Edit Aware Representation Learning via Levenshtein Prediction

Edison Marrese-Taylor[1,2], Machel Reid[2,3], Alfredo Solano[2]
[1]National Institute of Advanced Industrial Science and Technology
[2]The University of Tokyo
[3]Google Research, Brain Team
`edison.marrese@aist.go.jp`
`{machelreid, asolano}@weblab.t.u-tokyo.ac.jp`

Abstract

We propose a novel approach that employs token-level Levenshtein operations to learn a continuous latent space of vector representations to capture the underlying semantic information with regard to the document editing process. Though our model outperforms strong baselines when fine-tuned on edit-centric tasks, it is unclear if these results are due to domain similarities between fine-tuning and pre-training data, suggesting that the benefits of our proposed approach over regular masked language-modelling pre-training are limited.

1 Introduction

Editing documents has become a pervasive component of many human activities (Miltner et al., 2019). For example, right before a conference deadline technical papers worldwide are finalized and polished, often involving common fixes for grammar, clarity, and style (Yin et al., 2019). In light of this, it is reasonable to wonder if it would be possible to automatically extract rules from these common edits. This has led researchers to work on the task of learning distributed representations of edits (Yin et al., 2019; Marrese-Taylor et al., 2021; Reid and Neubig, 2022).

Auto-encoding approaches such as the ones proposed by Yin et al. (2019); Marrese-Taylor et al. (2021) have been used previously in the context of representation learning initially in the visual domain, but more recently have been extended to the natural language and video modalities. These approaches largely form the foundation of "self-supervised learning" which enables the learning of representations via objectives which solely require a source datum. An instance of this relevant to Natural Language Processing (NLP) is that of the pre-trained masked language model, BERT (Devlin et al., 2019), in which a source text is initially corrupted with a mask token `[MASK]` and then reconstructed into the original form with a Transformer encoder.

As an alternative to this approach, other works have instead produced representations of edits in an indirect manner, by instead focusing on edit-centric downstream tasks such as edit-based article quality estimation on Wikipedia (Sarkar et al., 2019; Marrese-Taylor et al., 2019), English grammatical error correction (GEC), and machine translation post-editing.

In this paper, differently from existing prior work, we propose a continued pre-training task not based on auto-encoding, which aims at learning distributed representations of natural language edits. In particular, we look at using the Levenshtein algorithm as a form of supervision to encourage a model to learn to convert a given input sequence into a desired output sequence, namely an edit. In particular, we look to answer whether creating a "neural Levenshtein algorithm" is conducive to improved downstream performance on edit-based tasks, given the edit-centricity of the algorithm. In addition to this, we also propose and test two complementary loss functions that help the encoder retain valuable information about the edit.

Our **E**dit **A**ware **R**epresentation **L**earning model, or EARL, is trained in large datasets of edits collected from Wikipedia, and we test it on a selection of edit-centric downstream tasks, including adversarial paraphrasing detection, grammatical error correction and edit-level article quality estimation. Our results show that EARL outperforms strong baselines when fine-tuned on such edit-centric tasks. However, it is unclear if these improvements are due to domain similarities between fine-tuning and pre-training data, suggesting that the benefits of our proposed approach over regular masked language-modelling pre-training are limited. We release[1] our code and trained models to encourage further research in this direction.

[1]`github.com/epochx/earl`

2 Related Work

Our work is primarily related to Yin et al. (2019), who did seminal work in proposing to directly learn distributed representations of edits by means of a task specifically designed for this purpose, based on auto-encoding. The work of Zhao et al. (2019) proposed a similar approach that was specifically tailored at source code. After that, Marrese-Taylor et al. (2021) proposed a variation of this model where a latent variable is introduced as a means to capture properties of natural language edits, which is then tested on a selection or edit-centric tasks.

Our approach is also related to prior work on edit-based generative models, which have utilized semi-autoregressive sequence generation approaches for various tasks. One such example is the work of Guu et al. (2018), who proposed a sentence-level generative model that first samples a prototype sentence and then edits it into a new sentence. Though related, our approach is fundamentally different as in our setting edits are clearly identified by two distinct versions of each item.

In the context of semi-autoregressive language generation, our approach is also related to prior work utilizing the Levenshtein algorithm for such goals. For example, the work of Gu et al. (2019) has explored non-autoregressive methods that use an iterative generation process for machine translation. More recently, the works of Reid and Zhong (2021); Reid and Neubig (2022) have relied on the Levenshtein algorithm to propose edit-based generative approaches for general-purpose tasks. In the former, an iterative edit-based generative model was proposed for the task of style-transfer, where a coarse-to-fine editor transforms text using Levenshtein edit operations similar to ours. In the latter, the authors extend this idea and propose a generic framework to describe the likelihood of multi-step edits, also describing neural models that can learn a generative model of sequences based on these.

Finally, other works have instead produced representations of edits in an indirect manner, by focusing on specific edit-centric downstream tasks. For example, Sarkar et al. (2019) proposed obtaining edit representations that are useful to predict changes in the quality of articles and similarly Marrese-Taylor et al. (2019) proposed to improve quality assessment by jointly predicting the quality of a given edit and generating a description of it in natural language.

3 Levenshtein Prediction

Differently from previous work, here we instead look to see if we can include the Levenshtein objective from a natural language understanding (NLU) perspective. In particular, we look to assess whether Transformer encoder representations can be trained to contain information relevant to an edit, which we hypothesize can be achieved by directly predicting relevant operations and their associated tokens —as produced by an oracle Levenshtein algorithm.

Concretely, we propose a new pre-training task based on self-supervision and look at using the Levenshtein algorithm as a means of pushing a model to learn to convert a given input sequence into a desired output sequence. Let x_- be the original version of an object, and x_+ its form after a change/edit has been applied. We assume that both x_- and x_+ are sequences of tokens such that $x_- = [x_-^1, \ldots, x_-^n]$ and $x_+ = [x_+^1, \ldots, x_+^m]$. We use a fast implementation of the Levenshtein algorithm to identify spans of tokens that have been replaced, inserted or deleted as a result of the edit, and define token-level edit operation labels to indicate how each token was changed.

To process each edit, we first tokenize the pair (x_-, x_+), then use the Levenshtein algorithm to identify the text spans that have changed, and finally further process this output to assign token-level labels capturing the transformations required to convert x_- into x_+.

Let $x_-^{i:j}$ be the sub-span on x_- that goes from positions i to j, our post-processing works on a case-by-case manner, as follows.

1. When a span has been inserted between positions $x_-^{i:j}$, such that it appears in $x_+^{k:j}$, we label the tokens in the latter as w^+, and also label token x_-^{i-1}, as $+$. We do this to provide the model with context of where the insertion was performed, in terms of x_-.

2. Similarly, if the span $x_-^{i:j}$ has been replaced by the span $x_-^{k:l}$, we label the tokens on the respective spans as $\Leftrightarrow$ and $w^{\Leftrightarrow}$.

3. If the span $x_-^{i:j}$ has been removed from the sequence as a result of the edit, we label each token as $-$.

4. Tokens that have not been involved in the edit are label with an empty tag, denoted as $=$.

[CLS]	My	name	is	John	[SEP]	My	last	name	is	Wayne
$=$	$+$	$=$	$=$	$\Leftrightarrow$	$=$	$=$	w^{+}	$=$	$=$	$w^{\Leftrightarrow}$

Figure 1: Example of model input-output for the edit defined by the sequences "My name is John" and " My last name is Wayne" (using whitespace tokenization), where the label $=$ denotes tokens that have not been directly involved in the edit.

Dataset	Edits	Avg. Len
WIKIATOMICEDITS		
Insertions	13.7M	24.5
Deletions	9.3M	25.1
WIKIEDITSMIX	114K	61.6

Table 1: Details of the data utilized for pre-training.

As a result of our post-processing, each token in both x_- and x_+ is mapped to a single Levenshtein operation label: $\Leftrightarrow$, $w^{\Leftrightarrow}$, $+$ or w^{+}, as shown in Figure 1. The end goal of our task is to predict these token-level Levenshtein operations relevant to transform x_- into x_+.

The input to our model is constructed by first prepending the [CLS] token to x_- and x_+, which are separated using the [SEP] token, whose total length we denote as $l = m + n + 2$. This input is embedded and then fed to a Transformer encoder that returns a sequence of hidden representations $\boldsymbol{h}_0, \ldots \boldsymbol{h}_l$. We add a classification head (a linear classifier) and require the model to predict the corresponding label for each token, ignoring tokens that have not been directly involved in the edit (label $=$), using a cross entropy loss ($\mathcal{L}_{lev}$).

We also consider an additional mechanism to enrich the quality of the learned representations, based on techniques that have proven useful in previous work (Marrese-Taylor et al., 2021). Concretely, we note that the vector associated to the [CLS] token ($\boldsymbol{h}_0$) is frequently used to represent the complete model input when using Transformer models such as ours. Since there is no specific token-level Levenshtein label associated to this token, we encourage its representation to contain information about the overall edit. We do this by requiring our model to predict the set of tokens that have been changed in the edit in an unordered fashion, using a separate model head (again, a simple linear projection) which receives this as input, setting $f = \text{MLP}(\boldsymbol{h}_0) \in \mathbb{R}^{|\mathbb{V}|}$, where $|\mathbb{V}|$ is the vocabulary size.

$$\mathcal{L}_{x_\Delta} := -\log p(x_\Delta|\boldsymbol{h}_0) = -\log \prod_{t=1}^{|x_\Delta|} \frac{\exp(f_{x_t})}{\sum_j^V \exp(f_j)} \quad (1)$$

We then let our model minimize the loss function defined in Equation 1, above, where x_Δ is the set of tokens that have been involved in the change (inserted, replaced or removed).

Finally, given the success of the masked language modelling task in model pre-traning (Devlin et al., 2019; Liu et al., 2019) we also experiment combining the Levenshtein prediction task with masked language modeling. Since our model input has a special structure, we propose a modified procedure to generate masks. Concretely, for each example, we either mask the tokens on x_- or on x_+, with a probability of 50% each. Once one side is chosen, we overall follow the approach by Liu et al. (2019) (RoBERTa) to choose which/how many tokens to mask. However, we require the tokens with the relevant Levenshtein operation labels ($\Leftrightarrow$, $w^{\Leftrightarrow}$, $+$, w^{+} or $-$) to always be masked. Once the locations of the masks have been determined, we require the model to predict the masked tokens using the standard masked language modelling loss $\mathcal{L}_{MLM}$. Finally, the total loss used to train our Edit Aware Representation Learning model is the simple summation of the above introduced losses, $\mathcal{L} = \mathcal{L}_{lev} + \mathcal{L}_{x_\Delta} + \mathcal{L}_{MLM}$.

4 Experimental Setup

Pre-training We leverage large available corpora containing natural language edits in a variety of domains. We specifically rely on two datasets of edits extracted from Wikipedia, WIKIEDITSMIX (Marrese-Taylor et al., 2021) and WIKIATOMICEDITS (Faruqui et al., 2018), from which we use the insertions and deletions portions together. Please see details in Table 1. Since pre-training is computationally very expensive, we first use WIKIEDITSMIX, which is much smaller, as a test-bed and for ablation experiments regarding our proposed $\mathcal{L}_{x_\Delta}$ and $\mathcal{L}_{MLM}$ losses. To evaluate the pre-training phase, we utilize the overall and per-token F1-score.

Downstream Tasks We consider a broad selection of datasets and probe the ability of the model to solve three edit-related downstream tasks.

- Paraphrasing Detection: we measure the ability of our edit encoder to model structure, context, and word order information, by means of using PAWS (Yang et al., 2019), an ad-

Model	WikiEditsMix (F1-score)						PAWS		WikiEdits		GEC	
	$+$	w^{+}	$\Leftrightarrow$	$w^{\Leftrightarrow}$	$-$	All	ZS	Ft	ZS	Ft	ZS	Ft
$\mathcal{L}_{lev}$	**89.4**	**96.1**	90.6	88.6	93.7	**91.8**	56.8	94.9	56.8	78.1	**49.5**	52.4
$\mathcal{L}_{lev}$+ $\mathcal{L}_{x_{\Delta}}$	87.8	95.6	89.9	**88.7**	93.5	91.2	**63.8**	94.9	56.7	78.2	48.6	**53.4**
$\mathcal{L}_{lev}$+ $\mathcal{L}_{MLM}$	80.0	94.7	**93.8**	86.3	**95.6**	90.2	60.7	**95.0**	**64.8**	**78.4**	48.8	53.1

Table 2: Results of our ablation experiments on WIKIEDITSMIX.

versarial dataset for paraphrasing detection. Naturally, paraphrases are strongly correlated to edits, as paraphrases are defined as sentences that are semantically similar to each other. PAWS main focus is on sentence pairs that have high lexical overlap but are not paraphrases, with a total of 49,401 pairs for training, and 8K sentences for validation and testing.

- Edit-level Article Quality Estimation: we evaluate the quality of edit representations by means of running a multi-class classification to predict the quality labels on WIKIEDITSMIX (Marrese-Taylor et al., 2021). Concretely, the task is edit-level quality prediction with 4 labels: *spam, vandalism, attack OK*, each corresponding to a different quality of the edit.

- Classification of Grammatical Errors: since grammatical errors consist of many different types, we follow previous work (Marrese-Taylor et al., 2021) and use the WI + LOCNESS (Bryant et al., 2019) dataset for GEC, where each example is labeled into one of 3 CEFR levels (A (beginner), B (intermediate), and C (advanced). We test the ability of the models to classify each edit using a multi-class setting over these three labels.

For evaluation on these downstream tasks, we use accuracy for PAWS, and F1-score for the other datasets. Following previous work, we test our model on two different settings, fine-tuning (Ft) and zero-shot (ZS). For the former, we simply add a new randomly-initialized classification head to our transformer model, and then train all the parameters using a cross-entropy loss based on the labeled data. For the latter, we feed the training examples through our models and extract the vector associated to the [CLS] token ($\boldsymbol{h}_0$) to represent each edit. These representations are then passed through a randomly-initialized MLP to perform classification.

Finally, we compare our model to relevant baselines selected from previous work. On the one hand, we consider the encoder proposed by Yin et al. (2019), but we omit the copy mechanism proposed in the paper in order to make our results comparable. On the other hand, we compare with EVE (Marrese-Taylor et al., 2021), which also uses an auto-encoding loss for training, but does so in variational inference framework. We additionally consider the approach by Guu et al. (2018), but skip their sampling procedure. As our task requires the model to capture structure, context, and word order information, we initialize our model with ROBERTA-base (Liu et al., 2019), which we also adopt as a baseline for downstream experiments.

4.1 Implementation Details

For pre-training, we split WIKIATOMICEDITS into train/valid/testing splits randomly, and use the splits provided by Marrese-Taylor et al. (2021) for WIKIEDITSMIX. For fine-tuning, we respect the original splits for each considered dataset.

Our pre-training is performed using data parallelism to speed up convergence time, but our proposed model can run on single GPUs. We use fairseq (Ott et al., 2019) to implement our model and perform distributed pre-training using 16 NVIDIA V100-16 GB GPUs, and fine-tuning with a single NVIDIA A100-40 GB GPU. We access the former by means of nodes on a large cluster, where each node has four GPUs. For WIKIEDITSMIX we used a single node with a maximum training time of 24 hours (or 100 epochs). on WIKIATOMICEDITS, we used 4 nodes simultaneously, also for a maximum of 24 hours (or 100 epochs).

We use the Adam (Kingma and Ba, 2015) optimizer with a learning rate of 1e-4 during pre-training, and of 1e-3 for fine-tuning on the downstream tasks. Instructions to replicate our experiments and the details of the exact hyper-parameter settings used for pre-training and fine-tuning can be found in our code release.

	Model	PAWS	WikiEditsMix	GEC
ZS	RoBERTa	58.1	63.2	**50.7**
	EARL$_{\text{Mix}}$	**63.8**	56.7	48.6
	EARL$_{\text{Ins+Del}}$	62.2	**57.0**	47.6
Ft	RoBERTa	94.5	**78.9**	54.0
	Guu (2018)	-	74.3	85.6
	Yin (2019)	-	66.8	83.1
	EVE (2021)	-	77.4	**95.8**
	EARL$_{\text{Mix}}$	**94.9**	78.2	53.4
	EARL$_{\text{Ins+Del}}$	94.5	78.3	54.5

Table 3: Results of our model on the downstream tasks, compared to our baselines. EARL$_{\text{Mix}}$ and EARL$_{\text{Ins+Del}}$ indicate models that have been pre-trained on WikiEditsMix and WikiAtomicEdits (Insertions+Deletions), respectively.

5 Results

As can be seen in Table 2, all of our models attain excellent performance on the pre-training task, with an overall F1-Score of more than 90%. We believe this shows that EARL is capable of successfully predicting the operations generated by our oracle Levenshtein editor, suggesting that the representations contain information relevant to the changes that are introduced. This would also explain the high performance attained when fine-tuning on PAWS and WikiEditsMix.

Regarding the impact of $\mathcal{L}_{x_\Delta}$, we see that when added, the overall performance of the model decreases slightly on the pre-training task, but leads to improvements downstream, specially on the zero-shot settings. We believe this result is consistent with previous work, validating the contribution of this loss applied to our setting. Finally, we also see that $\mathcal{L}_{MLM}$ further decreases performance on the pre-training task, but again leads to improved performance when fine-tuning on downstream tasks.

Based on the above findings, we use both losses for our final experiments, which are summarized in Table 3, where we also compare to previous work. We see that when fine-tuned, EARL is able to outperform RoBERTa in PAWS, suggesting that the representations induced by our task help the model learn relevant information about edits. We also see that our model struggles to attain good performance on the GEC tasks, falling considerably behind previous work. We surmise this is due to the pre-training domain being too different from the task. We further note that the best performing model in this task (EVE), is pre-trained on a large corpus of unlabeled GEC edits, a fact that supports our domain shift hypothesis.

Since our model is initialized with RoBERTa-base, we further assessed the impact of our pre-training on a standard NLP downstream task and checked whether it leads to catastrophic forgetting. We considered the widely-used GLUE benchmark (Wang et al., 2018) and selected the MNLI dataset (MNLI) as a test-bed. We find that both RoBERTa and EARL obtain the same accuracy of 87.6, suggesting that our training procedure is compatible with masked language modelling pre-training.

Regarding the models pre-trained on different datasets, we observe that the impact of additional training data is marginal, as the performance of models trained on WikiEditsMix and WikiAtomicEdits is similar across downstream tasks. As these results are well-aligned with our findings regarding the GEC tasks, we think this may suggest the results we are observing are due to pre-training/fine-tuning domain similarity, rather than to the effectiveness of our proposed pre-training.

6 Conclusions and Future Work

This paper proposes a novel approach for training a general-purpose edit representation model, which is not based on auto-encoding. Concretely, we propose a predictive task based on token-level Levenshtein operations where the token-level labels encode the set of operations necessary to transform a given input sentence into an output sentence. Our results show the task is effective at capturing edits, but is not substantially better than the masked language modeling task. We think this evidence still supports the idea that creating a neural model that implements the Levenshtein algorithm is conducive to improved downstream performance on edit-based tasks, suggesting a potential new path for the future of pre-training.

Acknowlegments

This paper is partially based on results obtained from project AAZ20285R1B at the National Institute of Advanced Industrial Science and Technology (AIST). For large pre-training experiments, computational resources of the AI Bridging Cloud Infrastructure (ABCI) provided by AIST were also used. Finally, we are also grateful to the NVIDIA Corporation, which donated one of the GPUs used for this research.

References

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 Shared Task on Grammatical Error Correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

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.

Manaal Faruqui, Ellie Pavlick, Ian Tenney, and Dipanjan Das. 2018. WikiAtomicEdits: A multilingual corpus of Wikipedia edits for modeling language and discourse. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 305–315, Brussels, Belgium. Association for Computational Linguistics.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein Transformer. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11181–11191. Curran Associates, Inc.

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.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*.

Edison Marrese-Taylor, Pablo Loyola, and Yutaka Matsuo. 2019. An Edit-centric Approach for Wikipedia Article Quality Assessment. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 381–386, Hong Kong, China. Association for Computational Linguistics.

Edison Marrese-Taylor, Machel Reid, and Yutaka Matsuo. 2021. Variational Inference for Learning Representations of Natural Language Edits. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13552–13560.

Anders Miltner, Sumit Gulwani, Vu Le, Alan Leung, Arjun Radhakrishna, Gustavo Soares, Ashish Tiwari, and Abhishek Udupa. 2019. On the fly synthesis of edit suggestions. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA):143:1–143:29.

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.

Machel Reid and Graham Neubig. 2022. Learning to Model Editing Processes. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3822–3832, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Machel Reid and Victor Zhong. 2021. LEWIS: Levenshtein editing for unsupervised text style transfer. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3932–3944, Online. Association for Computational Linguistics.

Soumya Sarkar, Bhanu Prakash Reddy, Sandipan Sikdar, and Animesh Mukherjee. 2019. StRE: Self Attentive Edit Quality Prediction in Wikipedia. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3962–3972, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. 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 3687–3692, Hong Kong, China. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. 2019. Learning to Represent Edits. In *Proceedings of the 7th International Conference on Learning Representations*.

Rui Zhao, David Bieber, Kevin Swersky, and Daniel Tarlow. 2019. Neural Networks for Modeling Source Code Edits. In *Proceedings of the 7th International Conference on Learning Representations*.