Variational Inference for Learning Representations of Natural Language Edits

Edison Marrese-Taylor, Machel Reid, Yutaka Matsuo
Graduate School of Engineering, The University of Tokyo
{emarrese,machelreid,matsuo}@weblab.t.u-tokyo.ac.jp

Abstract

Document editing has become a pervasive component of the production of information, with version control systems enabling edits to be efficiently stored and applied. In light of this, the task of learning distributed representations of edits has been recently proposed. With this in mind, we propose a novel approach that employs variational inference to learn a continuous latent space of vector representations to capture the underlying semantic information with regard to the document editing process. We achieve this by introducing a latent variable to explicitly model the aforementioned features. This latent variable is then combined with a document representation to guide the generation of an edited version of this document. Additionally, to facilitate standardized automatic evaluation of edit representations, which has heavily relied on direct human input thus far, we also propose a suite of downstream tasks, PEER, specifically designed to measure the quality of edit representations in the context of natural language processing.

Introduction

Editing documents has become a pervasive component of many human activities (Miltner et al. 2019). This is, to some extent, explained by the advent of the electronic storage of documents, which has greatly increased the ease with which we can edit them.

From source code to text files, specially over an extended period of time, users often perform edits that reflect a similar underlying change. For example, software programmers often have to deal with the task of performing repetitive code edits to add new features, refactor, and fix bugs during software development. On the other hand, 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 lead researchers to recently propose the task of learning distributed representations of edits (Yin et al. 2019).

In this paper, we explore the performance of latent models in capturing properties of edits. Concretely, we introduce a continuous latent variable to model features of the editing process, extending previous work and effectively proposing a new technique to obtain representations that can capture holistic semantic information in the document editing process. Since inference in latent variable models can often be difficult or intractable, our proposal follows previous work framing the inference problem as optimization (Kingma and Welling 2014; Bowman et al. 2016), which makes it an Edit Variational Encoder (EVE). Since it is a known fact that latent variable models for text face additional challenges due to the discrete nature of language (Bowman et al. 2016), in this paper, we also propose a specific mechanism to mitigate this issue.

In addition to proposing EVE, we also note that the empirical evaluation of edit representation has, so far, mainly been based on semi-automatic techniques. For example, including visual inspection of edit clusters or human evaluation of certain quality aspects of the representations. As these evaluations mechanisms are generally time consuming and labor intensive, in this paper we propose a set of extrinsic downstream tasks specifically designed to more comprehensively evaluate the quality of edit representations. Our motivation is to help advance research in this task by introducing a fully automatic, well-defined way to measure what the learned latent space is capable of capturing. Similar endeavors have been a key element in tracking progress and developing new approaches in computer vision (Russakovsky et al. 2015; Antol et al. 2015) and natural language processing (Wang et al. 2018). We draw inspiration from several relevant problems from the latter, and leverage resourced from three different tasks, namely Wikipedia editing, machine translation post-editing and grammatical error correction to present our evaluation scheme.

Our results indicate that evaluation metrics that are related to the task used to obtain edit representations are generally good predictors for the performance of these representations in downstream tasks, although not always. Compared to existing approaches, our model obtains better scores on the intrinsic evaluation, and the representations obtained by our approach can also consistently deliver better performance in our set of introduced downstream tasks. Our code and data are available on GitHub[1].

[1] https://github.com/epochx/PEER

Related Work

Learning distributed representations for edits was perhaps first proposed indirectly by Loyola, Marrese-Taylor, and Matsuo (2017); Jiang and McMillan (2017); Jiang, Armaly, and McMillan (2017). These works note that source code changes, or commits, are usually accompanied by short descriptions that clarify their purpose (Guzman, Azócar, and Li 2014) and explore whether this information could be used to create a mapping between the commits and their descriptive messages. Models that further improve the performance in this task have also been proposed in the last few years. For example, Loyola et al. (2018) proposed ways to provide additional context to the encoder or constrain the decoder with mild success, and Liu et al. (2019) augmented the sequence-to-sequence methods with a copy mechanism based on a pointer net obtaining better performance. Liu et al. (2018) tackled the problem using an approach purely based on machine translation.

Recently, Yin et al. (2019) have directly proposed to learn edit representations by means of a task specifically designed for those purposes. While their ideas were tested on both source code and natural language edits, the work of Zhao et al. (2019) proposed a similar approach that is specifically tailored at source code with relatively less success.

In natural language processing, edits have been studied mainly in two contexts. On one hand, edits are useful for the problem of machine translation post-editing, where humans amend machine-generated translations to achieve a better final product. This task has been crucial to ensure that production-level machine translation systems meet a given level of quality (Specia et al. 2017). Although research on this task has focused mainly on learning to automatically perform post-editing, some recent work has more directly addressed the problem of modelling different editing agents (Góis and Martins 2019) in an effort to understand the nature of the human post-editing process, which is key to achieve the best trade-offs in translation efficiency and quality.

On the other hand, edits have also been relevant in the context of English grammatical error correction (GEC). In this task, given an English essay written by a learner of English as a second language, the goal is to detect and correct grammatical errors of all error types present in the essay and return the corrected essay. This task has attracted recent interest from the research community with several shared tasks being organized in the last years (Ng et al. 2014; Bryant et al. 2019).

Additionally, given the importance that edits play in crowd-sourced resources such as Wikipedia, there has also been work on indirectly learning edit representations that are useful to predict changes in the quality of articles, which is cast as an edit-level classification problem (Sarkar et al. 2019). Similarly, Marrese-Taylor, Loyola, and Matsuo (2019) proposed to improve quality assessment of Wikipedia articles by introducing a model that jointly predicts the quality of a given Wikipedia edit and generates a description of it in natural language.

In terms of the proposed model, our approach is related to autoencoders (Rumelhart, Hinton, and Williams 1986), which aim to learn a compact representation of input data by way of reconstruction. Our approach is also related to variational autoencoders (Kingma and Welling 2014), which can be seen as a regularized version of autoencoders, specifically (Bowman et al. 2016), who introduced an RNN-based VAE that incorporates distributed latent representations of entire sentences. Our architecture is also similar to that of Gupta et al. (2018) who condition both the encoder and decoder sides of a VAE on an input sentence to learn a model suitable for paraphrase generation, but we depart from this classic VAE definition as our generative process includes two observable variables.

Finally, our proposals are also related to Guu et al. (2018), who proposed a generative model for sentences that first samples a prototype sentence from the training corpus and then edits it into a new sentence, with the assumption that sentences in a single large corpus can be represented as minor transformations of other sentences. Instead, in our setting, edits are clearly identified by two distinct versions of each item (i.e. x_- and x_+), which we can regard as a parallel corpus. Although this approach also captures the idea of edits using a latent variable, doing so is not the main goal of the model. Instead, our end goal is precisely to learn a function that maps an edit (represented by the aforementioned x_- and x_+) to a learned edit embedding space.

Proposed Approach

The task of learning edit representations assumes the existence of a set $x^{(i)} = \{x_-^{(i)}, x_+^{(i)}\}$, where $x_-^{(i)}$ is the original version of an object and $x_+^{(i)}$ is its form after a change has been applied. To model the applied change, i.e. the edit, we propose the following generative process:

$$p(\mathbf{x}_+|\mathbf{x}_-) = \int_z p(\mathbf{x}_+, z|\mathbf{x}_-)dz = \int_z p(\mathbf{x}_+|z, \mathbf{x}_-)p(z)dz \quad (1)$$

In the above equation, $\mathbf{x}_+$ and $\mathbf{x}_-$ are observed random variables associated to $x_+^{(i)}$ and $x_-^{(i)}$ respectively, and z represents our continuous latent variable. Since the incorporation of this variable into the above probabilistic model makes the posterior inference intractable, we use variational inference to approximate it. The variational lower bound for our generative model can be formulated as follows:

$$\begin{aligned}\text{ELBO}(\mathbf{x}_+, \mathbf{x}_-) = &-\text{KL}\left[q(z)||p(z)\right] \\ &+ \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{x}_+|z, \mathbf{x}_-)\right] \end{aligned} \quad (2)$$

In Equation 2, $p(z)$ is the prior distribution and $q(z)$ is the introduced variational approximation to the intractable posterior $p(z|\mathbf{x}_-, \mathbf{x}_+)$. We assume that the edits in our dataset are i.i.d., allowing us to compute the joint likelihood of the data as the product of the likelihood for each example. This assumption enables us to write the following expression:

$$\log p(x^{(i)}, \ldots, x^{(N)}) = \sum_{i=1}^{N} \log p(x_+^{(i)}|x_-^{(i)}) \quad (3)$$

Finally, we can write (Zhang et al. 2016):

$$\log p(x_+^{(i)}|x_-^{(i)}) \geq \text{ELBO}(x_+^{(i)}, x_-^{(i)}) \quad (4)$$

$$\geq \mathbb{E}_{\text{z}\sim q(z)}\left[\log p(x_+^{(i)}|x_-^{(i)}, \text{z})\right] - \text{KL}\left[q(\text{z})\|p(\text{z})\right] \quad (5)$$

From now on, we refer to $x_-^{(i)}$ and $x_+^{(i)}$ as x_- and x_+ respectively. We set $\boldsymbol{x}_-, \boldsymbol{x}_+ \in \mathbb{R}^d$ as continuous vectors to be the representation of the original version of an element x_- and its edited form x_+, and $\mathbf{z} \in \mathbb{R}^d$ to be a continuous random vector capturing latent semantic properties of edits. Our goal is to learn a representation function f_Δ that maps an edit (x_-, x_+) to a real-valued edit representation $f_\Delta(\boldsymbol{x}_-, \boldsymbol{x}_+) \in \mathbb{R}^n$. Following previous work, we utilize neural networks to estimate the following components of our generative process:

- $q(\text{z}) \approx q_\phi(\mathbf{z}|\boldsymbol{x}_-, \boldsymbol{x}_+)$ is our variational approximator for the intractable posterior, where q_ϕ denotes the function approximated by this neural network parameterized by ϕ.
- $p(\boldsymbol{x}_+|x_-, \boldsymbol{z}) \approx p_\theta(x_+|\boldsymbol{x}_-, \mathbf{z})$, where p_θ denotes the function defined by the neural net and its dependence on parameters θ.

Our model is optimized with the following loss function:

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{\text{z}\sim q_\phi}\left[\log p_\theta(x_+|\boldsymbol{x}_-, \mathbf{z})\right] - \text{KL}\left[q_\phi(\mathbf{z}|\boldsymbol{x}_-, \boldsymbol{x}_+)\|p(\text{z})\right] \quad (6)$$

From our component definitions, it follows that the neural network parameterizing $p_\theta(x_+|x_-, \boldsymbol{z})$ acts as a **variational neural editor** and is trained to minimize the negative log-likelihood of reconstructing the edited version of each element. On the other hand, the neural net that parameterizes the approximate posterior $q_\phi(\text{z})$ minimizes the Kullback-Leibler divergence with respect to the prior $p(\text{z})$. Since we have made this function depend explicitly on each edit, this component can be considered as a **variational neural edit encoder**.

Our loss function contains an expectation term computed over the random latent variable introduced. To be able to train our neural components using backpropagation, we utilize the reparameterization trick and express the random vector $\mathbf{z} = q_\phi(\boldsymbol{x}_-, \boldsymbol{x}_+)$ as a deterministic variable $\boldsymbol{z} = g_\phi(\boldsymbol{x}_-, \boldsymbol{x}_+, \mathbf{e})$, where $\mathbf{e}$ is an auxiliary variable with independent marginal $p(\mathbf{e})$, and g_ϕ is a function parameterized by a neural net with parameters ϕ. Details about how this function is specified are provided in the corresponding Section.

We can now rewrite the expectation term such that we can utilize a sampling-based method to estimate it. In addition to this, we set the prior distribution to be a Gaussian distribution $p(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_n)$ and make our approximate posterior distribution $q(\mathbf{z})$ also a normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma}$ a diagonal matrix. Since the Kullback-Leibler divergence for these distributions has a closed form, we can write the following loss function:

$$\mathcal{L}(\theta, \phi, \boldsymbol{x}_+) = \frac{1}{2}\sum_{k=1}^{d}\left(1 + \log(\sigma_k^2) - \mu_k^2 - \sigma_k^2\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(\boldsymbol{x}_+|\boldsymbol{x}_-, \mathbf{z}^l) \quad (7)$$

In Equation 7, L is the number of samples to take to obtain a good estimator of the expectation term. In principle, we follow previous work (Kingma and Welling 2014) and set the number of samples to 1 given that we train our model with a minibatch size that is large enough.

Variational Neural Edit Encoder

Edits are represented as sequences of tokens, such that $x_- = [x_-^{(1)}, \ldots, x_-^{(T)}]$ and $x_+ = [x_+^{(1)}, \ldots, x_+^{(N)}]$. To obtain an edit representation, we further process these sequences using matching techniques (Yin et al. 2019) to obtain tags which identify the tokens that have been added ($+$), removed ($-$), replaced ($\Leftrightarrow$), or remained the same ($=$). In this process, as shown in Figure 1, we obtain padded versions of the sequences $\tilde{x}_- = [\tilde{x}_-^{(1)}, \ldots, \tilde{x}_-^{(M)}]$ and $\tilde{x}_+ = [\tilde{x}_+^{(1)}, \ldots, \tilde{x}_+^{(M)}]$, alongside with a sequence of tags $\tilde{x}_{tags}$ with length M indicating the edit operations applied to each position. We denote the vocabulary for these tags as $\mathbb{V}^l = \{-, +, =, \Leftrightarrow\}$ and the vocabularies for the tokens in $\tilde{x}_-$ and $\tilde{x}_+$ as $\mathbb{V}^-$ and $\mathbb{V}^+$ respectively.

We then separately embed the three sequences returned from the matching operation and perform element-wise concatenation to get $\tilde{\boldsymbol{e}}$. We then feed $\tilde{\boldsymbol{e}}$ to a bidirectional LSTM (Graves and Schmidhuber 2005; Graves, Mohamed, and Hinton 2013) as follows:

$$\tilde{\boldsymbol{e}}_i = \begin{bmatrix} \boldsymbol{E}_+(\tilde{x}_-^{(i)}) \\ \boldsymbol{E}_-(\tilde{x}_-^{(i)}) \\ \boldsymbol{E}_{tags}(\tilde{x}_{tags}^{(i)}) \end{bmatrix} \quad (8)$$

$$\vec{\boldsymbol{h}}_e^{(i)} = \text{LSTM}(\vec{\boldsymbol{h}}_e^{(i-1)}, \tilde{\boldsymbol{e}}_i) \quad (9)$$

$$\overleftarrow{\boldsymbol{h}}_e^{(i)} = \text{LSTM}(\overleftarrow{\boldsymbol{h}}_e^{(i+1)}, \tilde{\boldsymbol{e}}_i) \quad (10)$$

$$\boldsymbol{h}_e^{(i)} = [\vec{\boldsymbol{h}}_e^{(i)}; \overleftarrow{\boldsymbol{h}}_e^{(M-i)}] \quad (11)$$

In the equations above $\boldsymbol{E}_+$, $\boldsymbol{E}_-$, $\boldsymbol{E}_{tags}$ are embedding matrices for $\tilde{x}_-$, $\tilde{x}_+$, and $\tilde{x}_{tags}$, respectively. The bi-directional LSTM returns a sequence of hidden states or annotations. Each one of these can be seen as a contextualized, position-aware representation of the edit. We choose the last hidden state, $\boldsymbol{h}_e^{(M)}$, as a fixed-length representation for the whole edit.

Document Encoder

To generate a fixed-length representation for each original document x_-, we use another bidirectional LSTM as follows:

$$\boldsymbol{h}_d^{(i)} = \text{BiLSTM}(\boldsymbol{h}_d^{(i-1)}, \boldsymbol{E}_-(x_-^{(i)})) \quad (12)$$

In a similar fashion to the variational edit encoder, we take the last hidden state as a fixed-length representation of x_-.

$\tilde{x}_-$:	Disposal	of	Waste	material	according	to	the	local	policies	,	respectively	.
$\tilde{x}_+$:	Disposal	of	waste	material	according	to	the	local	policies	.	ϕ	ϕ
$\tilde{x}_{tags}$:	=	=	⇔	=	=	=	=	=	=	⇔	−	−

Figure 1: Example of the edit matching pre-processing step. The example in this figure is taken from the QT21 De-En MQM dataset (index B1_A6_4w_620). Its labels indicate that this post edit solves with problems related to spelling, typography, and the deletion of extra terms that are not needed.

Variational Neural Inferer

As mentioned earlier, the posterior distribution is set to be a multivariate Gaussian parameterized by the mean and variance matrices. Specifically, we treat these as functions of both the original document x_- and the edited document x_+ as follows:

$$g_\phi(\mathbf{z}|\boldsymbol{x}_-, \boldsymbol{x}_+) \sim \mathcal{N}(\boldsymbol{\mu}_\phi(x_-, x_+), \boldsymbol{\Sigma}_\phi(x_-, x_+)) \quad (13)$$

To approximate this posterior, we project the representation of the edit onto the latent space by using a linear projection layer to derive the vector $\boldsymbol{\mu}$ for the mean and another linear projection layer to derive a vector σ for the variance (we assume that $\boldsymbol{\Sigma}_\phi$ is a diagonal matrix so we only need to estimate the values in its diagonal). We do this as follows:

$$\boldsymbol{\mu} = \boldsymbol{W}_\mu \boldsymbol{h}_e^{(M)} + \boldsymbol{b}_\mu \quad (14)$$

$$\log \boldsymbol{\sigma}^2 = \boldsymbol{W}_\sigma \boldsymbol{h}_e^{(M)} + \boldsymbol{b}_\sigma \quad (15)$$

In the equations above, $\boldsymbol{W}_\mu \in \mathbb{R}^{d_z \times d_e}$, $\boldsymbol{W}_\sigma \in \mathbb{R}^{d_z \times d_e}$ represent trainable weight matrices, and $\boldsymbol{b}_\mu \in \mathbb{R}^{d_z}$, $\boldsymbol{b}_\sigma \in \mathbb{R}^{d_z}$ represent the bias vectors of the linear projections we use. Finally, we can write the following:

$$\boldsymbol{z} = g_\phi(\boldsymbol{x}_-, \boldsymbol{x}_+, \mathbf{e}) := \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \mathbf{e} \quad (16)$$

Here, $\mathbf{e} \sim \mathcal{N}(0, \boldsymbol{I})$ is our introduced independent auxiliary random variable, and parameters of g_ϕ are therefore characterized by matrices $\boldsymbol{W}_\mu$, $\boldsymbol{W}_\sigma$ and bias vectors $\boldsymbol{b}_\mu$ and $\boldsymbol{b}_\sigma$.

For use during generation, we project our latent variable, z, to the target space with a linear transformation. We refer to this projected vector as h'_e. This is shown below:

$$h'_e = W_e z + b_e \quad (17)$$

Variational Neural Editor

To reconstruct x_+, we use a decoder which acts as a neural editor. This is implemented using another LSTM. This neural editor is conditioned both on the input document x_- and the edit representation $\mathbf{z}$, and it uses this information to apply the edit by generating x_+.

The procedure works as follows: (1) Firstly, the decoder is initialized with the concatenation of the projected latent vector and the representation of the original document $[\boldsymbol{z}; \boldsymbol{h}_d^{(T)}]$, (2) Since we want the decoder to reuse information from x_- as much as possible, the decoder attends its representation, making use of the set of annotation vectors $\boldsymbol{h}_d$ on each timestep, (3) At each timestep, $\boldsymbol{h}'_e$ is concatenated with the hidden state returned during the previous timestep as follows:

$$\boldsymbol{h}_d^{'(j)} = \text{LSTM}(\boldsymbol{h}_d^{'(j-1)}, [\boldsymbol{E}_+(\tilde{x}_+^{(j)}); \boldsymbol{h}'_e], \boldsymbol{c}_j) \quad (18)$$

The decoder's hidden state at timestep j is referred as $\boldsymbol{h}'_d(j)$ and the context vector $\boldsymbol{c}_j = \sum_i \alpha_{ji} \boldsymbol{h}_d$ is computed using general attention.

The x_Δ Loss

Variational auto-encoders are often found to ignore latent variables when using flexible generators like LSTMs. Thus, in order to increase the likelihood of the latent space to be useful, we propose to encourage the latent vector to contain information about the tokens that have been changed (added, replaced, or removed), which we denote as x_Δ.

Specifically, we require a decoder network to predict the set of tokens that have been changed in an unordered fashion. If we let $f = \text{MLP}(\boldsymbol{z}) \in \mathcal{R}^{|\mathbb{V}_+|}$, we have:

$$\log p(x_\Delta|\mathbf{z}) = \log \prod_{t=1}^{|x_\Delta|} \frac{\exp f_{x_t}}{\sum_j^V \exp f_j} \quad (19)$$

This term is added to Equation 7, and our model is trained to jointly minimize $-\log p(x_\Delta|\mathbf{z})$ together with the rest of the loss terms.

Experimental Setup

PEER: Performance Evaluation of Edit Representations

Previous research on evaluating the quality of edit representations has mainly been by proposed Yin et al. (2019). We mainly find two kinds: *intrinsic evaluations* of edit representations, for which no additional labels are required, and *extrinsic evaluations*, which require additional labels or information.

A detailed revision of the existing literature in terms of *intrinsic evaluations* showed us that this is performed mainly by measuring the gold-standard performance of the neural editor in terms of the average token-level accuracy, by visually inspecting the semantic similarity of neighbors in the latent space using human judgement, or by performing clustering and later visually inspecting some of the clusters obtained. We provide more details about each one of these techniques in our supplementary material. As can be seen, intrinsic evaluations are largely dependent on human studies, which are expensive and difficult to replicate. Instead of relying on this kind of evaluation, in this paper, we resort to automatic and more standard ways to do so. In addition to standard metrics used for generative models such as the cross entropy and BLEU-4, we propose the GLEU (Napoles et al. 2015) evaluation metric. This metric was developed for the GEC task and is essentially is a variant of BLEU modified to account for both the source and the reference, making it more adequate for our task. It can also be interpreted as a more general version of the token-level accuracy metric utilized by Yin et al. (2019).

Regarding *intrinsic evaluations*, we found that literature also offers a broad variety of alternatives. Among these,

the most relevant included (1) visual inspection of the 2D-projected edit space, generally performed for a subset of the edits known to be associated to a certain label, (2) one-shot performance of the neural editor on similar edits — previously identified by means of additional information— , and (3) the ability to capture other properties of the edit (Marrese-Taylor, Loyola, and Matsuo 2019; Sarkar et al. 2019), namely one or many labels associated to it.

Based on these findings, we propose a combination of training and evaluation datasets, each associated to a specific task in natural language processing, to automatically evaluate the quality of edit representations. We define a set of downstream tasks based on three different sources of edits, which we call **PEER** (Performance Evaluation for Edit Representations). Table 1 provides a descriptive summary of the datasets included in PEER, and we provide details about each below:

Dataset	Size	Only +	Only −	Length
WikiAtomicSample	104,000	50.0%	50.0%	25.1
WikiEditsMix	113,983	24.1%	16.6%	61.6
QT21 En-De	24,877	8.4%	8.0%	20.0
QT21 En-De MQM	1,255	10.3%	11.3%	19.2
Lang 8	498,359	13.2%	4.6%	13.5
WI + Locness	25,556	11.9%	4.1%	21.4

Table 1: Description of the datasets we utilize to train and evaluate our models.

Wikipedia Edits: We work with two large resources of human edits on Wikipedia articles.

- **WikiAtomicSample**: We randomly sampled approximately $150K$ insertion and deletion examples from the English portion of the WikiAtomicEdits (Faruqui et al. 2018). After cleaning, we keep $104K$ samples.
- **WikiEditsMix**: We randomly selected 20 of the 200 most edited Wikipedia articles and extract the diff for each revision for each article using the WikiMedia API. We make use of the Wikimedia's ORES (Halfaker and Geiger 2020) API and scrape the *draftquality* label for each revision. There are 4 *draftquality* labels: *spam, vandalism, attack*, and *OK*, each corresponding to a different quality of the edit.

For this task, we evaluate the quality of the edit representations by means of running a multi-class classifier over the edit representations to predict the quality labels in the WikiEditsMix datasets. We use both datasets to train models.

Post-editing: As explained earlier, post-editing is the process whereby humans amend machine-generated translation. We choose one of the largest resources of human-annotated examples to train and evaluate our models.

- **QT21 De-En**: We work with the German-English portion of the QT21 dataset (Specia et al. 2017), which originally contains a total of 43,000 examples of machine translation human post-edits. The machine translation output over which post-editing is performed to create this dataset is an implementation of the attentional encoder-decoder architecture and uses byte-pair encoding (Sennrich, Haddow, and Birch 2016).
- **QT21 De-En MQM**: A subset of 1,800 examples of the De-En QT21 dataset, annotated with details about the edits performed, namely the reason why each edit was applied. Since the dataset contains a large number of edit labels, we select the classes that are present in at least 100 examples and generate a modified version of the dataset for our purposes. Examples where no post-edit has been preformed are also ignored.

The evaluation scheme on the post-editing task is based on the unlabeled data in QT21 De-En for training and the labeled data in the QT21 De-En MQM dataset for testing. Since each test example is associated to a variable number of labels, this task is cast as multi-label classification.

Grammatical Error Correction (GEC): We consider the task of English GEC, which has attracted a lot of interest from the research community in the last few years. Since grammatical errors consist of many different types we follow previous work by Bryant et al. (2019) and use some of the datasets released for this shared task, which work with well-defined subsets of error types.

- **Lang-8 Corpus of Learner English (Lang 8)**: A corpus for GEC derived from the English subset of the Lang-8 platform, an online language learning website that encourages users to correct each other's grammar (Mizumoto et al. 2012). In particular, we work with the version of the dataset released by Bryant et al. (2019) and further process it to skip examples where there are no grammar corrections.
- **W&I + LOCNESS (WI + Locness)**: A dataset which was compiled by Bryant et al. (2019), built on top of (1) a subset of the LOCNESS corpus (Granger 2014), which consists of essays written by native English students manually annotated with grammar errors, and (2) manually annotated examples from the Write & Improve online web platform (Yannakoudakis et al. 2018). This dataset contains 3,600 annotated examples across three different CEFR levels (Little 2006): A (beginner), B (intermediate), and C (advanced). Again, we ignore examples where there are no grammar corrections.

The evaluation scheme for GEC consists on training models on the unlabeled Lang 8 dataset, and the evaluation is performed using the labels in WI + Locness, which associates CEFR difficulty levels to each example. Concretely, the problem is a multi-class classification problem.

Comparison to Prior Work

Using PEER as a test bed, we compare the performance of EVE against two relevant baselines. Firstly, we consider a variation of the deterministic encoder by Yin et al. (2019), with the only difference being that we do not include the copy mechanism in order to make results directly comparable.

Secondly, we consider the approach by Guu et al. (2018). To adapt this to our setting, we skip the sampling procedure required in their case since our edits are already pairs of sentences and proceed to directly incorporate their edit encoding mechanism into our model. Following their approach, we first identify the tokens that have been added and removed for each edit, which we denote as x_{Δ}^{+} and x_{Δ}^{-}. Each one of these token sequences is treated like a bag-of-words, encoded using trainable embedding matrix $\boldsymbol{E}$, aggregated using sum pooling and finally projected using two different linear layers to obtain $\boldsymbol{h}_{+}$ and $\boldsymbol{h}_{-}$. These are finally combined to obtain $\boldsymbol{f} = [\boldsymbol{h}_{+}; \boldsymbol{h}_{-}]$. In their approach, a sample from the approximate posterior q is simply a perturbed version of $\boldsymbol{f}$ obtained by adding von-Mises Fisher (vMF) noise, so they perturb the magnitude of $\boldsymbol{f}$ by adding uniform noise.

$$q(\mathbf{z}_{\text{dir}}|x_{\Delta}^{+}, x_{\Delta}^{-}) = \text{vMF}\left(\mathbf{z}_{\text{dir}}; \boldsymbol{f}_{\text{dir}}, \kappa\right) \quad (20)$$

$$q(\mathbf{z}_{\text{norm}}|x_{\Delta}^{+}, x_{\Delta}^{-}) = \text{Unif}(\mathbf{z}_{\text{norm}}; [\tilde{\boldsymbol{f}}_{\text{norm}}, \tilde{\boldsymbol{f}}_{\text{norm}} + \epsilon]) \quad (21)$$

In the equations above, $\boldsymbol{f}_{\text{norm}} = \|f\|$, $\boldsymbol{f}_{\text{dir}} = \boldsymbol{f}/\boldsymbol{f}_{\text{norm}}$, $\text{vMF}\left(v; \mu, \kappa\right)$ denotes a vMF distribution with mean vector μ and concentration parameter κ, and $\tilde{\boldsymbol{f}}_{\text{norm}} = \min(\boldsymbol{f}_{\text{norm}}, 10 - \epsilon)$ is the truncated norm. Finally, the resulting edit vector is $\mathbf{z} = \mathbf{z}_{\text{dir}} \cdot \mathbf{z}_{\text{norm}}$, resulting in a model whose KL divergence does not depend on model parameters. We adapt their code release[2] and integrate it into our codebase utilizing the same hyper-parameters. In order to make results comparable, we do not use pre-trained embeddings. For additional details, please refer to their paper and/or implementation.

Implementation Details

Despite the VAE's appeal as a tool to learn unsupervised representations through the use of latent variables, there exists the risk of "posterior collapse" (Bowman et al. 2016). This occurs when the training procedure falls into the trivial local optimum of the ELBO objective, in which both the variational posterior and true model posterior collapse to the prior. This often means these models end up ignoring the latent variables, which is undesirable because an important goal of VAEs is to learn meaningful latent features for inputs. To deal with these issues, we utilize word dropout, and we anneal the KL term in the loss utilizing a sigmoid function, following the work of Bowman et al. (2016). Additionally, we also follow recent work of Li et al. (2019), who discovered that when the inference network of a text VAE is initialized with the parameters of an encoder that is pre-trained using an auto-encoder objective, the VAE model does not suffer from the posterior collapse problem. Therefore, our model is first trained with zero KL weight until convergence. Then, the decoder is reset, and the whole model is re-trained.

For the intrinsic evaluation, BLEU and GLEU scores are computed over the beam-search-generated output. Following the scheme of PEER, we first pre-train each model using the training scheme (i.e. the intrinsic task) and obtain a function that maps edits to a fixed-length vector representing a point in the latent space. We then evaluate this mapping function using the extrinsic evaluation setup. Since EVE is a probabilistic model, we utilize MAP and select the vector that parameterizes the mean of the posterior distribution as a deterministic edit representation for each example.

[2]https://github.com/kelvinguu/neural-editor

Results

To assess the contribution of each of our proposals, we performed an ablation study in two settings, WikiAtomicSample → WikiEditsMix and Lang 8 → WI + Locness. Specifically, we were interested in studying the effect on performance of our x_{Δ} loss, of the Kullback-Leibler divergence and of the pre-training technique proposed by Li et al. (2019). We evaluated each model variation in both the intrinsic and extrinsic tasks using the validation set on each case.

Data	Model	BLEU	GLEU	Acc
WikiAtomicSample → WikiEditsMix	Base	0.81	0.79	0.672
	+ x_{Δ} loss	0.82	0.80	0.767
	+ KL loss	0.77	0.75	0.649
	EVE	**0.84**	**0.82**	**0.780**
Lang 8 → WI + Locness	Base	0.65	0.58	0.831
	+ x_{Δ} loss	0.65	0.57	0.939
	+ KL loss	0.56	0.46	0.409
	EVE	**0.68**	**0.61**	**0.958**

Table 2: Results of our ablation studies. BLEU and GLEU scores are computed over the validation split, and Acc stands for accuracy of the respective downstream classification task for each dataset, also computed on the validation split.

Table 2 summarizes the results of our ablation experiments. Results show the effectiveness of the introduced x_{Δ} loss, which consistently helps the baseline model obtain better performance. The fact that performance not only improves on the intrinsic tasks, but also on the extrinsic evaluation, suggests that this technique effectively helps the latent code store meaningful information about the edit. On the other hand, we see that the addition of the KL term to the loss tends to have negative effects on both the intrinsic and extrinsic tasks, evidencing the instability added by this constraint to the encoder. This result is not surprising, being consistent with previous findings in the context of text VAEs (Bahuleyan et al. 2018; Li et al. 2019). Finally, results of our full model show that both the x_{Δ} loss as well as the pre-training trick can be effectively combined to help the encoder stabilize and encourage the latent space to contain relevant information about the edits, leading to better performance overall.

Table 3 shows our obtained results and compares them to relevant prior work by means of PEER. If we focus on the intrinsic evaluations, we can see that our approach is able to provide better performance in two datasets, with the deterministic baseline by Yin performing better elsewhere. Since these metrics are highly concerned with the reconstructive capabilities of the neural editor, we think this evidence mostly suggests that the Guu et al. (2018) neural edit encoder is less capable of storing relevant information from

Train. Data	Model	Intrinsic Evaluation				Extrinsic Evaluation			
		Valid		Test		Eval. Data	Accuracy		
		BLEU	GLEU	BLEU	GLEU		Train	Valid	Test
WikiAtomicSample	Guu	0.63	0.60	0.28	0.26	WikiEditsMix	0.738	0.740	0.743
	Yin	0.81	0.79	0.81	0.79		0.671	0.672	0.668
	EVE	**0.84**	**0.82**	**0.84**	**0.82**		**0.782**	**0.780**	**0.774**
WikiEditsMix	Guu	0.56	0.53	0.54	0.52		**0.670**	**0.668**	**0.666**
	Yin	**0.65**	**0.65**	**0.65**	**0.65**		0.604	0.597	0.600
	EVE	0.58	0.61	0.55	0.57		0.637	0.642	0.638
Lang 8	Guu	0.53	0.43	0.51	0.41	WI + Locness	0.924	0.856	0.856
	Yin	0.65	0.58	0.65	0.58		0.836	0.831	0.831
	EVE	**0.68**	**0.61**	**0.68**	**0.60**		**0.971**	**0.958**	**0.958**
QT21 De-En	Guu	0.47	0.37	0.32	0.30	QT21 De-En MQM	0.925	0.896	0.933
	Yin	**0.57**	**0.49**	**0.57**	**0.49**		0.972	0.952	0.964
	EVE	0.53	0.45	0.54	0.46		**0.999**	**0.992**	**0.992**

Table 3: Result of the intrinsic and extrinsic evaluations on our datasets, as defined by the PEER framework.

the edits in the latent vector, which is probably because it depends only on the tokens that were modified, in an unordered manner.

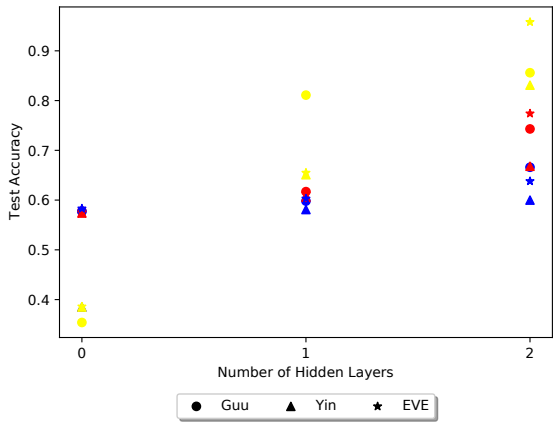

Figure 2: Effect of model depth on the accuracy on the test set of the extrinsic evaluation as a function of the number of hidden layers of the classifier. WikiAtomicSample → WikiEditsMix in red, WikiEditsMix→ WikiEditsMix in blue and Lang 8→ WI + Locness in yellow. Results on post editing are omitted for clarity.

Performance differences across settings are suggestive of the intrinsic difficulty of the task on each case. This is likely related to the nature of each dataset, which is evidenced in features such as average sentence length and vocabulary size. In this context, it is interesting to see models pre-trained on WikiAtomicSample outperforming models trained on WikiEditsMix. In this case, we think the much longer average sentence length may have hindered the learning process, since it is known that without the attention component, RNNs struggle on longer inputs (Bahdanau, Cho, and Bengio 2015).

In terms of the extrinsic evaluation, we see that our model obtains better performance in three out of the four settings, which we believe validates the effectiveness of our approach and shows that the information contained in our learned representations can actually be useful for downstream tasks. To select the best classifier, on each case we studied how performance varies with the depth of the classifier.

Finally, as Figure 2 shows, we see that at low depths, performance is poor, and differences across models tend to be small, suggesting that the classifiers are not capable of using the information stored in the vectors. Meanwhile, an increase in depth benefits all models, and as expected, it also allows us to clearly see the superiority of certain encoders. In this context, we think good results on WI + Locness and QT21 De-En MQM suggest that the labels in these are strongly correlated with the information stored in the edit representations, which is in agreement with the label nature (mostly related to the presence of certain misspelled/mistranslated terms). Conversely, the lower performance on WikiEditsMix suggests that a richer understanding of the edit semantics is needed.

Conclusions

In this paper, we have introduced a model that employs variational inference to learn a continuous latent space of vector representations to capture the underlying semantic information with regard to the document editing process. We have also introduced a set of downstream tasks specifically designed to evaluate the quality of edit representations, which we name PEER. We have utilized these to evaluate our model, compare it to relevant baselines, and offer empirical evidence supporting the effectiveness of our approach. We hope the development of PEER will help guide future research in this problem by providing a reliable programmatic way to test the quality of edit representations.

Acknowledgements

We are grateful to the NVIDIA Corporation, which donated two of the GPUs used for this research. We also thank Jorge Balazs, Pablo Loyola, Alfredo Solano, and Francis Zheng for their useful comments.

References

Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; Zitnick, C. L.; and Parikh, D. 2015. VQA: Visual Question Answering. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 2425–2433. Santiago, Chile: IEEE. ISBN 978-1-4673-8391-2. doi:10.1109/ICCV.2015.279.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*. San Diego, California. URL http://arxiv.org/abs/1409.0473.

Bahuleyan, H.; Mou, L.; Vechtomova, O.; and Poupart, P. 2018. Variational Attention for Sequence-to-Sequence Models. In *Proceedings of the 27th International Conference on Computational Linguistics*, 1672–1682. Santa Fe, New Mexico, USA: Association for Computational Linguistics.

Bowman, S. R.; Vilnis, L.; Vinyals, O.; Dai, A.; Jozefowicz, R.; and Bengio, S. 2016. Generating Sentences from a Continuous Space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 10–21. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/K16-1002. URL https://www.aclweb.org/anthology/K16-1002.

Bryant, C.; Felice, M.; Andersen, Ø. E.; and Briscoe, T. 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*, 52–75. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/W19-4406.

Faruqui, M.; Pavlick, E.; Tenney, I.; and Das, D. 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*, 305–315. Brussels, Belgium: Association for Computational Linguistics. URL http://aclweb.org/anthology/D18-1028.

Góis, A.; and Martins, A. F. T. 2019. Translator2Vec: Understanding and Representing Human Post-Editors. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, 43–54. Dublin, Ireland: European Association for Machine Translation.

Granger, S. 2014. *The Computer Learner Corpus: A Versatile New Source of Data for SLA Research*. Routledge. ISBN 978-1-315-84134-2. doi:10.4324/9781315841342-1.

Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 6645–6649. doi:10.1109/ICASSP.2013.6638947. ISSN: 2379-190X.

Graves, A.; and Schmidhuber, J. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18(5): 602–610. ISSN 0893-6080. doi:10.1016/j.neunet.2005.06.042. URL http://www.sciencedirect.com/science/article/pii/S0893608005001206.

Gupta, A.; Agarwal, A.; Singh, P.; and Rai, P. 2018. A Deep Generative Framework for Paraphrase Generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Guu, K.; Hashimoto, T. B.; Oren, Y.; and Liang, P. 2018. Generating Sentences by Editing Prototypes. *Transactions of the Association for Computational Linguistics* 6: 437–450. doi:10.1162/tacl_a_00030.

Guzman, E.; Azócar, D.; and Li, Y. 2014. Sentiment Analysis of Commit Comments in GitHub: An Empirical Study. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, 352–355. New York, NY, USA: ACM. ISBN 978-1-4503-2863-0. doi:10.1145/2597073.2597118. URL http://doi.acm.org/10.1145/2597073.2597118.

Halfaker, A.; and Geiger, R. S. 2020. ORES: Lowering Barriers with Participatory Machine Learning in Wikipedia. *Proc. ACM Hum.-Comput. Interact.* 4(CSCW2). doi:10.1145/3415219. URL https://doi.org/10.1145/3415219.

Jiang, S.; Armaly, A.; and McMillan, C. 2017. Automatically generating commit messages from diffs using neural machine translation. 135–146. IEEE. ISBN 978-1-5386-2684-9. doi:10.1109/ASE.2017.8115626. URL http://ieeexplore.ieee.org/document/8115626/.

Jiang, S.; and McMillan, C. 2017. Towards Automatic Generation of Short Summaries of Commits. 320–323. IEEE. ISBN 978-1-5386-0535-6. doi:10.1109/ICPC.2017.12. URL http://ieeexplore.ieee.org/document/7961530/.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]* URL http://arxiv.org/abs/1312.6114. ArXiv: 1312.6114.

Li, B.; He, J.; Neubig, G.; Berg-Kirkpatrick, T.; and Yang, Y. 2019. A Surprisingly Effective Fix for Deep Latent Variable Modeling of Text. 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)*, 3601–3612. Hong Kong, China: Association for Computational Linguistics. doi:10.18653/v1/D19-1370.

Little, D. 2006. The Common European Framework of Reference for Languages: Content, purpose, origin, reception and impact. *Language Teaching* 39(3): 167–190. doi:10.1017/S0261444806003557.

Liu, Q.; Liu, Z.; Zhu, H.; Fan, H.; Du, B.; and Qian, Y. 2019. Generating Commit Messages from Diffs Using Pointer-generator Network. In *Proceedings of the 16th International Conference on Mining Software Repositories*, MSR '19, 299–309. Piscataway, NJ, USA: IEEE Press. doi:10.1109/MSR.2019.00056. URL https://doi.org/10.1109/MSR.2019.00056. Event-place: Montreal, Quebec, Canada.

Liu, Z.; Xia, X.; Hassan, A. E.; Lo, D.; Xing, Z.; and Wang, X. 2018. Neural-machine-translation-based commit message generation: how far are we? In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, 373–384. Montpellier, France: Association for Computing Machinery. ISBN 978-1-4503-5937-5. doi:10.1145/3238147.3238190. URL https://doi.org/10.1145/3238147.3238190.

Loyola, P.; Marrese-Taylor, E.; Balazs, J.; Matsuo, Y.; and Satoh, F. 2018. Content Aware Source Code Change Description Generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, 119–128. Tilburg University, The Netherlands: Association for Computational Linguistics.

Loyola, P.; Marrese-Taylor, E.; and Matsuo, Y. 2017. A Neural Architecture for Generating Natural Language Descriptions from Source Code Changes. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 287–292. Vancouver, Canada: Association for Computational Linguistics. URL http://aclweb.org/anthology/P17-2045.

Marrese-Taylor, E.; Loyola, P.; and Matsuo, Y. 2019. An Edit-centric Approach for Wikipedia Article Quality Assessment. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, 381–386. Hong Kong, China: Association for Computational Linguistics. doi:10.18653/v1/D19-5550. URL https://www.aclweb.org/anthology/D19-5550.

Miltner, A.; Gulwani, S.; Le, V.; Leung, A.; Radhakrishna, A.; Soares, G.; Tiwari, A.; and Udupa, A. 2019. On the fly synthesis of edit suggestions. *Proceedings of the ACM on Programming Languages* 3(OOPSLA): 143:1–143:29. doi:10.1145/3360569. URL https://doi.org/10.1145/3360569.

Mizumoto, T.; Hayashibe, Y.; Komachi, M.; Nagata, M.; and Matsumoto, Y. 2012. The Effect of Learner Corpus Size in Grammatical Error Correction of ESL Writings. In *Proceedings of COLING 2012: Posters*, 863–872. Mumbai, India: The COLING 2012 Organizing Committee.

Napoles, C.; Sakaguchi, K.; Post, M.; and Tetreault, J. 2015. Ground Truth for Grammatical Error Correction Metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 588–593. Beijing, China: Association for Computational Linguistics. doi:10.3115/v1/P15-2097.

Ng, H. T.; Wu, S. M.; Briscoe, T.; Hadiwinoto, C.; Susanto, R. H.; and Bryant, C. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, 1–14. Baltimore, Maryland: Association for Computational Linguistics. doi:10.3115/v1/W14-1701.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning Representations by Back-propagating Errors. *Nature* 323(6088): 533–536. doi:10.1038/323533a0. URL http://www.nature.com/articles/323533a0.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision* 115(3). ISSN 1573-1405. doi:10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

Sarkar, S.; Reddy, B. P.; Sikdar, S.; and Mukherjee, A. 2019. StRE: Self Attentive Edit Quality Prediction in Wikipedia. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3962–3972. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/P19-1387. URL https://www.aclweb.org/anthology/P19-1387.

Sennrich, R.; Haddow, B.; and Birch, A. 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)*, 1715–1725. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/P16-1162. URL http://aclweb.org/anthology/P16-1162.

Specia, L.; Harris, K.; Burchardt, A.; Turchi, M.; Negri, M.; and Skadina, I. 2017. Translation Quality and Productivity: A Study on Rich Morphology Languages. 55–71. URL https://cris.fbk.eu/handle/11582/313118#.XiFXEeGRVGM.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. 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*, 353–355. Brussels, Belgium: Association for Computational Linguistics. doi:10.18653/v1/W18-5446. URL https://www.aclweb.org/anthology/W18-5446.

Yannakoudakis, H.; Andersen, Ø. E.; Geranpayeh, A.; Briscoe, T.; and Nicholls, D. 2018. Developing an Automated Writing Placement System for ESL Learners. *Applied Measurement in Education* 31(3): 251–267. ISSN 0895-7347. doi:10.1080/08957347.2018.1464447.

Yin, P.; Neubig, G.; Allamanis, M.; Brockschmidt, M.; and Gaunt, A. L. 2019. Learning to Represent Edits. In *Proceedings of the 7th International Conference on Learning Representations*. URL https://openreview.net/forum?id=BJl6AjC5F7.

Zhang, B.; Xiong, D.; Su, J.; Duan, H.; and Zhang, M. 2016. Variational Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 521–530. Austin, Texas: Association for Computational Linguistics. doi:10.18653/v1/D16-1050. URL https://www.aclweb.org/anthology/D16-1050.

Zhao, R.; Bieber, D.; Swersky, K.; and Tarlow, D. 2019. Neural Networks for Modeling Source Code Edits. In *Proceedings of the 7th International Conference on Learning Representations*. URL https://openreview.net/forum?id=Sklr9i09KQ&noteId=Sklr9i09KQ.