Simple and Effective Retrieve-Edit-Rerank Text Generation

Nabil Hossain[†] **Marjan Ghazvininejad**[♠] **Luke Zettlemoyer**[♠]

[†]Dept. Computer Science, University of Rochester
[♠]Facebook AI Research

Abstract

Retrieve-and-edit seq2seq methods typically retrieve an output from the training set and learn a model to edit it to produce the final output. We propose to extend this framework with a simple and effective post-generation ranking approach. Our framework (i) retrieves several potentially relevant outputs for each input, (ii) edits each candidate independently, and (iii) re-ranks the edited candidates to select the final output. We use a standard editing model with simple task-specific re-ranking approaches, and we show empirically that this approach outperforms existing, significantly more complex methodologies. Experiments on two machine translation (MT) datasets show new state-of-art results. We also achieve near state-of-art performance on the Gigaword summarization dataset, where our analyses show that there is significant room for performance improvement with better candidate output selection in future work.

1 Introduction

Retrieve-and-edit text generation methods have received significant recent interest; editing human-authored text can potentially avoid many of the challenges that are seen while generating text from scratch, including the tendency to be overly repetitive or to degrade on longer texts (Holtzman et al., 2018, 2019). Retrieve-and-edit methods have been developed for summarization (Cao et al., 2018), machine translation (Wu et al., 2019), language modeling (Guu et al., 2018), and conversation generation (Weston et al., 2018). These methods first retrieve a single output from the training set, and then use a learned model to edit it into the final output.

In this paper, we show that generation performance can be improved with a retrieve-edit-rerank approach that instead retrieves a set of outputs from the training set, edits each independently, and then re-ranks the results to produce the final output. Figure 1 shows an overview of the approach.

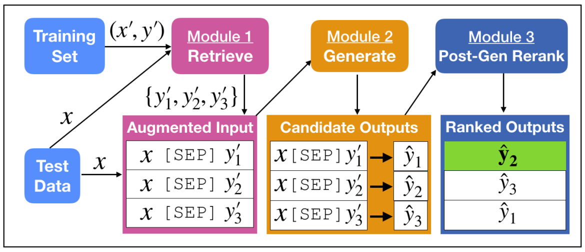

Figure 1: Our retrieve-edit-rerank framework, generating candidate outputs with three retrieved outputs, and re-ranking $\hat{y_2}$ as the best candidate post-generation.

We use standard keyword-based retrieval and a simple editor, where the retrieved output is concatenated to the original input to train a Transformer-based seq2seq editing model. Our final re-ranking step is task specific, but again very simple in every case. Our goal here is not to find the best possible way to do the re-ranking. Instead, we show that gains are possible and that it helps to see what edits are made for multiple candidates before making the final decision, instead of following previous work by trying to select a single candidate before editing.

We evaluate performance on the Gigaword summarization dataset (Rush et al., 2015) and on the English to Dutch (EN-NL) and the English to Hungarian (EN-HU) machine translation (MT) tasks, following Bulte and Tezcan (2019). For MT, we experimented with different re-ranking schemes but found that the original model score (log-likelihood) worked best, amounting to extended beam search within the complete retreive-edit-rerank pipeline. We improve performance by 6.5 BLEU points on EN-NL and 7.5 on EN-HU over the state-of-art Neural Fuzzy Repair system (Bulte and Tezcan, 2019). On Gigaword, we simply re-rank by returning the most common output, and we achieve up to 1.2 ROUGE improvement over the comparable Re^{3}Sum model (Cao et al., 2018). Finally, through qualitative analysis, we find evidence that better post-generation ranking is feasible and can lead to substantial performance improvement, which emphasizes the need for future work in developing new post-generation ranking techniques.

2 Related Work

Recent work has developed retrieve-and-edit approaches for many tasks, including dialogue generation (Weston et al., 2018), language modeling (Guu et al., 2018), code generation (Hashimoto et al., 2018), neural machine translation (NMT) (Gu et al., 2018; Zhang et al., 2018; Cao and Xiong, 2018) and post-editing for NMT (Hokamp, 2017; Dabre et al., 2017). Candidate ranking has served as a core part in some retrieval-based models (Ji et al., 2014; Yan et al., 2016), but these models do not edit the retrieved candidates.

For machine translation, Bulte and Tezcan (2019) developed a retrieve-and-edit based LSTM model called Neural Fuzzy Repair (NFR), which they applied on two MT datasets obtained from (Steinberger et al., 2012). Using a keyword based followed by a token edit distance based retrieval method called *sss+ed*, they showed that concatenating the source and retrieved outputs as the input significantly boosts translation quality. NFR is trained by augmenting the source with up to 3 retrieved outputs, which are fed together into the editing model in several ways. Our approach, instead, simply edits multiple candidates separately and then re-ranks the final results.

For summarization, Re^{3}Sum (Cao et al., 2018) is an LSTM-based model developed under the retrieve-and-edit framework, and tested on the Gigaword summarization (also headline generation) task (Rush et al., 2015). Re^{3}Sum retrieves 30 headlines from the training set using the popular information retrieval method Lucene[1]. Next, it learns a model to pick the single best retrieved headline, which is then edited. BiSET (Wang et al., 2019) is a retrieve-and-edit framework with more complex retrieval ranking and editing stages, which again edits only a single output.

We compare our framework's performance against those of NFR, Re^{3}Sum, and BiSET, showing the effectiveness of post-generation ranking.

3 Framework

Figure 1 shows our proposed retrieve-edit-rerank framework. It has three components: (i) a retrieval mechanism to extract output from the training set; (ii) a seq2seq model to generate output from the source concatenated with the retrieved output; and (iii) a post-generation ranking module to select a high quality output from a set of generated candidates.

For the rest of this paper, we will use (x, y) to represent a source and target pair, (x', y') to denote a retrieved source and output pair from the training set, and $\hat{y}$ to represent the edited/generated output.

3.1 Retrieve

Given input x, the goal of the retrieve module is to find a similar training example (x', y'). We experiment with both Lucene and sss+ed. These can be replaced with any other retrieval methods in the literature.

3.2 Joint Pre-ranking and Generation

Similar to Re^{3}Sum, we design a model that can jointly learn to produce the edited output $\hat{y}$ and re-rank the retrieved outputs y', which we refer to as **pre-ranking**, a common practice to determine which retrieved outputs are worth editing.

For editing, we use a Transformer as our seq2seq model. We provide the model a concatenated input $x\,\texttt{[SEP]}\,y'$, where `[SEP]` is a separator token, and we train it to produce the original target y with a standard cross entropy loss.

For pre-ranking, we add a `[RANK]` token to the Transformer's encoder analogous to the `[CLS]` token in BERT (Devlin et al., 2019). We train the model to predict the similarity between y' and y as the output of the `[RANK]` token, akin to predicting a token from a different vocabulary (Ghazvininejad et al., 2019). We use a cross entropy loss based on a text similarity metric[2], adding it to the Transformer's loss function.

3.3 Post-generation Ranking

For source x, given a set of N input (x concatenated with N retrieved outputs y') and generated candidate output pairs:

$$\{(x\,\texttt{[SEP]}\,y'_1;\ \hat{y_1}), \ldots, (x\,\texttt{[SEP]}\,y'_N;\ \hat{y_N})\}$$

[1] https://lucene.apache.org/

[2] we use BLEU for MT and ROUGE-L for Gigaword. This can be any other text similarity metric.

this module's objective is to select a high quality candidate output. Ideally, we want to find:

$$\hat{y}^* = \arg\max_{\hat{y}_i} \ \textbf{similarity}(\hat{y}_i \, , \, y), 1 \leq i \leq N$$

For **post-ranking**, we use simple ranking functions that work effectively. For MT, we calculate the log-likelihood score of the generated candidate outputs using our trained model (Transformer based) and we choose the candidate that gets the highest model score. For Gigaword, our ranking function simply chooses the most frequently generated output from the list of candidates. In preliminary experiments, we tried other ranking methods, but we did not see a gain compared to our simple post-ranking methods.

Our goal here is not to find the best possible way to do the post-ranking, but only to show that gains are possible. In particular, running the preranker over a larger candidate list is not enough; we find that it is better to see what edits are made for multiple candidates before making the final decision. This strongly suggests that the direction is worthy of future work, to determine how to best combine the evidence from x, x', y' and $\hat{y}$.

4 Experiments

4.1 Datasets and Evaluation Metrics

We test our proposed framework on the machine translation datasets English to Dutch (EN-NL) and English to Hungarian (EN-HU) following previous work (Bulte and Tezcan, 2019). The training, validation, and test set sizes, respectively, are 2.4M, 3000 and 3207, and both datasets have the same source English sentences.

Additionally, we apply our framework on the Gigaword summarization task (Rush et al., 2015). Here, the training, validation, and test set sizes are 3.8M, 189k, and 1951 respectively.

We evaluate MT performance using BLEU[3] scores. For evaluation on Gigaword, we use the F1 scores for ROUGE-1, ROUGE-2, and ROUGE-L with commonly used evaluation parameters[4].

4.2 Implementation Details

We preprocess the data with Byte Pair Encoding (BPE) (Sennrich et al., 2016). Our model is built using the Fairseq library (Ott et al., 2019). We follow most of the Transformer base hyperparameter configurations Vaswani et al. (2017). We use a 6-layer Transformer with 8 attention heads per layer, 512 model dimensions, 2048 hidden dimensions and shared embeddings. Our Transformer uses segment embeddings, with one segment for x and another for y'. For training, we use a learning rate of $5e^{-4}$, a batch size of 128k tokens, the Adam optimizer (Kingma and Ba, 2014), a dropout of 0.3, and a joined dictionary. We train our models for 200k update steps, and we calculate validation loss following each epoch to choose our final model.

For test, we use a beam size of 5.

4.3 Training

For MT, we use the 3 best retrieved outputs per source x to create 4 training examples:

$$\{x, \ x\,\texttt{[SEP]}\,y'_1, \ x\,\texttt{[SEP]}\,y'_2, \ x\,\texttt{[SEP]}\,y'_3\}$$

This is similar to NFR, which then uses for test, the input $x\,\texttt{[SEP]}\,y'_1$ if it exists, and only x otherwise. We use both sss+ed and Lucene to compare how retrieval impacts translation quality.

For Gigaword, we train with 10 retrieved outputs as opposed to 30 retrieved by (Cao et al., 2018), and for testing we use 30 retrieved outputs.

As a baseline, we also train a Transformer without retrieval.

4.4 Results

The MT results in Table 1 show that for both EN-NL and EN-HU, the Transformer without retrieval slightly outperforms the LSTM based NFR which includes retrieval. Replacing LSTM with Transformer in NFR (Tr + sss+ed) gives roughly a 4 point increase in BLEU. Replacing sss+ed with Lucene further increases BLEU by 2 points.

Generating from x concatenated with the best pre-ranked output further improves performance,

System	**EN-NL**	**EN-HU**
LSTM	51.45	40.47
NFR	58.91	48.24
Transformer (Tr)	59.88	49.61
Tr + sss+ed (NFR equivalent)	62.86	52.74
Tr + Lucene + x [SEP] y'_1	64.92	55.16
Tr + Lucene + pre-rank	65.20	55.36
Tr + Lucene + post-rank (ours)	**65.43**	**55.73**

Table 1: BLEU scores on the MT datasets. y'_1 implies using the best retrieved output from Lucene. LSTM results are reported from Bulte and Tezcan (2019).

[3] we use the *multi-bleu.perl* script from Moses.

[4] ROUGE evaluation parameters: -m -n 2 -w 1.2

System	R-1	R-2	R-L
LSTM (from Cao et al. (2018))	35.01	16.55	32.42
Re^{3}Sum	37.04	19.03	34.46
Transformer (Tr)	37.68	18.79	34.87
Tr + Luc + x [SEP] y'_1	37.51	19.15	34.86
Tr + Luc + pre-rank	36.46	18.01	33.85
Tr + Luc + post-rank (ours)	**38.23**	**19.58**	**35.60**
BiSET	39.11	19.78	36.87

Table 2: ROUGE scores for Gigaword summarization. y'_1 implies using the best retrieved output from Lucene.

and the best results are obtained by post-ranking, for which we use the highest scored output according to the model. Overall, our retrieve-edit-rerank system with Transformer, Lucene, and a simple but effective post-ranking function obtains a BLEU score increase of 6.52 on EN-NL and 7.49 on EN-HU over the current state of art NFR model.

Results on Gigaword are shown in Table 2. The Transformer baseline obtains more than a 2 point increase in ROUGE over the LSTM baseline, and it achieves comparable performance to Re^{3}Sum which is LSTM based and uses retrieval. While pre-ranking before editing hurts performance, with post-ranking, our model is able to outperform the Transformer baseline and Re^{3}Sum, obtaining between 0.55-1.24 improvement in ROUGE scores.

Our model comes slightly short of the retrieve-and-edit based state-of-art BiSET (Wang et al., 2019). However, BiSET uses more complex pre-ranking and editing stages which could also incorporated into our model. We leave this exploration to future work as it is largely orthogonal to post-ranking, which is the focus of our efforts.

Overall, with retrieve-edit-rerank, our model outperforms comparable systems which use retrieve-and-edit but no post-generation ranking, demonstrating that a simple post-ranking can boost the performance across two challenging tasks.

5 Post-ranking Analysis

5.1 Oracle Experiments

We report a more detailed analysis on Gigaword, which strongly suggests performance can be further improved by using better post-ranking methods.

For this purpose, we use an Oracle that has access to the gold target outputs. Using this Oracle, we find the N-best generated candidate outputs (out of 30 total generated) in terms of ROUGE-1 similarity to the target. We vary N from 1 to 30, and for each N, we randomly select one of the N-best Oracle-chosen outputs. The ROUGE-1 scores obtained for each N are shown in Figure 2. We also provide lower bounds which show the performance obtained with the candidate from the best N that is least similar to the target.

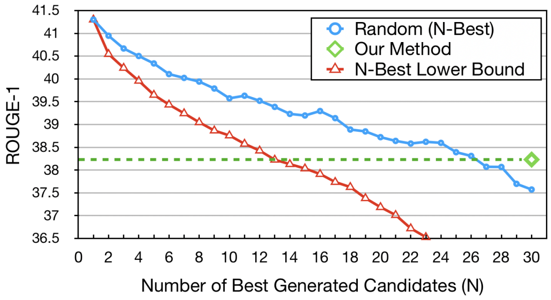

Figure 2: Comparison with Oracle-based post-ranking methods in Gigaword.

Figure 2 shows that our post-generation ranker, which selects the most-frequent candidate output, performs better than choosing a random candidate output (N=30). We also observe that randomly choosing from one of the 1st - 26th best (out of 30) generated outputs surpasses the summarization performance achieved with our post-ranking function. Moreover, choosing any of the 12-best candidates is a feasible strategy that outperforms our ranking function. These observations suggest that many of the 30 retrieved outputs are useful for effective summary generation, and hence, there is a large room for improving by designing new post-generation ranking algorithms.

Similar analysis on MT shows that a ranker that always selects the optimal of the three candidate outputs gets about 3-5 BLEU points improvement over our post-ranking based models, leaving room for further performance gains.

5.2 Examples

To analyze the impact of post-ranking, we compare various outputs from our models for the Gigaword test set, as shown in Table 3.

For the sample 3A, when augmenting the source with y'_1 or the pre-ranked y', the model simply copies the retrieved text and ignores important details from the source. However, the Transformer output indicates that most of the salient information can be obtained from the source itself. By generating multiple outputs with multiple augmented inputs and then choosing the most-frequent output, our post-ranking function helps to lessen the sensitivity of the model to certain retrieved outputs.

Source	jurors visited phil spector s mansion thursday to see the place where actress lana clarkson died , some of them sitting in a chair to mimic the position in which her body was found	
Target	jurors in spector trial visit mansion where actress died	
Transformer	phil spector jury visits scene of actress s death	
Ret-ID	**Retrieved Output**	**Candidate Output**
y'_1	jurors tour phil spector s home	jurors tour phil spector s home
pre-rank (y'_{15})	spector jury tours scene of clarkson s death	spector jury tours scene of clarkson s death
post-rank (y'_{19})	phil spector found guilty of #nd-degree murder	jurors visit phil spector s mansion to see where actress died

Example 3A.

Source	puerto rico ended water rationing for nearly half a million residents tuesday after heavy rain partly replenished a reservoir serving the san juan metropolitan area	
Target	puerto rico ends water rationing	
Ret-ID	**Retrieved Output**	**Candidate Output**
y'_1	for second time in # years water rationed in san juan	puerto rico ends water rationing
pre-rank (y'_4)	water rationing resumes tuesday for ###,### puerto ricans	water rationing resumes tuesday for ###,### puerto ricans
post-rank (y'_3)	puerto rico just days away from water rationing if rain does n't	puerto rico ends water rationing

Example 3B.

Table 3: Sample outputs from the Gigaword test set. "Ret-ID" indicates which of the 30 retrieved y' was used in the input, for example, y'_1 and the pre-ranked y'. For the (most-frequent) post-ranked output, we show the y' for which the generated output had the highest generation score (log-likelihood) from the model.

For sample 3B, post-ranking chooses the output generated using y'_1 which is also the actual target. However, due to a poor retrieval, pre-ranking forces the model to generate an output that largely differs from the target.

We also found some examples where both the retrieve-only y'_1 and the pre-ranked y' were the same, and they were copied verbatim to generate the candidate output. However, several of these copied retrieved outputs were too general summaries, and since the source was ignored during generation, the generated candidate output was missing some article specific information present in the target summary. In many of these cases, simply using the source without any retrieval in the input resulted in an output more representative of the target summary, and also post-ranking helped select this better output. These examples highlight the cases where simply relying on the best retrieval or on the pre-ranking can hurt results since the output generated using only the source without any retrieval is the same as the higher quality post-ranked output.

Overall, these examples demonstrate the flexibility offered by our post-ranking module. It allows the framework to choose between combinations of generations ignoring retrieval, generations using the closest retrieved output and generations using the pre-ranked output. The post-ranking function also acts like a voting scheme, helping to convey the salient information from the inputs to the output while ignoring noise in the inputs.

6 Conclusion and Future Work

In this paper, we presented a retrieve-edit-rerank framework for seq2seq text generation. We used Lucene for retrieval, a Transformer model for editing, and simple task-specific post-generation ranking techniques. We applied the framework on two MT datasets and the Gigaword summarization dataset. Our results show that our simple ranking functions are effective in helping our model outperform the comparable retrieve-and-edit based methods for these datasets.

By performing analysis on Gigaword, we find that there exists room to improve summarization performance with better post-ranking algorithms, a promising direction for future research.

This is in line with our overall goal, which is not to find the best possible way to do the post-ranking, but only to show that gains are possible by editing multiple candidates and then comparing the results. Moving forward, we would like to apply this framework to other retrieve-and-edit based generation scenarios such as dialogue, conversation, and code generation.

References

Bram Bulte and Arda Tezcan. 2019. Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1800–1809, Florence, Italy. Association for Computational Linguistics.

Qian Cao and Deyi Xiong. 2018. Encoding gated translation memory into neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3042–3047, Brussels, Belgium. Association for Computational Linguistics.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

Raj Dabre, Fabien Cromieres, and Sadao Kurohashi. 2017. Enabling multi-source neural machine translation by concatenating source sentences in multiple languages. *arXiv preprint arXiv:1702.06135*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6111–6120, Hong Kong, China. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems*, pages 10052–10062.

Chris Hokamp. 2017. Ensembling factored neural machine translation models for automatic post-editing and quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 647–654.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, Melbourne, Australia. Association for Computational Linguistics.

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

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

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

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. 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. Association for Computational Linguistics.

Ralf Steinberger, Andreas Eisele, Szymon Klocek, Spyridon Pilos, and Patrick Schlüter. 2012. DGT-TM: A freely available translation memory in 22 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 454–459, Istanbul, Turkey. European Language Resources Association (ELRA).

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

Kai Wang, Xiaojun Quan, and Rui Wang. 2019. BiSET: Bi-directional selective encoding with template for abstractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2153–2162, Florence, Italy. Association for Computational Linguistics.

Jason Weston, Emily Dinan, and Alexander H Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. *arXiv preprint arXiv:1808.04776*.

Jiawei Wu, Xin Wang, and William Yang Wang. 2019. Extract and edit: An alternative to back-translation for unsupervised neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1173–1183, Minneapolis, Minnesota. Association for Computational Linguistics.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335, New Orleans, Louisiana. Association for Computational Linguistics.