A Closer Look at k-Nearest Neighbors Grammatical Error Correction

Justin Vasselli and **Taro Watanabe**
Nara Institute of Science and Technology
{vasselli.justin_ray.vk4, taro}@is.naist.jp

Abstract

In various natural language processing tasks, such as named entity recognition and machine translation, example-based approaches have been used to improve performance by leveraging existing knowledge. However, the effectiveness of this approach for Grammatical Error Correction (GEC) is unclear. In this work, we explore how an example-based approach affects the accuracy and interpretability of the output of GEC systems and the trade-offs involved. The approach we investigate has shown great promise in machine translation by using the k nearest translation examples to improve the results of a pretrained Transformer model. We find that using this technique increases precision by reducing the number of false positives, but recall suffers as the model becomes more conservative overall. Increasing the number of example sentences in the datastore does lead to better performing systems, but with diminishing returns and a high decoding cost. Synthetic data can be used as examples, but the effectiveness varies depending on the base model. Finally, we find that finetuning on a set of data may be more effective than using that data during decoding as examples.

1 Introduction

Grammatical Error Correction (GEC) is the task of identifying and correcting grammatical mistakes in ungrammatical text. While it can be used to assist native speakers as well, it is frequently applied to text written by language learners, and can be used pedagogically to help them improve their writing skills. Providing feedback on grammatical errors in a learner's writing allows them to learn from their mistakes and improve their writing over time. For this feedback to be effective, it must be interpretable to the learner.

GEC models are often based on neural machine translation (NMT) models and treated as similar to sequence-to-sequence (seq2seq) tasks (Junczys-Dowmunt et al., 2018; Kiyono et al., 2019). Unfortunately, Transformer-based seq2seq models produce corrections that are uninterpretable, because they simply output a corrected sentence without any indication of how or why elements of the sentence were corrected. This lack of interpretability can make it difficult for learners to understand the nature of their mistakes and how to avoid them in the future. In contrast, example-based approaches to GEC can provide a motivating example for each correction, making the results more interpretable and therefore more helpful for learners. This can make the difference between a learner simply correcting a mistake and actually understanding why it is a mistake and how to avoid it in the future.

Example-based, or instance-based, methods have recently been applied to tasks across the field such as named entity recognition (Ouchi et al., 2020), summarization (Cao et al., 2018), and machine translation (Khandelwal et al., 2020). In their recent work, Kaneko et al. (2022) presented their findings on the interpretability of GEC corrections using human evaluation and three example selection methods: token-based retrieval, BERT-based retrieval, and their example-based grammatical error correction (EB-GEC) system. The study found that presenting examples is more useful to learners than providing none, with EB-GEC providing the most useful examples for language learners' understanding and acceptance of the model corrections.

EB-GEC is based on the k-nearest neighbors approach to machine translation proposed by Khandelwal et al. (2020). This method uses a datastore constructed from a set of example sentence pairs during the decoding of the vanilla Transformer. At each timestep, the vector being passed into the final feedforward network of the decoder is used to locate the k nearest neighbor examples in the datastore. This vector represents the translation context, which is composed of the ungrammatical sentence plus the prefix of the output. The datas-

Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023), pages 220–231

tore itself is constructed from a corpus, authentic or synthetic, of training examples. One entry into the datastore is made for each token of the corrected sentence of each example pair, using the encoded translation context as the key, and the ground-truth token as the value. During inference, the retrieved values of the k nearest contexts form a distribution of target tokens. The distribution of target tokens collected from the datastore is then interpolated with the distribution from the base Transformer. In this way, the output of the vanilla Transformer is influenced by the most similar examples from the datastore, and motivating examples are returned for each token of the output.

Khandelwal et al. (2020) reports high BLEU score gains in resource-rich languages with large databases, but less impressive performance in low-resource languages. Treating GEC as a low-resource machine translation task was proposed by Brockett et al. (2006) and has resulted in many high performing systems (Junczys-Dowmunt et al., 2018). However, there are key differences between grammatical error correction and machine translation. GEC is a monolingual task, where both input and output share a vocabulary, and a large number of tokens from the input sentence remain unchanged in the output sentence. This difference may very well affect the viability of using k-nearest neighbors for grammatical error correction.

Kaneko et al. (2022) found that their EB-GEC system improved the $F_{0.5}$ score on three out of four test sets, relative to the vanilla Transformer. However, there are several factors to consider about these results. The three test sets that performed better using EB-GEC came from datasets with training splits used for both training the vanilla Transformer and as the datastore of example sentences. The fourth test set that performed better with the vanilla Transformer did not have any representation in the datastore or the training. This may indicate that EB-GEC is not generalizable, as in a way, the three test sets with better scores can be thought of as in-domain, because they had similar sentences used in the datastore. It is possible that using example sentence pairs produced in a different context would produce lower scores on the test sets. It is worth investigating how using different data for the example corrections than during training affects the results.

The reported scores of this system are lower than those reported by Kiyono et al. (2019), using a vanilla Transformer pretrained on synthetic data. It is unclear whether applying the same kNN method to a higher performing Transformer would yield the same gains. A detailed analysis of the costs and benefits of using the k-nearest neighbor approach to grammatical error correction as proposed by Kaneko et al. (2022) has yet to be carried out, but the results of the initial experiments are worth investigating further.

This work aims to address some outstanding questions about the effectiveness of k-nearest neighbors for grammatical error correction (kNN-GEC). Specifically, we seek to determine whether kNN-GEC always improves the performance of the base Transformer, or whether the impact varies. Additionally, we investigate how the size of the datastore affects performance, and whether synthetic data can be used to bolster the datastore. We also explore whether using synthetic data produces the same level of interpretability. Furthermore, we examine how the choice of data for the datastore impacts the effectiveness of the model on test sets. Finally, we compare the effectiveness of finetuning a Transformer on a set of data versus using that data as the datastore for kNN-GEC.

We found that the effectiveness of kNN-GEC varies depending on the base Transformer. Higher performing Transformers show little to no improvement. Using synthetic data does not appear to impact the interpretability of the corrections, and can be used to increase the size of the datastore. However, very large datastores may not improve the system's performance enough to warrant the increase in computational cost. Bolstering the datastore with error-targeted example sentences does not seem to be a viable way of improving the system's performance on those error types or in general. We also found that finetuning a Transformer on a set of in-domain data can be more effective than using kNN-GEC for in-domain data.

2 Prior Work

2.1 Example-based machine translation

First proposed by Nagao (1984), using examples to anchor text generation has been explored in many other tasks from summarization (Cao et al., 2018) to response generation (Weston et al., 2018). In machine translation, this process requires two steps: retrieving a relevant translation example, and using that to guide the translation of a new sentence.

Retrieving relevant example pairs is most often

done by comparing the source sentence to a datastore of source-target example pairs, and retrieving the k nearest neighbors of the source sentence. Distance may be calculated with edit-distance (Bulte and Tezcan, 2019; Hossain et al., 2020; Zhang et al., 2018), sentence embeddings (Tezcan et al., 2021; Wu et al., 2019), or a combination of both (Xu et al., 2020).

There is variety in how the retrieved example is used to produce the output sentence. Once the nearest examples are retrieved, they must be integrated into the generation. A common approach is to train a Transformer with a concatenated input of the input text and one or more retrieved target sentences (the input sentences for the translation examples are only used in retrieval) (Bulte and Tezcan, 2019; Tezcan et al., 2021; Hossain et al., 2020). This method finds the most similar examples up front, and uses a standard encoder decoder to generate the hypothesis.

Other methods involve using retrieved examples to alter the probability distribution of tokens during the autoregressive decoding. Zhang et al. (2018) proposed increasing the probabilities of the n-grams found in the output of the translation example at each timestep of decoding. Khandelwal et al. (2020) proposed an approach that could use a pre-trained seq2seq Transformer and improve it's performance by retrieving examples during decoding. The nearest neighbor machine translation (kNN-MT) system uses the decoder of a pre-trained Transformer model to generate translation context vectors for each target token of the example sentences. The translation context is the source sentence and the partially generated target sentence. The vector that is passed into the final feedforward network of the decoder is considered to represent the full translation context at each time step. This vector serves as the key with the target token as the value in an example datastore of key-value pairs.

The system translates new pieces of text by consulting the datastore at each decoding step and finding the k nearest neighbors of the vector and weighting the possible output tokens by the L2 distance to the nearest neighbor keys. The authors reported significant gains using this method, especially on language pairs with a considerable number of example sentences, such as DE-EN, ZH-EN, and EN-ZH with datastore sizes of 5.56, 1.19, and 1.13 billion translation context-token pairs respectively.

2.2 Example-based grammatical error correction

Kaneko et al. (2022) applied kNN-MT to grammatical error correction in their EB-GEC system. The authors conducted a study using human evaluation to demonstrate that the example sentences retrieved through the decoding process improve the interpretability of the results for language learners as compared to the closest sentence pairs using edit distance or BERT-based retrieval.

EB-GEC showed mixed results compared to the vanilla Transformer model. The authors report improved performance using the k nearest neighbors at inference time on CoNLL14 (Ng et al., 2014), the test data of the BEA2019 shared task (Bryant et al., 2019), and FCE (Yannakoudakis et al., 2011), but not JFLEG (Napoles et al., 2017). As JFLEG is the only one of these test sets to focus on fluency, these results are interpreted to mean that the approach is successful at increasing accuracy of error corrections, but may not be as effective at improving the fluency of the sentence. An alternate explanation could be that the three test sets that performed better using EB-GEC had training splits that were used to train the model and also that contributed to the datastore of example sentences. JFLEG, which performed better with the vanilla model, did not have any representation in the datastore or the training.

Despite the improved accuracy on most of the test sets, EB-GEC did not perform strongly compared to the state-of-the-art tagging-based approaches to GEC. One possible reason for this may be the size of the datastore being insufficient. With only 600,000 sentences generating 17 million key-value pairs for the datastore, there may not be enough examples for the kNN system to retrieve from.

2.3 Synthetic examples

While most kNN-MT systems reuse bilingual training data for the datastore, it is possible that using different data or even synthetic data for translation examples could yield better results. Deguchi et al. (2022) showed that using a larger back-translated monolingual corpus for the datastore can outperform a smaller training data corpus. The reason for this has yet to be explored thoroughly. It may be simply due to the larger number of examples for the system to draw from, or it may be because the Transformer has already learned from the train-

	C4+BEA+CWEB	EB-GEC Base	PretLargeSSE
BEA-train (Bryant et al., 2019)	finetuning/datastore	training/datastore	finetuning/datastore
CWEB (Flachs et al., 2020)	finetuning	-	-
JFLEG (Napoles et al., 2017)	finetuning	-	-
gec-pseudodata (Kiyono et al., 2019)	-	-	pretraining
$C4_{200M}$ (Stahlberg and Kumar, 2020)	pretraining/datastore	datastore	datastore

Table 1: How each dataset used for training, finetuning, or as the datastore was used across the three models.

ing examples and the synthetic data provides novel translation examples.

3 Experiments

3.1 The Vanilla Transformers

In order to investigate whether the impact of kNN-GEC varies depending on the base transformer, we applied this approach to three base models trained differently on different data: C4+BEA+CWEB, EB-GEC Base, and PretLargeSSE.

C4+BEA+CWEB was trained from scratch using a combination of synthetic and authentic data with the base Transformer architecture. It was first pretrained on the synthetic corpus $C4_{200M}$ (Stahlberg and Kumar, 2020), which is a cleaned version of the Common Crawl. The source sentences were corrupted from the targets using a tagged seq2edits corruption method. C4+BEA+CWEB was then finetuned on BEA-train, which is composed of the training split of the First Certificate in English corpus (FCE), Lang-8 Corpus of Learner English (Tajiri et al., 2012), National University of Sinagpore Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013), and the training split of Write & Improve + LOCNESS (Bryant et al., 2019).

C4+BEA+CWEB was also finetuned on the development split of JFLEG (Napoles et al., 2017), and CWEB (Flachs et al., 2020). JFLEG is a fluency oriented corpus which contains larger sentence edits than BEA-train. CWEB is a corpus of edits made to websites and contains much fewer and smaller sentences edits than BEA-train.

The second Transformer, EB-GEC Base was trained directly on BEA-train, with no synthetic pretraining. It was trained using the data and settings outlined in Kaneko et al. (2022)[1].

The third Transformer, PretLargeSSE was pretrained on the gec-psuedodata synthetic data Kiyono et al. (2019)[2]. It was then fine-tuned on BEA-train.

[1] https://github.com/kanekomasahiro/eb-gec

[2] https://github.com/butsugiri/gec-pseudodata

Datastore name	Sentences	Tokens
BEA-train	1.3M	16-17M
Synthetic 2M	2M	55-78M
Synthetic 20M	20M	547-777M
Synthetic 40M	40M	1-1.6B
Synthetic Full	147M	4-7B

Table 2: The size of the datastores as measured by number of example pairs and number of resulting entries in the datastore, which is equivalent to the number of tokens of the target sentences.

We provide a summary of the datasets used, which are listed in Table 1, and the detailed hyperparameters of all three base models in Appendix A.

3.2 Evaluation

Testing was done on the CoNLL-2014 test data (Ng et al., 2014) using M^2 (Dahlmeier and Ng, 2012), the BEA-2019 shared task test data (Bryant et al., 2019) and FCE test data (Yannakoudakis et al., 2011) using ERRANT (Bryant et al., 2017), and JFLEG (Napoles et al., 2017) using GLEU (Napoles et al., 2015). M^2 and ERRANT report $F_{0.5}$ scores.

3.3 Datastores

We conducted experiments using datastores made from BEA-train and subsets of different sizes from $C4_{200M}$ to understand how datastore size affects performance, and whether synthetic data can improve performance. The data was preprocessed using the same method as the training data of the respective model, which leads to different sizes of each datastore depending on the vocabulary size used for subword tokenization, which varies between the three base models. The datastore ranges for the systems are noted in Table 2. C4+BEA+CWEB has a larger vocabulary size (128k), resulting in smaller datastores. The other baselines have a vocabulary size of 8k and larger datastores.

The vector that is passed into the final feedforward network of the decoder is considered to be the hidden state of the context and is used as the key

	CoNLL14 M^2 $F_{0.5}$	BEA2019 ERRANT $F_{0.5}$	FCE ERRANT $F_{0.5}$	JFLEG GLEU
C4+BEA+CWEB	47.84	47.51	41.51	49.47
+ BEA-train	44.55	48.89	42.77*	48.84
+ Synthetic 2M	49.47	50.67	44.32***	50.88***
+ Synthetic 20M	51.73***	53.15	43.90***	51.83***
+ Synthetic 40M	52.09***	54.92	45.63***	51.98***
+ Synthetic Full	**54.17*****	**55.69**	**45.67*****	**52.49*****
EB-GEC Base	**50.01**	48.44	40.18	55.65
+ BEA-train	49.68	**51.40**	42.00***	**56.26**
+ Synthetic 20M	48.75	47.77	**42.26*****	52.80
PretLargeSSE	62.11	65.17	51.73	**60.99**
+ BEA-train	61.73	**66.44**	53.76***	**60.99**
+ Synthetic 2M	61.79	60.90	53.08***	59.94
+ Synthetic 20M	**62.46**	61.82	53.50***	60.15
+ Synthetic 40M	62.07	65.97	**53.86*****	60.02

Table 3: Results on test sets using λ of 0.5. The best result of each system is bolded. p values were calculated on CoNLL14, FCE, and JFLEG between each kNN-GEC datastore and the base model using paired bootstrap resampling. $p < 0.05$ is denoted with *, $p < 0.01$ is denoted with **, and $p < 0.001$ is denoted with ***. BEA2019 test set is not released publicly, so we did not calculate the resampling for this data.

Computers ~~is~~ are the most important inventions in our ~~life~~ lives.

is → are	*invention → inventions*	*life → lives*
Trees ~~is~~ are the most spiritually advanced living beings on the Earth who are constantly in a ~~deflative~~ meditative state, and ~~substile~~ subtle energy is what they speak ~~like~~ as a language.	Bitcoin ~~it~~ is one of the most important inventions ~~along~~ in all of human history.	They're the ~~carrying~~ beginnings of AI everywhere in our ~~life~~ lives.

Table 4: An example of a sentence correction and the examples used to justify each correction from the synthetic datastore Synthetic 40M.

vector in the datastore, with the target token used as the value. The datastores were indexed using FAISS (Johnson et al., 2019)[3], with a training size of 5,242,880 and a chunk size of 10,000,000. Product quantization was applied to split the vectors into 64 subspaces and quantize each subspace. In addition, the vectors were clustered using k-means clustering into 131,072 clusters to speed up search.

During decoding, the k nearest vectors to the hidden state passed to the final feedforward network in the decoder are retrieved from the datastore. In this work, k is set to 16 and "nearest" is defined by shortest Euclidean distance.

Using the full synthetic datastore is computationally expensive, and the results on our initial experiments show marginal improvement from the next smaller datastore (40M). For this reason, the results of using the full synthetic datastore for kNN-GEC were not calculated for the other two systems. The results of each system are listed in Table 3.

Synthetic data was most effective when used with C4+BEA+CWEB. Even the smallest synthetic datastore improved the score compared to the non-synthetic datastore, and larger datastores resulted in even higher scores. However, the same gains were not seen by PretLargeSSE, which was the highest scoring base model. PretLargeSSE showed mixed results with the authentic datastore, and the synthetic datastore was less effective. It took at least 20M sentences for the synthetic datastore to perform as well as the much smaller authentic one.

3.4 Interpretability

One advantage of kNN-GEC is that it can provide motivating examples from the datastore for the corrections it makes. Kaneko et al. (2022) showed that the examples sentence pairs used during decoding provided more relevant models for learners than those retrieved by word overlap or BERT embeddings. As we investigate the effectiveness of using synthetic sentences in the datastore to improve the quality of model corrections, it is important to ensure that synthetic sentences can also serve as effective models for learners. To accomplish this, our code generates the kNN examples for each to-

[3] https://github.com/facebookresearch/faiss

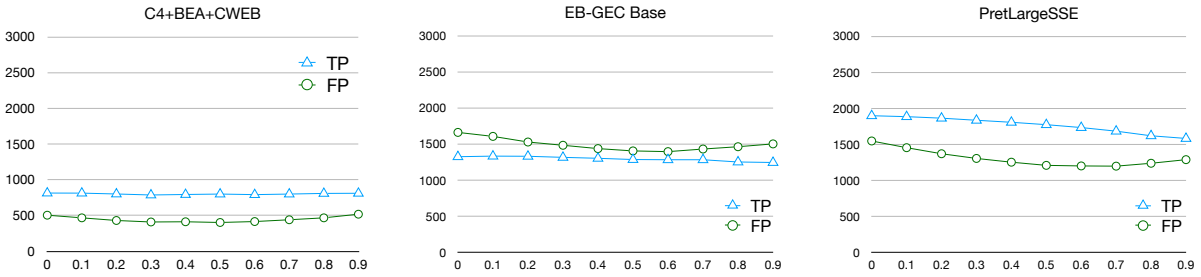

Figure 1: The number of true positives and false positives in the FCE test set for different values of λ using the BEA-train datastore.

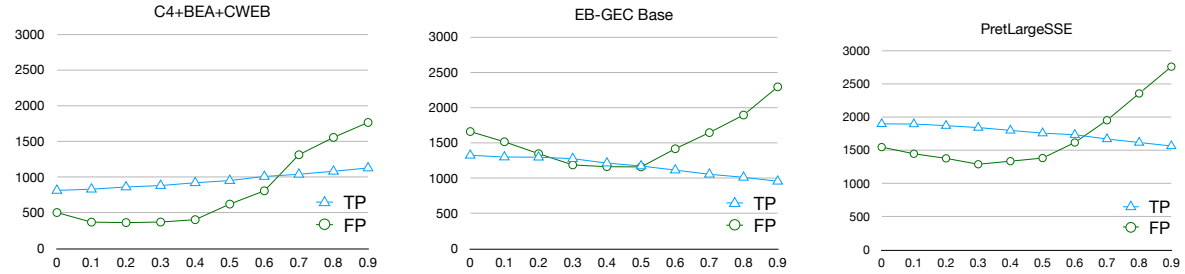

Figure 2: The number of true positives and false positives in the FCE test set for different values of λ using the Synthetic 20M datastore.

ken in the corrected sentence. In a post-processing step, we align the source sentence and corrected sentence using ERRANT (Bryant et al., 2017) and extract the nearest example for each corrected token. To explore the interpretability of the synthetic datastore, we present a randomly selected sentence correction with motivating examples in Table 4.

In this single example, the synthetic datastore provides reasonable examples for three corrections. A larger study measuring the effectiveness of synthetic data on the quality of the examples provided by the model is left for future work.

3.5 Impact of kNN on corrections

To investigate the effects of kNN-GEC on the models, we adjusted the hyperparameter λ. This parameter regulates the proportion of the probability distribution for the next token that comes from the datastore. kNN-GEC uses a linear interpolation between the output of the kNN token distribution, p_{kNN}, and the vanilla decoder, p_{GEC}, as follows:

$$P(y_i|x, y_{1:i-1}) = \lambda p_{\text{kNN}}(y_i|x, y_{1:i-1}) + (1-\lambda)p_{\text{GEC}}(y_i|x, y_{1:i-1}) \quad (1)$$

The hyperparameter λ in equation 1 is used to balance the probability distribution generated by the example sentences and that generated by the base Transformer. Setting λ to 0 is equivalent to using the vanilla Transformer without kNN-GEC. The larger the λ, the more the system will use the retrieved examples when generating the next token. However, using only the examples can lead to errors, so we did not calculate λ of 1. Figure 1 shows the number of true positives and false positives generated in the FCE test data by each system using the BEA-train datastore.

In general, the use of kNN-GEC does not increase the number of corrections made until λ values exceed 0.5. In all three systems, using the BEA-train datastore leads to a more conservative approach to corrections, which results in fewer incorrect changes being made. One possible explanation is that the method of retrieving example sentence pairs returns pairs that have similar meanings or are on similar topics but may not necessarily contain the same errors. In the absence of an error, GEC will copy from the input sentence to the hypothesis sentence. If the k retrieved sentence pairs do not contain the error, kNN-GEC may copy more and correct less.

Figure 1 and Figure 2 show how λ affects the number of true positives and false positives in the FCE test set. These figures show that the gain in $F_{0.5}$ score below $\lambda = 0.5$ is due to an increase in precision resulting from a decrease in false positives, rather than an increase in true positives. While the number of true positives does not decrease rapidly, the number of false positives does, leading to better model performance despite the decrease in recall.

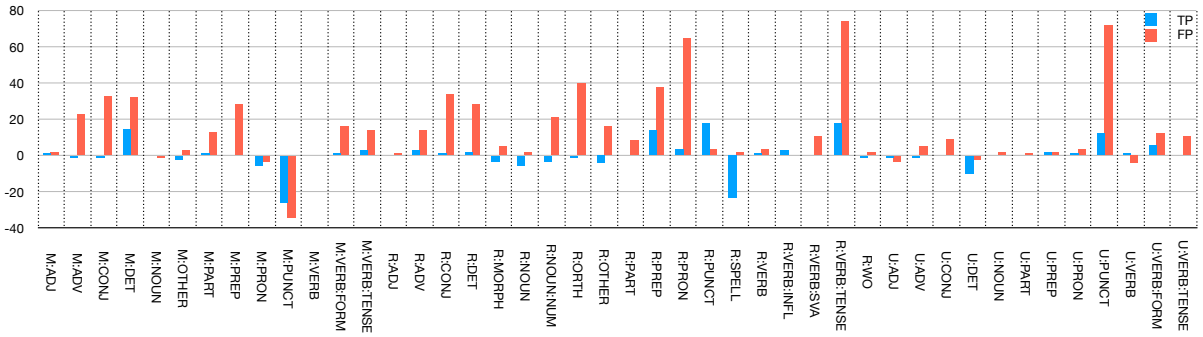

Figure 3: The difference in the number of true positives (TP) and false positives (FP) generated using the corresponding targeted datastore compared to the vanilla Transformer.

Interestingly, the number of true positives increases when using the synthetic datastore with C4+BEA+CWEB, as shown in figure 2. However, this effect doesn't transfer to PretLargeSSE. This may be because C4+BEA+CWEB is already a very conservative model and performs poorly in comparison to PretLargeSSE. C4+BEA+CWEB may not be as effective at producing corrections as PretLargeSSE, and having more example sentences to use improves the results. Further work is needed to determine the reason for this difference.

All systems exhibit an increase in false positives when $\lambda > 0.5$, with the synthetic datastore demonstrating this most dramatically. Generally, using $\lambda = 0.4$ resulted in the best balance of precision and recall. However, as many papers use 0.5 as the balancing point between the kNN distribution and the vanilla Transformer, the rest of the experiments in this work use $\lambda = 0.5$.

3.6 Error type targeted datastores

To observe how changing the error distribution in the datastore impacts the effectiveness of the model on that error type, as well as the performance of the system as a whole, we conducted experiments using datastores that contained examples with a single error type. We extracted 10,000 sentences from synthetic data for most[4] of the ERRANT error tags. The ERRANT error tags consist of an error category and error type. The error categories are Missing (M), Replacement (R), and Unnecessary (U). The error types include Adjective (ADJ), Adverb (ADV), Morphology (MORPH), Orthography (ORTH), and more. There are a total of 54 error tags, of which 8 didn't have enough data to generate a targeted datastore. For each of the remaining 46 error tags, a datastore was constructed with 10,000 sentences that contained only that error. We used the target datastore with PretLargeSSE and tested it on FCE-test with a λ value of 0.5.

[4]Some errors were very rare and did not occur more than a handful of times in the data.

Datastore	TP	FP	P	R	$F_{0.5}$
None	1,896	1,548	55.05	41.68	51.73
BEA	1,819	1,273	58.83	39.99	53.76
+30K	1,355	921	59.53	29.79	49.62

Table 5: A comparison of different datastores, BEA-train+30K includes 10K synthetic example pairs each of Missing Adjectives, Missing Particles, and Replacing Verb Inflections corrections

Using much smaller targeted datastores alone lowers both precision and recall compared to the base Transformer and performs much worse than the BEA-train datastore. Instead of looking at the overall performance, we examine how the number of true positives and false positives changes within the targeted error type compared to the base Transformer. Figure 3 illustrates the difference in the number of true or false positives between using the targeted datastore and the vanilla Transformer.

Using an error-targeted datastore tends to increase the number of false positives for a particular error, likely due to the system overapplying the correction. Surprisingly, for many error types, the targeted datastore does not increase the number of true positives. However, it does increase the accuracy of correcting missing determiners (M:DET), incorrect prepositions (R:PREP), verb tense (R:VERB:TENSE), and unnecessary punctuation (U:PUNCT). The number of false positives often increases much more than the number of true positives, resulting in lower precision. Replacing incorrect punctuation (R:PUNCT) is an exception, as it can increase precision without significantly increasing false positives.

It is reasonable to assume that a datastore con-

	CoNLL14	BEA2019	FCE	JFLEG
Finetuned	**62.11**	**65.17**	**51.73**	60.99
kNN-GEC	56.34	58.58	47.06	**61.92**

Table 6: Results of finetuning a Transformer on BEA-train compared to using kNN-GEC with BEA-train as the datastore.

taining only one type of error will perform poorly on a test set with many diverse errors, since there are no examples for any of the other types of corrections to use as a model. However it's not immediately obvious if targeted datastores could be used to supplement a base datastore in order to compensate for low performance with a particular error type. Preliminary experiments suggest that this approach may not be effective for all error types. Table 5 presents the results of adding 30,000 synthetic pairs to the BEA-train datastore to address three types of low-performing errors: missing adjectives, missing particles, and incorrect verb inflections. These three types were selected because using kNN-GEC with the BEA-train datastore alone did not produce any more true positives than the base Transformer, but each of these three types saw a slight increase in true positives in their respective categories when the targeted datastore was used.

Adding the 30,000 new sentence pairs decreased the number of false positives, but it also significantly reduced the number of true positives, making the overall system more conservative. This resulted in an increase in precision. Unfortunately, the decrease in recall lowered the overall $F_{0.5}$ score to less than that of the vanilla Transformer.

3.7 Finetuning vs kNN-GEC

To determine the effectiveness of finetuning a Transformer on data versus using that data as a kNN-GEC datastore, we tested using a checkpoint of PretLargeSSE before the finetuning phase (Kiyono et al., 2019). We applied the kNN-GEC method with this pretrained-only model, using BEA-train as the datastore. The results of this experiment are shown in Table 6.

The Transformer model that was finetuned outperformed the pretrained-only model using kNN-GEC on three of the four test sets. The three datasets that performed better with finetuning (CoNLL14, BEA2019, and FCE) all have training sets used in the finetuning or kNN-GEC. This suggests that finetuning is more effective for in-domain test sets.

4 Discussion

Overall, kNN-GEC makes a base GEC system more conservative about making corrections, which lowers its recall. This is likely due to the fact that the chosen examples may not contain a similar error to the one being corrected, but are closer in content to the example pair. The retrieval method involves comparing embedded vectors from the decoder, which contain a mixture of information about the syntax and semantics. As a result, there can be times when the closest sentence pair to the one being decoded overlaps more heavily on semantics than syntax. It is likely that the tendency to retrieve example sentence pairs that are similar in content but not grammatically incorrect promotes more copying, or more conservative corrections, as the target word may not even be incorrect in the example pair.

This is a key difference between the tasks of machine translation and grammatical error correction. Machine translation must generate the appropriate content words for the translation, while GEC mostly uses the content words from the source sentence, or a different form of the existing word. MT may benefit from the influence of the kNN probability distribution on word choice because success in MT often includes selecting the correct content word for the context. In the case of GEC, however, example sentence pairs may not contain the same grammatical errors as the query sentence, which leads to more copying and less correcting.

5 Conclusion

In this work, we investigate how using training examples during decoding with the kNN-GEC method affects the precision and recall of grammatical error correction. We used three different base models and found that the effectiveness of kNN-GEC varies greatly depending on the base model. In general, this method makes models more conservative in making corrections, improving precision but lowering recall. Synthetic data can be used to increase the size of the datastore, but its effectiveness depends on the base model. While kNN-GEC using authentic or synthetic datastores increases the interpretability of corrections for learners, this comes with the trade-off of fewer corrections made and a longer decoding time. We also explored the effect of the hyperparameter λ on the performance of the kNN-GEC method. A value of 0.4 tended to produce the best balance of precision and recall,

though many papers use a value of 0.5. Finally, we compared the effectiveness of finetuning a Transformer on a set of data versus using that data as the datastore for kNN-GEC. Our results showed that finetuning the Transformer on the data generally outperformed using the data as a datastore.

Given that the more conservative corrections indicate that kNN-GEC is retrieving examples that are more semantically similar than those containing similar corrections, a future direction for kNN-GEC could be selecting the k nearest neighbors based on similarity of errors rather than similarity of content. This would require a target output that expresses the difference between the input sentence and the hypothesis sentence. During decoding, the necessary edits would be applied to change the example source to the example target token to the input. Future work could involve separating the syntax from the semantics of the encoded input sentence to retrieve the nearest neighbors with syntactically similar example sentences. This would help overcome the limitations of the kNN system with regards to making new corrections.

Limitations

The three base models used for the experiments were trained with different settings. As a result, it is challenging to understand the exact source of discrepancies between the results. Additionally, each of the three models used different subword tokenizations, resulting in variable datastore sizes. Although we have some hypotheses about why kNN affects GEC differently from MT, more experiments need to be conducted to confirm them.

References

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Sydney, Australia. Association for Computational Linguistics.

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.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

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.

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.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

Hiroyuki Deguchi, Kenji Imamura, Masahiro Kaneko, Yuto Nishida, Yusuke Sakai, Justin Vasselli, Huy Hien Vu, and Taro Watanabe. 2022. Naist-nict-tit wmt22 general mt task submission. In *Proceedings of the Seventh Conference on Machine Translation*, pages 244–250, Abu Dhabi. Association for Computational Linguistics.

Simon Flachs, Ophélie Lacroix, Helen Yannakoudakis, Marek Rei, and Anders Søgaard. 2020. Grammatical error correction in low error density domains: A new benchmark and analyses. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8467–8478, Online. Association for Computational Linguistics.

Nabil Hossain, Marjan Ghazvininejad, and Luke Zettlemoyer. 2020. Simple and effective retrieve-edit-rerank text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2532–2538, Online. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. 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 595–606, New Orleans, Louisiana. Association for Computational Linguistics.

Masahiro Kaneko, Sho Takase, Ayana Niwa, and Naoaki Okazaki. 2022. Interpretability for language learners using example-based grammatical error correction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7176–7187, Dublin, Ireland. Association for Computational Linguistics.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Nearest neighbor machine translation. *CoRR*, abs/2010.00710.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. 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 1236–1242, Hong Kong, China. Association for Computational Linguistics.

Makoto Nagao. 1984. A framework of a mechanical translation between japanese and english by analogy principle.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 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)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Hiroki Ouchi, Jun Suzuki, Sosuke Kobayashi, Sho Yokoi, Tatsuki Kuribayashi, Ryuto Konno, and Kentaro Inui. 2020. Instance-based learning of span representations: A case study through named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6452–6459, Online. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159, Online. Association for Computational Linguistics.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202, Jeju Island, Korea. Association for Computational Linguistics.

Arda Tezcan, Bram Bulté, and Bram Vanroy. 2021. Towards a better integration of fuzzy matches in neural machine translation through data augmentation. *Informatics*, 8(1).

Jason Weston, Emily Dinan, and Alexander Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, pages 87–92, Brussels, Belgium. Association for Computational Linguistics.

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.

Jitao Xu, Josep Crego, and Jean Senellart. 2020. Boosting neural machine translation with similar translations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1580–1590, Online. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

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.

A Hyperparameters

The detailed hyperparameters of the base Transformers and the settings used for generation.

	C4+BEA+CWEB	**EB-GEC Base**	**PretLargeSSE**
Architecture	Transformer Base	Transformer Big	Transformer Big
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$)		Pretrained with Adam, Fine-tuned with Adafactor
Learning Rate Schedule	Inverse square root decay	Inverse square root decay	Fixed
Warmup Steps	4,000	4,000	-
Dropout	0.2	0.3	0.1
FFN size	2096	4096	4096
Gradient Clipping	1.0	0.0	0.0
Label Smoothing	$\epsilon_{ls} = 0.1$	$\epsilon_{ls} = 0.1$	None
Layers	Encoder 6, Decoder 4	Encoder 6, Decoder 6	Encoder 6, Decoder 6
Mini-batch Size	4096 tokens	4096 tokens	unknown
Number of Updates	10,800 steps	20 epochs	unknown

Table 7: Hyperparameters of the vanilla Transformers.

	Generation settings
Length Penalty	1.0
Beam Size	5
Temperature	100
λ	0.5

Table 8: Settings for the kNN generation