposed a unified framework to include the DAGGER and the REINFORCE algorithms as special cases and further presented some variant learning algo-

rithms. The experiments demonstrated the benefits and limitations of these algorithms and provided the state-of-the-art results on the Quora dataset.

References

JA Bagnell, Joel Chestnutt, David M Bradley, and Nathan D Ratliff. 2007. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*, pages 1153–1160.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

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

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 16–23, Stroudsburg, PA, USA. Association for Computational Linguistics.

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2017. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. *arXiv preprint arXiv:1502.02206*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Peter Dayan and Yael Niv. 2008. Reinforcement learning: the good, the bad and the ugly. *Current opinion in neurobiology*, 18(2):185–196.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. *arXiv preprint arXiv:1708.06022*.

Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the second international workshop on Paraphrasing-Volume 16*, pages 57–64.

Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K. Reddy. 2018. Deep reinforcement learning for sequence to sequence models. *arXiv preprint arXiv:1805.09461*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark. Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1874–1883.

Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie J Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 120–127. Association for Computational Linguistics.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 102–109.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Badri N Patro, Vinod K Kurmi, Sandeep Kumar, and Vinay P Namboodiri. 2018. Learning semantic sentence embeddings using pair-wise discriminator. *arXiv preprint arXiv:1806.00807*.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. *arXiv preprint arXiv:1610.03098*.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 142–149.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368.*

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433.*

Andreas Vlachos. 2013. An investigation of imitation learning algorithms for structured prediction. In *European Workshop on Reinforcement Learning*, pages 143–154.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1808.08866.*

Qingyu Yin, Yu Zhang, Weinan Zhang, Ting Liu, and William Yang Wang. 2018. Deep reinforcement learning for chinese zero pronoun resolution. *arXiv preprint arXiv:1806.03711.*

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 834–842, Stroudsburg, PA, USA. Association for Computational Linguistics.