Large-scale Semi-supervised Learning of Neural NLP Models
ニューラルネットワークを用いた自然言語処理のための大規模半教師あり学習

Shun Kiyono

Graduate School of Information Sciences
Tohoku University

This dissertation is submitted for the degree of
Doctor of Information Science

January 2022

Acknowledgements

本研究の遂行にあたり、多くの方々にご協力をいただきました。ここに、心より感謝の意を表します。

主指導教官である乾健太郎教授には、学部 3 年次での研究室配属から、6 年間の研究活動において、多大なご助言を頂戴しました。また、社会人として博士号の取得を目指すという非常に貴重な機会をいただきました。深く感謝いたします。

ご多忙の中、審査委員として本論文を査読してくださいました岡谷貴之教授、ならびに大関真之教授に深く感謝いたします。

鈴木潤教授には実装、実験、論文執筆や発表資料の作成など、研究活動に含まれるありとあらゆる営みについて細部までご指導いただきました。研究遂行のための技術に留まらず、もう一段メタな技術（ものの見方・考え方・取り組み方）を学ばせていただき、心より感謝しています。

東京工業大学の高瀬翔さんには、NTT コミュニーケーション科学基礎研究所でのインターンシップにおいて大変お世話になりました。心より感謝しております。研究の組み立て方、プログラミングのテクニックや論文の書き方だけではなく、けいはんなの美味しい食べ物やクラフトビールを、ときには夜遅くまでご指導いただき、ゼロから国際会議論文を仕上げることができました。また、現在も週次でミーティングする機会をいただき、活発かつ熱心に議論させていただいております。今後ともよろしくお願いいたします。

産業技術総合研究所の渡邉研斗さんには、学部 2 年生時の StepQi アドバンス創造工学研修において大変お世話になりました。回文生成という研究課題への半年間の取り組みは、私が自然言語処理やプログラミングの面白さに魅了されるきっかけでした。また、乾・岡崎研究室（当時）で研究に熱心に取り組んでいる研斗さんの姿は、座学中心のモノトーンな大学生活を送っていた私にとって非常に魅力的に映り、研修終了後に現在の研究室へ配属希望を出す大きな理由の一つとなりました。この場を借りて深く感謝申し上げます。

株式会社 Preferred Networks の小林颯介さんには、主に博士後期課程の研究で大変お世話になりました。心より感謝しております。学部 4 年に弊研究室に配属され

て以来、小林さんはずっと憧れの先輩です。週次のミーティングでの議論は大変刺激的で、私にとって他に代えがたい研究の糧になっています。今後ともよろしくお願いいたします。

乾研究室の伊藤拓海さんには、主に課外活動、つまりは週末の登山やスノーボードで大変お世話になりました。長い研究室生活において心中穏やかではいられないような時期もあった中で、良いリフレッシュの機会をいただきました。大変感謝しております。今後も一緒に遊びに行ってもらえると嬉しいです。

株式会社リクルートホールディングスの今野颯人さんには、研究室内部での共同研究でお世話になりました。私は今野さんのメンターという立場ではありましたが、実際には私のほうが多くのことを学ばせてもらったと思っています。深く感謝申し上げます。また東京で会いましょう。

菅原真由美さん、相澤知佳さん、磯部順子さんには、研究活動だけではなく私の未熟な事務処理能力を辛抱強くサポートしていただきました。深く感謝申し上げます。

山口健史さんには、研究室の計算機クラスタの管理で大変お世話になりました。無茶な計算機の使い方をしてご迷惑をおかけすることもありましたが、その度に手厚くサポートしていただきました。心より感謝いたします。

また、博士課程の研究を通して、私の大規模な実験を支えてくれた計算機クラスタである RAIDEN と miniRAIDEN、およびその管理者の方々に感謝いたします。DGX-1 と DGX-2 シリーズの演算能力なくしては、本博士論文は執筆できていなかったことは間違いありません。

最後に、乾研究室の皆様からは様々なご助言をいただき、研究生活を暖かく支えていただきました。心より感謝を申し上げます。本当にありがとうございました。

Abstract

Deep neural network (DNN) based approaches have achieved remarkable performance in multiple research fields, such as natural language processing, computer vision, and speech processing. However, it is widely acknowledged that DNNs are extremely data-hungry, i.e., in order to take full advantage of DNN, one needs to incorporate large-scale data into the training. Semi-supervised learning (SSL), the method of incorporating *unlabeled data* into the training, is one of the promising approaches for mitigating the difficulty of training DNNs. The goal of this thesis is to establish the SSL methodology with "more unlabeled data, better performance" property, i.e., simply scaling the amount of unlabeled data leads to better model performance.

In order to accomplish the goal, one needs to consider the scalability of the given SSL method. In other words, the method needs to (i) process large-scale unlabeled data in a reasonable amount of computational time and (ii) improve the model performance through more unlabeled data. Pretrained language model is a *generic* SSL method in which a single pretrained model can be applied to arbitrary downstream tasks, satisfies these scalability requirements. On the other hand, the scalability of the remaining *task-oriented* SSL methods is under-explored in the literature; this fact leaves a series of issues to address to achieve our final goal.

We first develop a scalable task-oriented SSL method for the text classification task, which is one of the simplest tasks of NLP. The proposed method is called a mixture of expert/imitator networks. The method demonstrates promising scalability concerning the amount of unlabeled data, which leads to the state-of-the-art performance on text classification benchmark datasets. Moreover, the experimental findings provide several implications for adapting the method for more complex NLP tasks.

Towards adapting the scalable SSL method to more complex tasks, we investigate the scalability of the existing SSL method(s) on two sequence-to-sequence tasks, namely, grammatical error correction and machine translation. As a result, we demonstrate that the SSL method with the highest scalability can achieve state-of-the-art performance. In addition, we reveal the limits of the "more unlabeled data" paradigm on both tasks.

Finally, we tackle the limits of SSL by making a tiny modification in a training procedure. Using the position generalization issue of Transformer models, we simulate the situation such that more data does not improve the performance. Our proposed method, shifted absolute position embeddings, makes a minimal modification to the position representation of the Transformer. The method significantly improves the models' generalization to positions.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

To date, deep neural networks (DNNs) have been achieving excellent performance on many tasks across multiple research fields, such as machine translation (MT) (Vaswani et al., 2017), image recognition (He et al., 2016), and automatic speech recognition (Amodei et al., 2016). Several studies even claim that their models surpass the human-level performance on benchmark datasets (Hassan et al., 2018; He et al., 2020; Kiela et al., 2021).

However, DNNs are extremely data-hungry, i.e., in order to take full advantage of DNN's performance, one needs to feed the model with large-scale data for training. Otherwise DNNs may severely over-fit the training data with little or no generalization. For example, Koehn and Knowles (2017) conducted an experiment on MT for comparing neural machine translation model (NMT) (Bahdanau et al., 2015; Luong et al., 2015; Sutskever et al., 2014) with the conventional phrase-based statistical machine translation model (PBSMT) (Koehn et al., 2007). They trained two models while varying the amount of parallel corpus, i.e., training data, from low-resource to high-resource setting. They demonstrated that NMT underperforms PBSMT by a large margin when the models are trained in low-resource settings. Sennrich and Zhang (2019) conducted a similar experiment. They conducted an extensive hyper-parameter search on NMT and found that NMT can outperform PBSMT. Nevertheless, the performance gap between the two models is smaller for the low-resource settings than for high-resource settings. Thus, the amount of data plays an increasingly important role in the current DNN paradigm.

A naive approach for boosting the performance of DNNs is to collect more labeled data for training. However, this approach is often infeasible because one cannot afford extra annotation. For example, Dua et al. (2019) reported that annotation of around 100k question-answer pairs cost US$60k. Such high cost originates in the annotation difficulty of many

NLP tasks; they often require a highly skilled annotator with a sophisticated linguistic background[1].

Semi-supervised learning (SSL) is one of the promising approaches for overcoming the lack of labeled data. SSL incorporates unlabeled data into the training in addition to labeled data. SSL takes advantage of the fact that, compared to labeled data, unlabeled data is relatively easy to collect. For example, in NLP, unlabeled data is essentially a raw text; thus, large-scale unlabeled data can be collected by crawling the Web. In fact, several terabyte-scale corpus of such crawled data are publicly available (Gao et al., 2021; Ortiz Suárez et al., 2019; Raffel et al., 2020). The goal of this thesis is to establish SSL methodology with the property of "more unlabeled data, better performance" and make full use of such gigantic unlabeled data.

SSL methodologies can be classified into two categories: (1) *generic* SSL and (2) *task-oriented* SSL. The former refers to the methodologies that utilize unlabeled data in a manner that is independent of the target task, e.g., word vectors (Mikolov et al., 2013; Pennington et al., 2014) and pretrained language models (PLMs) (Devlin et al., 2019; Peters et al., 2018). In other words, the generic SSL aims to acquire useful information for downstream tasks in general. On the other hand, the task-oriented methodologies incorporate the target task information into their algorithm, e.g., using labeled training data of the target task (Miyato et al., 2017; Suzuki and Isozaki, 2008; Suzuki et al., 2009). Here, task-oriented SSL attempts to acquire information specifically useful for the target downstream task. It has been empirically demonstrated that generic SSL and task-oriented SSL are complementary to each other (Kiyono et al., 2019; Miyato et al., 2017).

Towards the goal of utilizing terabyte-scale unlabeled data, we must consider the *scalability* of SSL. Here, scalability refers to the following two distinct concepts:

Computational Scalability considers if a given SSL method can process large-scale unlabeled data in a reasonable amount of computational time.

Performance Scalability considers if scaling the amount of unlabeled data indeed improves the model performance.

PLM, the current most successful generic SSL method in NLP, satisfies both scalability requirements. Specifically, PLMs use Transformer architecture (Vaswani et al., 2017), whose computation is highly parallelizable, to satisfy computational scalability. In addition, Kaplan et al. (2020) empirically demonstrated the performance scalability through the "scaling law," that is, there is a power-law relationship between the amount of unlabeled data and the test

[1]The PropBank (Bonial et al., 2010), which is a widely-used dataset for semantic role labeling, has the annotation guideline of 89 pages.

performance (i.e., perplexity) of a language model. As a result, PLM has been achieving excellent performance on a variety of downstream tasks. However, on the other hand, the scalability of task-oriented SSL methods are relatively under-explored, which motivates us to investigate the following research issues.

1.1 Research Issues

What makes task-oriented SSL scalable?: As we elaborate in Chapter 2, findings on the performance scalability of task-oriented SSL are mixed in the research field. Thus, the properties that make the underlying method scalable (or unscalable) are unclear.

Scalable task-oriented SSL method applicable for arbitrary tasks: The task-oriented SSL method often makes the huge assumption on the target task. Thus, it is unclear if a finding on a specific task can transfer to the other. One must confirm the effectiveness of scalable task-oriented SSL across multiple tasks.

What are the limits of task-oriented SSL methods?: Suppose that we have a scalable SSL method at hand; is incorporating more unlabeled data all we need for improving model performance? What are limitations or remaining challenges of scalable task-oriented SSL?

1.2 Contributions

This thesis makes the following contributions:

Building a scalable task-oriented SSL method: We propose a novel task-oriented SSL method. We demonstrate that our method scales to the amount of unlabeled data using the text classification benchmark data. Through the analysis, we investigate the requirements for a scalable task-oriented SSL method. Finally, we demonstrate that our method can be combined with the state-of-the-art generic SSL method to improve the performance.

Expanding the applicability of scalable task-oriented SSL method: We expand the applicability of scalable task-oriented SSL by tackling one of sequence-to-sequence problems, namely, grammatical error correction (GEC). We conduct a controlled empirical comparison of existing task-oriented SSL methods on GEC. We identify the best scalable method through the experiment and achieve the state-of-the-art performance on multiple GEC benchmark datasets.

Exploring the limits of task-oriented SSL methods: For MT and GEC, we analyze the models trained on a massive amount of unlabeled data. We reveal the limitations of the scaling the amount of unlabeled data through the analysis.

Minimal architecture modification as a means of compensating the lack of data with desired property: We enhance the state-of-the-art Transformer model to address the limitations of current SSL methods. Using the issue of length extrapolation, we demonstrate that an enhancement in Transformers' position representation can improve the model's generalization to the sequences that are longer than those observed during the training.

1.3 Thesis Overview

The rest of this thesis is structured as follows:

Chapter 2: Semi-supervised Learning for NLP. This chapter provides the overview of current SSL methodologies for neural networks. We first organize the methodologies into two categories and then discuss their scalability aspects to highlight the importance of this thesis' research issues.

Chapter 3: Large-scale Task-oriented Semi-supervised Learning for Text Classification. This chapter proposes a novel SSL method: a Mixture of Expert/Imitator Networks (MEIN). The experiment on multiple text classification benchmark dataset reveals that the proposed method not only outperforms the state-of-the-art methods but also shows superior computational scalability. MEIN also demonstrates performance scalability; that is, increasing the amount of unlabeled data leads to better model performance.

Chapter 4: Massive Exploration of Pseudo Data for Grammatical Error Correction. This chapter presents an extensive empirical comparison of current SSL methods. Then we incorporate large-scale unlabeled data (70M sentence pairs) into the training of the DNN model and achieve state-of-the-art performance on GEC benchmark datasets. Finally, we conduct a thorough analysis of the model through the viewpoint of English proficiency and grammatical error type. We then reveal both the limitations of the SSL method and the remaining challenges of the task.

Chapter 5: The Role of Semi-supervised Learning in the State-of-the-Art Machine Translation System. This chapter introduces the winning system that we submitted to the WMT 2020 news translation shared task. We demonstrate that merely using the current best SSL method (back-translation) is insufficient for building the winning system. Instead, combining multiple enhancements, e.g., model ensembles and reranking, is crucial.

Chapter 6: Shifted Absolute Position Embedding for Transformers. The length extrapolation is one of the crucial abilities for a robust natural language generation system. We exploit the concept of length extrapolation for simulating the situation in which SSL methods cannot generate data with the desirable property. We then propose a novel position representation for the Transformer to improve the length extrapolation ability.

Chapter 7: Conclusion. This chapter summarizes the contribution.

Chapter 2

Semi-supervised Learning for NLP

This chapter gives an overview of Semi-supervised Learning (SSL) for NLP. SSL is one of the machine learning paradigms in which a model is trained with a set of unlabeled data in addition to a set of labeled data. Here, the goal is to achieve a better generalization of the target task. Historically, SSL has been actively studied for various machine learning methods, including support vector machines (Bennett and Demiriz, 1999) and conditional random fields (Suzuki and Isozaki, 2008). Recently, almost the entire research field has shifted to neural network-based methods; thus, the neural SSL method has been actively developed (Yang et al., 2021). Neural SSL methods can be categorized into a task-oriented SSL and generic SSL.

2.1 Task-oriented SSL

Task-oriented SSL is an SSL method that exclusively aims to achieve better performance on the target task. To do this, the family of this SSL method incorporates the target task information into their algorithm, e.g., through the use of labeled training data of the target task.

Consistency regularization is one of the most successful task-oriented SSL approaches for NLP, whose idea is to train a model to generate probability distributions that are consistent across the given input and its nearby data points. In consistency regularization, the model with parameter set θ generates two probability distributions for a given input x; the one is for the input *without* the noise $p(y|x, \theta)$ and the other is the one *with* the noise $p(y|x + \epsilon, \theta)$. These predictions are regarded as the soft target and the model prediction, respectively.

Then we compute the loss $J(\theta)$ from two distributions and update the model parameters. Here, KL divergence is typically used as a loss function $J(\theta)$. Consistency regularization has been successfully applied to a variety of NLP tasks, including not only a naive text classification (Chen et al., 2020a; Miyato et al., 2017; Sato et al., 2018) but also sequential tagging (Chen et al., 2020b; Clark et al., 2018) and sequence generation (Sato et al., 2019; Takase and Kiyono, 2021).

The underlying challenge of consistency regularization is to formulate a "good" noise ϵ that can effectively improve the model's generalization on the target task; in fact, several sophisticated noise generation algorithms have been proposed to address such challenge. For example, adversarial noise (Goodfellow et al., 2015) is generated by computing the perturbation that most severely increases the loss value. Similarly, virtual adversarial noise (Miyato et al., 2017) can be computed from unlabeled data by computing the perturbation to which the model's prediction is the most sensitive. In addition, Cross-view training (CVT) (Clark et al., 2018) casts the concept of the noise to the restricted views of the given input; that is, the model is forced to make a prediction from a part of the input. For example, in the case of sequential tagging problems, such restricted views include the sequence without the future (i.e., right-hand-side) and the past (i.e., left-hand-side) contexts.

2.2 Generic SSL

In contrast to task-oriented SSL, generic SSL uses unlabeled data in a manner that is entirely independent of the target task. Generic SSL aims to acquire a set of features that are generally useful for arbitrary downstream tasks.

A most common approach is to use unlabeled data for pretraining on language modeling objectives, that is, the objective of predicting the missing word from given contexts (i.e., surrounding words). More specifically, two methodologies exist (1) word embeddings and (2) pretrained language models (PLMs). The latter approach, PLM, is dominant in the research field (Bommasani et al., 2021).

The concept of PLM is to pretrain a gigantic model on a language modeling task with a massive amount of unlabeled data. Then, for a given labeled training data for a target task, a pretrained model can be used as a feature extractor or an initial parameter for finetuning. The effectiveness of PLM is first demonstrated by Dai and Le (2015). They pretrained the RNN-based classifier on language modeling and reported that pretraining not only stabilizes the training on labeled data but also improves the generalization. Peters et al. (2017) adopted a similar approach on a much larger scale; they pretrained a multi-layer RNN on huge unlabeled data, namely, a one billion word benchmark (Chelba et al., 2014), and reported the improved performance on sequential labeling tasks. The use of such large-scale pretrained RNN is then expanded to various NLP tasks (Peters et al., 2018). In order to better utilize the bidirectional context, Devlin et al. (2019) replaced the internal architecture with Transformer model (Vaswani et al., 2017) and the language modeling objective with the *masked* language modeling objective. Both modifications further improved the downstream performance. Interestingly, the language modeling objective is empirically superior to the other objectives (Wang et al., 2019a). In addition, there exists an attempt to understand the reason for the effectiveness of the objective from the theoretical viewpoint (Saunshi et al., 2021).

2.3 Scalability of SSL

As discussed in Chapter 1, the final goal of this thesis is to make full use of ever-increasing unlabeled data in SSL. To achieve the goal, we must consider the scalability of SSL methods. More specifically, a scalable SSL must satisfy both of the following two concepts:

Computational Scalability: A given SSL method can process large-scale unlabeled data in a reasonable amount of computational time.

Performance Scalability: Scaling the amount of unlabeled data improves the model performance.

Among generic SSL (Section 2.2) methods, PLM satisfies both requirements. For example, the current state-of-the-art PLM methods satisfy the computational scalability by using the Transformer model (Vaswani et al., 2017). The internal operations of the Transformer model, i.e., (self-)attention mechanism (Bahdanau et al., 2015; Lin et al., 2017), are highly parallelizable; thus, the computation can be accelerated by modern hardware such as GPU and TPU (Jouppi et al., 2017). In addition, the performance scalability is also satisfied; Kaplan et al. (2020) empirically demonstrated the existence of scaling law, that is, there exists a power-law relationship between test set loss and one of (1) amount of compute, (2) dataset size, and (3) number of parameters. The effectiveness of scaling the PLM on the downstream performance has also been reported in various studies (Brown et al., 2020; Devlin et al., 2019; Radford et al., 2019).

On the other hand, in task-oriented SSL (Section 2.1), the scalability of each method is relatively under-explored, and no consensus has been established yet. Although several studies report the effectiveness of scaling the amount of unlabeled data, no generic findings are available across different tasks. For example, Edunov et al. (2018) have successfully scaled the amount of unlabeled data on machine translation; however, it is unclear whether their findings can generalize to tasks other than machine translation. In fact, Chen et al. (2021) applied several task-oriented SSL methods, including VAT and CVT, for sequential labeling tasks. They reported that scaling the amount of unlabeled data from 500K to 1M hurts the performance; this report suggests the difficulty of achieving the performance scalability in task-oriented SSL. A similar difficulty is also reported in image classification (Oliver et al., 2018). Given this background, the scalability of task-oriented SSL leaves us with research questions discussed in Chapter 1.

Chapter 3

Large-scale Task-oriented Semi-supervised Learning for Text Classification

3.1 Introduction

It is commonly acknowledged that deep neural networks (DNNs) can achieve excellent performance in many tasks across numerous research fields, such as image classification (He et al., 2016), speech recognition (Amodei et al., 2016), and machine translation (Wu et al., 2016). Recent progress in these tasks has been primarily driven by the following two factors: (1) A large amount of labeled training data exists. For example, ImageNet (Deng et al., 2009), one of the major datasets for image classification, consists of approximately 14 million labeled images. (2) DNNs have the property of achieving better performance when trained on a larger amount of labeled training data, namely, the *more data, better performance* property.

However, collecting a sufficient amount of labeled training data is not always easy for many actual applications. We refer to this issue as the *labeled data scarcity* issue. This issue is particularly crucial in the field of natural language processing (NLP), where only a few thousand or even a few hundred labeled data are available for most tasks. This is because, in typical NLP tasks, creating the labeled data often requires the professional supervision of several highly skilled annotators. As a result, the cost of data creation is high relative to the amount of data.

Unlike labeled data, unlabeled data for NLP tasks is essentially a collection of raw texts; thus, an enormous amount of unlabeled data can be obtained from the Internet, such as

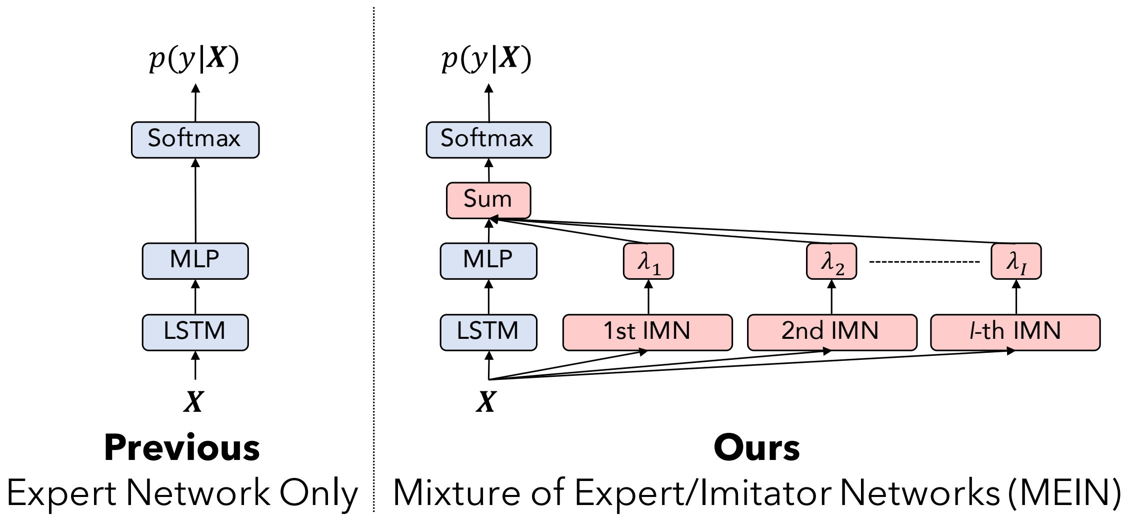


Figure 3.1 Overview of our framework: the Mixture of Expert/Imitator Networks (MEIN)

through the Common Crawl website[1], at a relatively low cost. With this background, semi-supervised learning (SSL), which leverages unlabeled data in addition to labeled training data for training the parameters of DNNs, is one of the promising approaches to practically addressing the labeled data scarcity issue in NLP. In fact, some intensive studies have recently been undertaken with the aim of developing SSL methods for DNNs and have shown promising results (Clark et al., 2018; Dai and Le, 2015; Mikolov et al., 2013; Miyato et al., 2017; Peters et al., 2018).

In this chapter, we also follow this line of research topic, i.e., discussing SSL suitable for NLP. Our interest lies in the *more data, better performance* property of the SSL approach over the unlabeled data, which has been implicitly demonstrated in several previous studies (Pennington et al., 2014; Peters et al., 2018). In order to take advantage of the huge amount of unlabeled data and improve performance, we need an SSL approach that scales with the amount of unlabeled data. However, the scalability of an SSL approach has not yet been widely discussed, since the primary focus of many of the recent studies on SSL in DNNs has been on improving the performance. For example, several studies have utilized unlabeled data as additional training data, which essentially increases the computational cost of (often complex) DNNs (Clark et al., 2018; Miyato et al., 2017; Sato et al., 2018). Another SSL approach is to (pre-)train a gigantic bidirectional language model (Peters et al., 2018). Nevertheless, it has been reported that the training of such a network requires 3 weeks using 32 GPUs (Jozefowicz et al., 2016). By developing a scalable SSL method, we hope to

[1]http://commoncrawl.org

broaden the usefulness and applicability of DNNs since, as mentioned above, the amount of unlabeled data can be easily increased.

In this chapter, we propose a novel scalable method of SSL, which we refer to as the Mixture of Expert/Imitator Networks (MEIN). Figure 3.1 gives an overview of the MEIN framework, which consists of an expert network (EXN) and at least one imitator network (IMN). To ensure scalability, we design each IMN to be computationally simpler than the EXN. Moreover, we use unlabeled data exclusively for training each IMN; we train the IMN so that it *imitates* the label estimation of the EXN over the unlabeled data. The basic idea underlying the IMN is that we force it to perform the imitation with only a limited view of the given input. In this way, the IMN effectively learns a set of features, which potentially contributes to the EXN. Intuitively, our method can be interpreted as a variant of several training techniques of DNNs, such as the mixture-of-experts (Jacobs et al., 1991; Shazeer et al., 2017), knowledge distillation (Ba and Caruana, 2014; Hinton et al., 2015), and ensemble techniques.

We conduct experiments on well-studied text classification datasets to evaluate the effectiveness of the proposed method. We demonstrate that the MEIN framework consistently improves the performance for three distinct settings of the EXN. We also demonstrate that our method has the *more data, better performance* property with promising scalability to the amount of unlabeled data. In addition, a current popular SSL approach in NLP is to pre-train the language model and then apply it to downstream tasks (Dai and Le, 2015; McCann et al., 2017; Mikolov et al., 2013; Peters et al., 2017, 2018). We empirically prove in our experiments that MEIN can be easily combined with this approach to further improve the performance of DNNs.

3.2 Related Work

There have been several previous studies in which SSL has been applied to text classification tasks. A common approach is to utilize unlabeled data as additional training data of the DNN. Studies employing this approach mainly focused on developing a means of effectively acquiring a teaching signal from the unlabeled data. For example, in virtual adversarial training (VAT) (Miyato et al., 2017) the perturbation is computed from unlabeled data to make the baseline DNN more robust against noise. Sato et al. (2018) proposed an extension of VAT that generates a more interpretable perturbation. In addition, cross-view training (CVT) (Clark et al., 2018) considers the auxiliary loss by making a prediction from an unlabeled input with a restricted view. On the other hand, in our MEIN framework, we do not use unlabeled data as additional training data for the baseline DNN. Instead, we use the unlabeled data to train the IMNs to imitate the baseline DNN. The advantage of such usage is that one can choose an arbitrary architecture for the IMNs. Specifically, we design the IMN to be computationally simpler than the baseline DNN to ensure better scalability with the amount of unlabeled data (Table 3.4).

The idea of our *expert-imitator* approach originated from the SSL framework proposed by Suzuki and Isozaki (2008). They incorporated several simple generative models as a set of additional features for a supervised linear conditional random field classifier. Our EXN and IMN can be regarded as their linear classifier and the generative models, respectively. In addition, they empirically demonstrated that the performance has a linear relationship with the logarithm of the unlabeled data size. We empirically demonstrate that the proposed method also exhibits similar behavior (Figure 3.3), namely, increasing the amount of unlabeled data reduces the error rate of the EXN.

One of the major SSL approaches in NLP is to pre-train a language model over unlabeled data. The pre-trained weights have many uses, such as parameter initialization (Dai and Le, 2015) and as a source of additional features (McCann et al., 2017; Peters et al., 2017, 2018), in downstream tasks. For example, Peters et al. (2018) have recently trained a bi-directional LSTM language model using the One Billion Word Benchmark dataset (Chelba et al., 2014). They utilized the hidden state of the LSTM as contextualized embedding, called *ELMo* embedding, and achieved state-of-the-art results in many downstream tasks. In our experiment, we empirically demonstrate that the proposed MEIN is complementary to the pre-trained language model approach. Specifically, we show that by combining the two approaches, we can further improve the performance of the baseline DNN.

3.3 Task Description and Notation Rules

This section gives a formal definition of the text classification task discussed in this chapter. Let $\mathscr{V}$ represent the vocabulary of the input sentences. $\boldsymbol{x}_t \in \{0,1\}^{|\mathscr{V}|}$ denotes the one-hot vector of the t-th token (word) in the input sentence, where $|\mathscr{V}|$ represents the number of tokens in $\mathscr{V}$. Here, we introduce the short notation form $(\boldsymbol{x}_t)_{t=1}^{T}$ to represent a sequence of vectors for simplicity, that is, $(\boldsymbol{x}_t)_{t=1}^{T} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$. Suppose we have an input sentence that consists of T tokens. For a succinct notation, we introduce $\boldsymbol{X}$ to represent a sequence of one-hot vectors that corresponds to the tokens in the input sentence, namely, $\boldsymbol{X} = (\boldsymbol{x}_t)_{t=1}^{T}$. $\mathscr{Y}$ denotes a set of output classes. Let $y \in \{1, \ldots, |\mathscr{Y}|\}$ be an integer that represents the output class ID. In addition, we define $\boldsymbol{X}_{a:b}$ as the subsequence of $\boldsymbol{X}$ from index a to index b, namely, $\boldsymbol{X}_{a:b} = (\boldsymbol{x}_a, \boldsymbol{x}_{a+1} \ldots, \boldsymbol{x}_b)$ and $1 \leq a \leq b \leq T$. We also define $\mathbf{x}[i]$ as the i-th element of vector $\mathbf{x}$. For example, if $\mathbf{x} = (5, 2, 1, -1)^{\top}$, then $\mathbf{x}[2] = 2$ and $\mathbf{x}[4] = -1$.

In the supervised training framework for text classification tasks modeled by DNNs, we aim to maximize the (conditional) probability $p(y|\boldsymbol{X})$ over a given set of labeled training data $(\boldsymbol{X}, y) \in \mathscr{D}_s$ by using DNNs. In the semi-supervised training, the objective of maximizing the probability is identical but we also use a set of unlabeled training data $\boldsymbol{X} \in \mathscr{D}_u$.

3.4 Baseline Network: LSTM with MLP

In this section, we briefly describe a baseline DNN for text classification. Among the many choices, we select the *LSTM-based text classification model* described by Miyato et al. (2017) as our baseline DNN architecture since they achieved the current best results on several well-studied text classification benchmark datasets. The network consists of the LSTM (Hochreiter and Schmidhuber, 1997) cell and a multi layer perceptron (MLP).

First, the LSTM cell calculates a hidden state sequence $(\boldsymbol{h}_t)_{t=1}^{T}$, where $\boldsymbol{h}_t \in \mathbb{R}^H$ for all t and H is the size of the hidden state, as $\boldsymbol{h}_t = \mathrm{LSTM}(\boldsymbol{E}\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$. Here, $\boldsymbol{E} \in \mathbb{R}^{D\times|\mathscr{V}|}$ is the word embedding matrix, D denotes the size of the word embedding, and $\boldsymbol{h}_0$ is a zero vector.

Then the T-th hidden state $\boldsymbol{h}_T$ is passed through the MLP, which consists of a single fully connected layer with ReLU nonlinearity (Glorot et al., 2011), to compute the final hidden state $\boldsymbol{s} \in \mathbb{R}^M$. Specifically, $\boldsymbol{s}$ is computed as $\boldsymbol{s} = \mathrm{ReLU}(\boldsymbol{W}_h\boldsymbol{h}_T + \boldsymbol{b}_h)$, where $\boldsymbol{W}_h \in \mathbb{R}^{M\times H}$ is a trainable parameter matrix and $\boldsymbol{b}_h \in \mathbb{R}^M$ is a bias term. Here, M denotes the size of the final hidden state of the MLP.

Finally, the baseline DNN estimates the conditional probability from the final hidden state $\boldsymbol{s}$ as follows:

$$z_y = \boldsymbol{w}_y^\top \boldsymbol{s} + b_y, \tag{3.1}$$

$$p(y|\boldsymbol{X}, \boldsymbol{\Theta}) = \frac{\exp(z_y)}{\sum_{y'\in\mathscr{Y}} \exp(z_{y'})}, \tag{3.2}$$

where $\boldsymbol{w}_y \in \mathbb{R}^M$ is the weight vector of class y and b_y is the scalar bias term of class y. Also, $\boldsymbol{\Theta}$ denotes all the trainable parameters of the baseline DNN.

For the training process of the parameters in the baseline DNN $\boldsymbol{\Theta}$, we seek the (sub-)optimal parameters that minimize the (empirical) negative log-likelihood for the given labeled training data $\mathscr{D}_s$, which can be written as the following optimization problem:

$$\mathbf{\Theta}' = \arg\min_{\mathbf{\Theta}} \left\{ L_s(\mathbf{\Theta}|\mathscr{D}_s) \right\}, \tag{3.3}$$

$$L_s(\mathbf{\Theta}|\mathscr{D}_s) = -\frac{1}{|\mathscr{D}_s|} \sum_{(\mathbf{X},y)\in\mathscr{D}_s} \log\left(p(y|\mathbf{X},\mathbf{\Theta})\right), \tag{3.4}$$

where $\mathbf{\Theta}'$ represents the set of obtained parameters in the baseline DNN, by solving the above minimization problem. Practically, we apply a variant of a stochastic gradient descent algorithm such as Adam (Kingma and Ba, 2015).

3.5 Proposed Model: Mixture of Expert/Imitator Networks (MEIN)

Figure 3.1 gives an overview of the proposed method, which we refer to as MEIN. MEIN consists of an expert network (EXN) and a set of imitator networks (IMNs). Once trained, the EXN and the set of IMNs jointly predict the label of a given input $\boldsymbol{X}$. Figure 3.1 shows the baseline DNN (LSTM with MLP) as an example of the EXN. Note that MEIN can adopt an arbitrary classification network as the EXN.

3.5.1 Basic Idea

A brief description of MEIN is as follows: (1) The EXN is trained using labeled training data. Thus, the EXN is expected to be very accurate over inputs that are similar to the labeled training data. (2) IMNs (we basically assume that we have more than one IMN) are trained to imitate the EXN. To accomplish this, we train each IMN to minimize the Kullback–Leibler (KL) divergence between estimations of label distributions of the EXN and the IMNs over the unlabeled data. (3) Our final classification network is a mixture of the EXN and IMN(s). Here, we fine-tune the EXN using the labeled training data jointly with the estimations of all the IMNs.

The basic idea underlying MEIN is that we force each IMN to imitate estimated label distributions with only a limited view of the given input. Specifically, we adopt a sliding window to divide the input into several fragments of n-grams. Given a large amount of unlabeled data and the estimation by the EXN, the IMN learns to represent the label "tendency" of each fragment in the form of a label distribution (i.e., certain n-grams are more likely to have positive/negative labels than others). Our assumption here is that this tendency can potentially contribute a set of features for the classification. Thus, after training the IMNs,

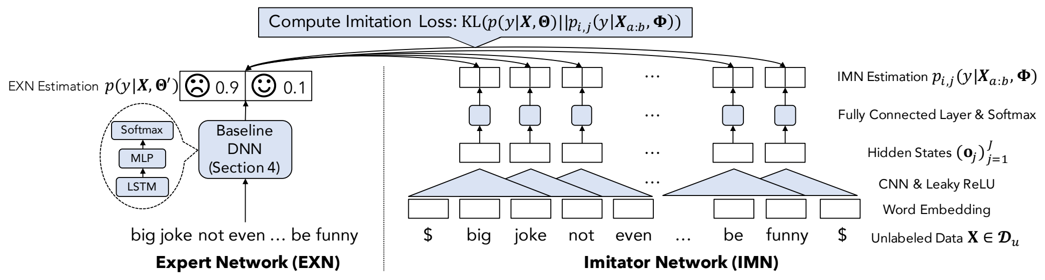

Compute Imitation Loss: $\mathrm{KL}(p(y|\boldsymbol{X},\boldsymbol{\Theta})||p_{i,j}(y|\boldsymbol{X}_{a:b},\boldsymbol{\Phi}))$
EXN Estimation $p(y|\boldsymbol{X},\boldsymbol{\Theta}')$
☹ 0.9
☺ 0.1
Softmax
MLP
LSTM
Baseline DNN (Section 4)
big joke not even … be funny
Expert Network (EXN)
IMN Estimation $p_{i,j}(y|\boldsymbol{X}_{a:b},\boldsymbol{\Phi})$
Fully Connected Layer & Softmax
Hidden States $(\mathbf{o}_j)_{j=1}^{J}$
CNN & Leaky ReLU
Word Embedding
$ big joke not even … be funny $
Unlabeled Data $\mathbf{X} \in \mathcal{D}_u$
Imitator Network (IMN)

Figure 3.2 Overview of the 1st IMN ($c_1 = 1$). The IMN must predict the label estimation of the EXN from a limited amount of information. $ denotes a special token used to pad the input (a zero vector).

we jointly optimize the EXN and the weight of each feature. Here, MEIN may control the contribution of each feature by updating the corresponding weight.

Intuitively, our MEIN approach can be interpreted as a variant of several successful machine learning techniques for DNNs. For example, MEIN shares the core concept with the mixture-of-experts technique (MoE) (Jacobs et al., 1991; Shazeer et al., 2017). The difference is that MoE considers a mixture of several EXNs, whereas MEIN generates a mixture from a single EXN and a set of IMNs. In addition, one can interpret MEIN as a variant of the ensemble, bagging, voting, or boosting technique since the EXN and the IMNs jointly make a prediction. Moreover, we train each IMN by minimizing the KL-divergence between the EXN and the IMN through unlabeled data. This process can be seen as a form of "knowledge distillation" (Ba and Caruana, 2014; Hinton et al., 2015). We utilize these methodologies and formulate the framework as described below.

3.5.2 Network Architecture

Let $\sigma(\cdot)$ be the sigmoid function defined as $\sigma(\lambda) = (1 + \exp(-\lambda))^{-1}$. $\boldsymbol{\Phi}$ denotes a set of trainable parameters of the IMNs and I denotes the number of IMNs. Then, the EXN combined with a set of IMNs models the following (conditional) probability:

$$p(y|\mathbf{X}, \boldsymbol{\Theta}, \boldsymbol{\Phi}, \boldsymbol{\Lambda}) = \frac{\exp(z'_y)}{\sum_{y' \in \mathcal{Y}} \exp(z'_{y'})}, \tag{3.5}$$

$$\text{where} \quad z'_y = z_y + \sum_{i=1}^{I} \sigma(\lambda_i)\boldsymbol{\alpha}_i[y]. \tag{3.6}$$

λ_i is a scalar parameter that controls the contribution of logit $\boldsymbol{\alpha}_i$ of the i-th IMN and $\boldsymbol{\Lambda}$ is defined as $\boldsymbol{\Lambda} = \{\lambda_1, \ldots, \lambda_I\}$. Here, logit $\boldsymbol{\alpha}_i$ represents an estimated label distribution, which we assume to be a feature. Note that the first term of Equation 3.6 is the baseline DNN logit $z_y = \mathbf{w}_y^{\top}\boldsymbol{s} + b_y$ (Equation 3.1). In addition, if we set $\sigma(\lambda_i) = 0$ for all i, then Equation 3.5 becomes identical to Equation 3.2 regardless of the value of $\boldsymbol{\Phi}$.

Given an input $\boldsymbol{X}$ and the i-th IMN, we create J inputs with a sliding window of size c_i, where c_i denotes the window size of the i-th IMN. Then the IMN predicts the EXN for each input and generates J predictions as a result. We compute the i-th imitator logit $\boldsymbol{\alpha}_i$ by taking the average of these predictions. Specifically, $\boldsymbol{\alpha}_i$ is defined as

$$\boldsymbol{\alpha}_i = \log\left(\frac{1}{J}\sum_{j=1}^{J} p_{i,j}(y|\boldsymbol{X}_{a:b}, \boldsymbol{\Phi})\right), \qquad \text{where} \quad a = j - c_i \quad \text{and} \quad b = j + c_i. \tag{3.7}$$

Here, a is a scalar index that represents the beginning of the window. Similarly, b represents the last index of the window.

3.5.3 Definition of IMNs

Note that the architecture of the IMN used to model Equation 3.7 is essentially arbitrary. In this research, we adopt a single-layer CNN for modeling $p_{i,j}(y|\boldsymbol{X}_{a:b}, \boldsymbol{\Phi})$. This is because a CNN has high computational efficiency (Gehring et al., 2017), which is essential for our primary focus: scalability with the amount of unlabeled data.

Figure 3.2 gives an overview of the architecture of the IMN. First, the IMN takes a sequence of word embeddings of input $\boldsymbol{X}$ and computes a sequence of hidden states $(\boldsymbol{o}_j)_{j=1}^{J}$ by applying a *one-dimensional convolution* (Kalchbrenner et al., 2014) and leaky ReLU nonlinearity (Maas et al., 2013). We ensure that J is always equal to T. To achieve this, we pad the beginning and the end of the input $\boldsymbol{X}$ with zero vectors $\mathbf{0} \in \mathbb{R}^{|\mathcal{V}'| \times c_i}$, where $|\mathcal{V}'|$ denotes the vocabulary size of the IMN.

As explained in Section 3.5.2, each IMN has a predetermined and fixed window size c_i. One can choose an arbitrary window size for the i-th IMN. Here, we define c_i as $c_i = i$ for simplicity. For example, as shown in Figure 3.2, the 1st IMN ($i = 1$) has a window size of $c_1 = 1$. Such a network imitates the estimation of the EXN from three consecutive tokens.

Then the i-th IMN estimates the probability $p_{i,j}(y|\mathbf{X}, \boldsymbol{\Phi})$ from each hidden state $\boldsymbol{o}_j$ as

$$p_{i,j}(y|\boldsymbol{X}_{a:b}, \boldsymbol{\Phi}) = \frac{\exp(\mathbf{w}'^{\top}_{i,y}\boldsymbol{o}_j + b'_{i,y})}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}'^{\top}_{i,y'}\boldsymbol{o}_j + b'_{i,y'})}, \tag{3.8}$$

Algorithm 1: Training framework of MEIN

```
Data: Labeled data $\mathscr{D}_s$ and unlabeled data $\mathscr{D}_u$
Result: Trained set of parameters $\widehat{\mathbf{\Theta}}, \widehat{\mathbf{\Phi}}, \widehat{\mathbf{\Lambda}}$
1 $\mathbf{\Theta}' \leftarrow \arg\min_{\mathbf{\Theta}}\{L_s(\mathbf{\Theta}|\mathscr{D}_s)\}$    ▷ Train EXN (Equation 3.3)
2 $\widehat{\mathbf{\Phi}} \leftarrow \arg\min_{\mathbf{\Phi}}\{L_u(\mathbf{\Phi}|\mathbf{\Theta}', \mathscr{D}_u)\}$    ▷ Train IMN(s) (Equation 3.11)
3 $\widehat{\mathbf{\Theta}}, \widehat{\mathbf{\Lambda}} \leftarrow \arg\min_{\mathbf{\Theta},\mathbf{\Lambda}}\{L'_s(\mathbf{\Theta}, \mathbf{\Lambda}|\widehat{\mathbf{\Phi}}, \mathscr{D}_s)\}$    ▷ Train EXN (Equation 3.13)
```

where $\boldsymbol{w}'_{i,y} \in \mathbb{R}^N$ is the weight vector of the i-th IMN and $b'_{i,y}$ is the scalar bias term of class y. N denotes the CNN kernel size.

3.5.4 Training Framework

First, we define the *imitation loss* of each IMN as the KL-divergence between the estimations of the label distributions of the EXN and the IMN given (unlabeled) data $\boldsymbol{X}$, namely, $\mathrm{KL}(p(y|\boldsymbol{X}, \mathbf{\Theta})||p_{i,j}(y|\boldsymbol{X}_{a:b}, \mathbf{\Phi}))$. Note that this imitation loss is defined for an input with the sliding window $\boldsymbol{X}_{a:b}$. Thus, this definition effectively accomplishes the concept, i.e., the IMN making a prediction $p_{i,j}(y|\boldsymbol{X}_{a:b}, \mathbf{\Phi})$ from only a limited view of the given input $\boldsymbol{X}_{a:b}$.

Next, our objective is to estimate the set of optimal parameters by minimizing the negative log-likelihood of Equation 3.5 while also minimizing the total imitation losses for all IMNs as biases of the network. Therefore, we jointly solve the following two minimization problems for the parameter estimation of MEIN:

$$\widehat{\mathbf{\Theta}}, \widehat{\mathbf{\Lambda}} = \arg\min_{\mathbf{\Theta},\mathbf{\Lambda}}\{L'_s(\mathbf{\Theta}, \mathbf{\Lambda}|\widehat{\mathbf{\Phi}}, \mathscr{D}_s)\} \tag{3.9}$$

$$\widehat{\mathbf{\Phi}} = \arg\min_{\mathbf{\Phi}}\{L_u(\mathbf{\Phi}|\mathbf{\Theta}', \mathscr{D}_u)\}. \tag{3.10}$$

As described in Equations 3.9 and 3.10, we update the different sets of parameters depending on the labeled/unlabeled training data. Specifically, we use the labeled training data $(\boldsymbol{X}, y) \in \mathscr{D}_s$ to update the set of parameters in the EXN, $\mathbf{\Theta}$, and the set of mixture parameters of the IMNs, $\mathbf{\Lambda}$. In addition, we use the unlabeled training data $\boldsymbol{X} \in \mathscr{D}_u$ to update the parameters of the IMNs, $\mathbf{\Phi}$.

To ensure an efficient training procedure, the training framework of MEIN consists of three consecutive steps (Algorithm 1). First, we perform standard supervised learning to obtain $\mathbf{\Theta}'$ using labeled training data while keeping $\lambda_i = -\infty$ unchanged for all i during the training process to ensure that $\sigma(\lambda_i) = 0$ in Equation 3.6. Note that this optimization step is essentially equivalent to that of the baseline DNN (Equation 3.4).

Second, we estimate the set of IMN parameters $\mathbf{\Phi}$ by solving the minimization problem in Equation 3.10 with the following loss function:

$$L_u(\mathbf{\Phi}|\mathbf{\Theta}', \mathscr{D}_u) = \frac{1}{|\mathscr{D}_u|} \sum_{\mathbf{X}\in\mathscr{D}_u} \sum_{i=1}^{I} \sum_{j=1}^{J} \mathrm{KL}(p||p_{i,j}), \tag{3.11}$$

$$\mathrm{KL}(p||p_{i,j}) = -\sum_{y\in Y} p(y|\mathbf{X}, \mathbf{\Theta}') \log \left(p_{i,j}(y|\mathbf{X}_{a:b}, \mathbf{\Phi})\right) + const, \tag{3.12}$$

where $\mathrm{KL}(p||p_{i,j})$ is a shorthand notation of the imitation loss $\mathrm{KL}(p(y|\mathbf{X}, \mathbf{\Theta})||p_{i,j}(y|\mathbf{X}_{a:b}, \mathbf{\Phi}))$ and $const$ is a constant term that is independent of $\mathbf{\Phi}$.

Finally, we estimate $\mathbf{\Theta}$ and $\mathbf{\Lambda}$ by solving the minimization problem in Equation 3.9 with the following loss function:

$$L'_s(\mathbf{\Theta}, \mathbf{\Lambda}|\widehat{\mathbf{\Phi}}, \mathscr{D}_s) = -\frac{1}{|\mathscr{D}_s|} \sum_{(\mathbf{X},y)\in\mathscr{D}_s} \log \left(p(y|\mathbf{X}, \mathbf{\Theta}, \widehat{\mathbf{\Phi}}, \mathbf{\Lambda})\right). \tag{3.13}$$

3.6 Experiments

To investigate the effectiveness of MEIN, we conducted experiments on two text classification tasks: (1) a sentiment classification (SEC) task and (2) a category classification (CAC) task.

3.6.1 Datasets

For SEC, we selected the following widely used benchmark datasets: IMDB (Maas et al., 2011), Elec (Johnson and Zhang, 2015), and Rotten Tomatoes (Rotten) (Pang and Lee, 2005). For the Rotten dataset, we used the Amazon Reviews dataset (McAuley and Leskovec, 2013) as unlabeled data, following previous studies (Dai and Le, 2015; Miyato et al., 2017; Sato et al., 2018). For CAC, we used the RCV1 dataset (Lewis et al., 2004). Table 3.1 summarizes the characteristics of each dataset[2].

[2]DBpedia (Lehmann et al., 2015) is another widely adopted CAC dataset. We did not use this dataset in our experiment because it does not contain unlabeled data.

Table 3.1 Summary of datasets. Each value represents the number of instances contained in each dataset.

Task	Dataset	Classes	Train	Dev	Test	Unlabeled
SEC	Elec	2	22,500	2,500	25,000	200,000
	IMDB	2	21,246	3,754	25,000	50,000
	Rotten	2	8,636	960	1,066	7,911,684
CAC	RCV1	55	14,007	1,557	49,838	668,640

3.6.2 Baseline DNNs

In order to investigate the effectiveness of the MEIN framework, we combined the IMN with following three distinct EXNs and evaluated their performance:

- LSTM: This is the baseline DNN (LSTM with MLP) described in Section 3.4.
- LM-LSTM: Following Dai and Le (2015), we initialized the embedding layer and the LSTM with a pre-trained RNN-based language model (LM) (Bengio et al., 2003). We trained the language model using the labeled training data and unlabeled data of each dataset. Several previous studies have adopted this network as a baseline (Miyato et al., 2017; Sato et al., 2018).
- ADV-LM-LSTM: Adversarial training (ADV) (Goodfellow et al., 2015) adds small perturbations to the input and makes the network robust against noise. Miyato et al. (2017) applied ADV to LM-LSTM for a text classification. We used the reimplementation of their network.

Note that these three EXNs have an identical network architecture, as described in Section 3.4. The only difference is in the initialization or optimization strategy of the network parameters.

To the best of our knowledge, ADV-LM-LSTM provides a performance competitive with the current best result for the configuration of supervised learning (using labeled training data only). Thus, if the IMN can improve the performance of a strong baseline, the results will strongly indicate the effectiveness of our method.

3.6.3 Network Configurations

Table 3.2 summarizes the hyperparameters and network configurations of our experiments. We carefully selected the settings commonly used in the previous studies (Dai and Le, 2015; Miyato et al., 2017; Sato et al., 2018).

Table 3.2 Summary of hyperparameters

	Hyperparameter	Value
EXN (baseline DNN)	Word Embedding Dim. (D)	256
	Embedding Dropout Rate	0.5
	LSTM Hidden State Dim. (H)	1024
	MLP Dim. (M) for SEC Task	30
	MLP Dim. (M) for CAC Task	128
	Activation Function	ReLU
IMN	CNN Kernel Dim. (N)	512
	Word Embedding Dim.	512
	Activation Function	Leaky ReLU
	Number of IMNs (I)	4
Optimization	Algorithm	Adam
	Mini-Batch Size	32
	Initial Learning Rate	0.001
	Fine-tune Learning Rate	0.0001
	Decay Rate	0.9998
	Baseline Max Epoch	30
	Fine-tune Max Epoch	30

We used a different set of vocabulary for the EXN and the IMNs. We created the EXN vocabulary $\mathscr{V}$ by following the previous studies (Dai and Le, 2015; Miyato et al., 2017; Sato et al., 2018), i.e., we removed the tokens that appear only once in the whole dataset. We created the IMN vocabulary $\mathscr{V}'$ by byte pair encoding (BPE) (Sennrich et al., 2016c)[3]. The BPE merge operations are jointly learned from the labeled training data and unlabeled data of each dataset. We set the number of BPE merge operations to 20,000.

3.6.4 Results

Table 3.3 summarizes the results on all benchmark datasets, where the evaluation metric is the error rate. Therefore, a lower value indicates better performance. Here, all the reported results are **the average of five distinct trials** using five different random seeds. Moreover, for each trial, we automatically selected the best network in terms of the performance on the validation set among the networks obtained at every epoch. For comparison, we also performed experiments on training baseline DNNs (LSTM, LM-LSTM, and ADV-LM-LSTM) with incorporating random vectors as the replacement of IMNs, which is denoted as "+IMN (Ran-

[3]We used sentencepiece (Kudo and Richardson, 2018) (https://github.com/google/sentencepiece) for the BPE operations.

Table 3.3 Test performance (error rate (%)) on each dataset. **A lower error rate indicates better performance.** Models using the unlabeled data are marked with †. Results marked with * are statistically significant compared with ADV-LM-LSTM. `Miyato 2017`: the result reported by Miyato et al. (2017). `Sato 2018`: the result reported by Sato et al. (2018).

Method	Elec	IMDB	Rotten	RCV1
LSTM	10.09	10.98	26.47	14.14
LSTM+IMN (Random)†	9.87	10.75	27.27	14.04
LSTM+IMN†	**8.83**	**10.04**	**24.93**	**12.31**
LM-LSTM†	5.72	7.25	16.80	8.37
LM-LSTM+IMN (Random)†	5.71	7.01	16.78	7.83
LM-LSTM+IMN†	**5.48**	**6.51**	**15.91**	**7.53**
ADV-LM-LSTM†	5.38	6.58	15.73	7.89
ADV-LM-LSTM+IMN (Random)†	5.34	6.27	15.11	7.78
ADV-LM-LSTM+IMN†	**5.14***	**6.07***	**13.98**	**7.51***
VAT-LM-LSTM (rerun) †	5.47	6.20	18.50	8.44
VAT-LM-LSTM (`Miyato 2017`)†	5.54	5.91	19.1	7.05
VAT-LM-LSTM (`Sato 2018`)†	5.66	5.69	14.26	11.80
iVAT-LSTM (`Sato 2018`)†	5.18	5.66	14.12	11.68

dom)”. Moreover, we present the published results of VAT-LM-LSTM (Miyato et al., 2017) and iVAT-LSTM (Sato et al., 2018) in the bottom three rows of Table 3.3, which are the current state-of-the-art networks that adopt unlabeled data. For VAT-LM-LSTM, we also report the result of the reimplemented network, denoted as “VAT-LM-LSTM (rerun)”.

As shown in Table 3.3, incorporating the IMNs consistently improved the performance of all baseline DNNs across all benchmark datasets. Note that the source of these improvements is not the extra set of parameters $\mathbf{\Lambda}$ but the outputs of the IMNs. We can confirm this fact by comparing the results of IMNs, “+IMN”, with those of random vectors, “+IMN (Random)”, since the difference between these two settings is the incorporation of IMNs or random vectors.

The most noteworthy observation about MEIN is that the amount of the improvement upon incorporating the IMN is nearly consistent, regardless of the performance of the base EXN. For example, Table 3.3 shows that the IMN reduced the error rates of LSTM, LM-LSTM, and ADV-LM-LSTM by 1.54%, 0.89%, and 1.22%, respectively, for the Rotten dataset. From these observations, the IMN has the potential to further improve the performance of much stronger EXNs developed in the future.

We also remark that our best configuration, ADV-LM-LSTM+IMN, outperformed VAT-LM-LSTM (rerun) on all datasets[4]. In addition, the best configuration outperformed the current best published results on the Elec and Rotten datasets, establishing new state-of-the-art results.

As a comparison with the current strongest SSL method, we combined the IMN with the current state-of-the-art VAT method, namely, VAT-LM-LSTM+IMN. In the Elec dataset, the IMN improved the error rate from 5.47% to 5.16%. This result indicates that the IMN and VAT have a complementary relationship. Note that utilizing VAT is challenging in terms of the scalability with the amount of unlabeled data. However, if sufficient computing resources exist, then VAT and the IMN can be used together to achieve even higher performance.

3.7 Analysis

3.7.1 More Data, Better Performance Property

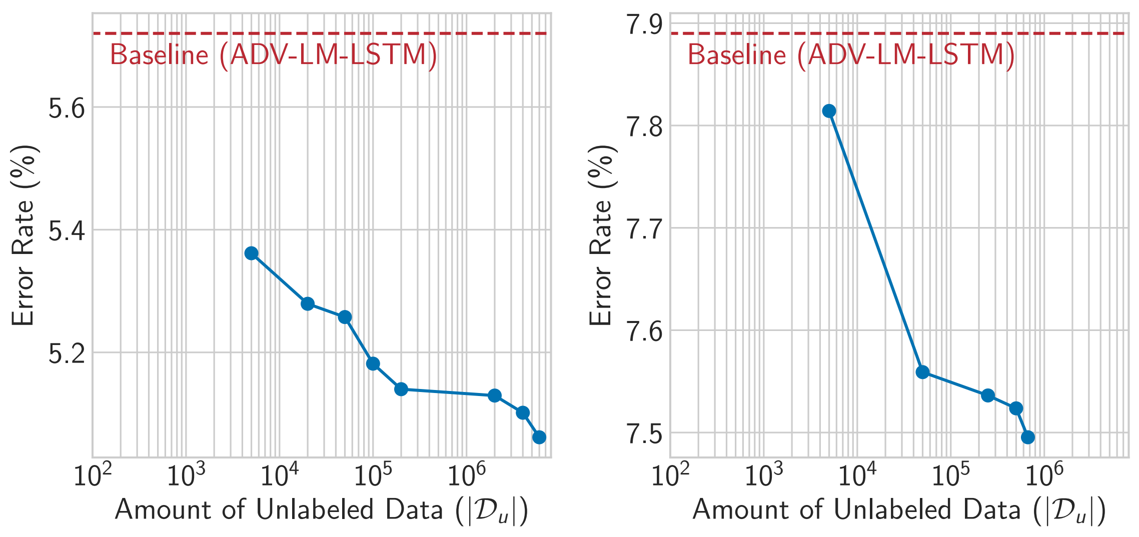
We investigated whether the MEIN framework has the *more data, better performance* property for unlabeled data. Ideally, MEIN should achieve better performance by increasing the amount of unlabeled data. Thus, we evaluated the performance while changing the amount of unlabeled data used to train the IMN.

We selected the Elec and RCV1 datasets as the focus of this analysis. We created the following subsamples of the unlabeled data for each dataset: {5K, 20K, 50K, 100K, Full Data} for Elec and {5K, 50K, 250K, 500K, Full Data} for RCV1. In addition, for the Elec dataset, we sampled extra unlabeled data from the electronics section of the Amazon Reviews dataset (McAuley and Leskovec, 2013) and constructed {2M, 4M, 6M} unlabeled data[5]. For each (sub)sample, we trained ADV-LM-LSTM+IMN as explained in Section 3.6.

Figures 3.3a and 3.3b demonstrate that increasing the amount of unlabeled data improved the performance of the EXN. It is noteworthy that in Figure 3.3a, ADV-LM-LSTM+IMN trained with 6M data achieved an error rate of 5.06%, outperforming the best result in Table 3.3 (5.14%). These results explicitly demonstrate the *more data, better performance*

[4]The performance of our VAT-LM-LSTM (rerun) is lower than the performances reported by Miyato et al. (2017) except for the Elec and Rotten datasets. Through extensive trials to reproduce their results, we found that the hyperparameter of the RNN language model is extremely important in determining the final performance; therefore, the strict reproduction of the published results is significantly difficult. In fact, a similar difficulty can be observed in Table 3.3, where VAT-LM-LSTM (`Sato 2018`) has lower performance than VAT-LM-LSTM (`Miyato 2017`) on the Elec and RCV1 datasets. Thus, we believe that VAT-LM-LSTM (rerun) is the most reliable result for the comparison.

[5]We discarded instances from the unlabeled data when the non stop-words overlap with instances in the Elec test set. Thus, the unlabeled data and the Elec test set had no instances in common.



(a) Elec (b) RCV1

Figure 3.3 Error rate (%) at different amounts of unlabeled data. The x-axis is in log-scale. **A lower error rate indicates better performance.** The dashed horizontal line represents the performance of the base EXN (ADV-LM-LSTM).

property of the MEIN framework. We also report that the training process on the largest amount of unlabeled data (6M) only took approximately a day.

3.7.2 Scalability with Amount of Unlabeled Data

The primary focus of the MEIN framework is its scalability with the amount of unlabeled data. Thus, in this section, we compare the computational speed of the IMNs with that of the base EXN. We also compare the IMNs with the state-of-the-art SSL method, VAT-LM-LSTM, and discuss their scalability. Here, we focus on the computation in the training phase of the network, where the network processes both forward and backward computations.

We measured the number of tokens that each network processes per second. We used identical hardware for each measurement, namely, a single NVIDIA Tesla V100 GPU. We used the cuDNN implementation for the LSTM cell since it is highly optimized and substantially faster than the naive implementation (Bradbury et al., 2017).

Table 3.4 summarizes the results. The table shows that even the slowest IMN ($c_i = 1, 2, 3, 4$) was 1.8 times faster than the optimized cuDNN LSTM network and eight times faster than VAT-LM-LSTM. This indicates that it is possible to use an even larger amount of unlabeled data in a practical time to further improve the performance of the EXN. In

Table 3.4 Number of tokens processed per second during the training

Method	Tokens/sec	Relative Speed
LM-LSTM	41,914	-
ADV-LM-LSTM	13,791	0.33x
VAT-LM-LSTM	9,602	0.23x
IMN ($c_i = 1$)	555,613	13.26x
IMN ($c_i = 1, 2$)	236,065	5.63x
IMN ($c_i = 1, 2, 3$)	122,076	2.91x
IMN ($c_i = 1, 2, 3, 4$)	75,393	1.80x

addition, note that each IMN can be trained in *parallel*. Thus, if multiple GPUs are available, the training can be carried out much faster than reported in Table 3.4.

3.7.3 Effect of Window Size of the IMN

In this section, we investigate the effectiveness of combining IMNs with different window sizes c_i on the final performance of the EXN. Figure 3.4 summarizes the results across all datasets. The figure shows that integrating an IMN with a greater window size consistently reduced the error rate, and the IMN with the greatest window size (**D**: $c_i = 1, 2, 3, 4$) achieved the best performance. This observation implies that the context, which is captured by a greater window size, contributes to the performance.

3.8 Discussion

3.8.1 Variations of the IMN

In this section, we discuss two possible variations of the IMN to better understand its effectiveness in the MEIN framework.

Incorporating IMN with Greater Window Size

As discussed in Section 3.7.3, Figure 3.4 demonstrates that increasing the window size of the IMN consistently improves the performance. From this observation, one may hypothesize that integrating an IMN with an even greater window size will be beneficial. Thus, we carried out an experiment with such a configuration, i.e., $c_i = 1, 2, 3, 4, 5$, and found that the hypothesis is valid. For example, the error rates of ADV-LM-LSTM+IMN ($c_i = 1, 2, 3, 4, 5$)

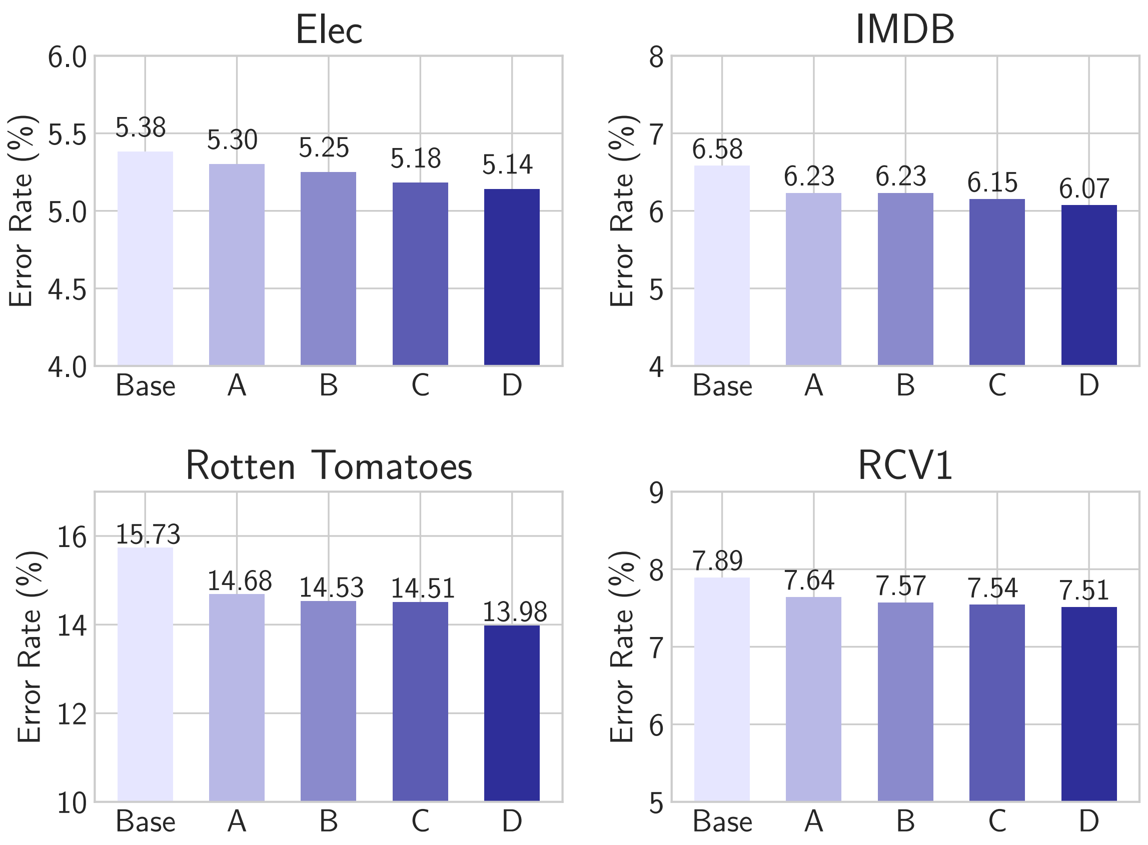


Figure 3.4 Effect of the IMN with different window sizes c_i on the final error rate (%) of ADV-LM-LSTM. **A lower error rate indicates better performance.** **Base**: EXN (ADV-LM-LSTM) without the IMN, **A**: $c_i = 1$, **B**: $c_i = 1, 2$, **C**: $c_i = 1, 2, 3$, **D**: $c_i = 1, 2, 3, 4$.

were 5.12% and 6.00% for Elec and IMDB, respectively, which are better than the values reported in Table 3.3.

However, we found that a large window size has a major drawback; the training of IMNs becomes significantly slower. This undesirable property must be avoided as our primary focus is the scalability with the amount of unlabeled data. Thus, we do not report these values as the main results of the experiment in Table 3.3.

Removing IMNs with Smaller Window Sizes

We also investigated the effectiveness of utilizing IMNs with smaller window size in addition to the larger window sizes. Table 3.5 gives the results of this investigation, and we can see that combining IMNs with smaller window sizes works better than incorporating a single IMN with the greatest window size.

Table 3.5 Effect of removing IMNs with smaller window sizes on the error rate (%) of ADV-LM-LSTM on the Elec dataset. **A lower error rate indicates better performance.**

Window Size	Error Rate (%)
$c_i = 1,2,3,4$	5.14
$c_i = 2,3,4$	5.18
$c_i = 3,4$	5.26
$c_i = 4$	5.23

3.8.2 Stronger Baseline DNN

In this section, we discuss the results of two attempts to improve the performance of baseline DNNs.

Increasing Number of Parameters

The most straightforward means of improving the performance of baseline DNNs is to increase the number of parameters. Thus, we doubled the word embedding dimension and trained ADV-LM-LSTM, namely, the ADV-LM-LSTM-Large model. This model has approximately the same number of parameters as the ADV-LM-LSTM+IMN. However, the performance did not improve from that of the original ADV-LM-LSTM. Specifically, the error rate degraded by 0.08 points for the IMDB dataset and was unchanged for the Elec dataset.

Combining ELMo

ELMo (Peters et al., 2018) is one of the strongest SSL approaches in the research field. Thus, we conducted an experiment with a baseline that utilizes ELMo. Specifically, we combined LSTM with the ELMo embeddings, namely, ELMo-LSTM[6]. The error rate of this network on the IMDB test set was 8.67%, which is worse than that of LM-LSTM reported in Table 3.3. This result suggests that, at least in this task setting, pre-training the RNN language model for initialization is more effective than using the ELMo embeddings.

3.9 Conclusion

In this chapter, we proposed a novel method for SSL, which we named Mixture of Expert/Imitator Networks (MEIN). The MEIN framework consists of a baseline DNN, i.e.,

[6]We used the implementation available in AllenNLP (Gardner et al., 2018).

an EXN, and several auxiliary networks, IMNs. The unique property of our method is that the IMNs learn to “imitate” the estimated label distribution of the EXN over the unlabeled data with only a limited view of the given input. In this way, the IMNs effectively learn a set of features that potentially contributes to improving the classification performance of the EXN.

Experiments on text classification datasets demonstrated that the MEIN framework consistently improved the performance of three distinct settings of the EXN. We also trained the IMNs with extra large-scale unlabeled data and achieved a new state-of-the-art result. This result indicates that our method has the *more data, better performance* property. Furthermore, our method operates eight times faster than the current strongest SSL method (VAT), and thus, it has promising scalability to the amount of unlabeled data.

Chapter 4

Massive Exploration of Pseudo Data for Grammatical Error Correction

4.1 Introduction

To date, a number of studies have tackled grammatical error correction (GEC) as a machine translation (MT) task, in which ungrammatical sentences are regarded as the source language and grammatical sentences are regarded as the target language. This approach allows for the use of cutting-edge neural MT models. For example, the encoder-decoder (EncDec) model (Bahdanau et al., 2015; Luong et al., 2015; Sutskever et al., 2014; Vaswani et al., 2017), which was originally proposed for MT, has been widely applied to GEC with remarkable results (Chollampatt and Ng, 2018; Ge et al., 2018; Grundkiewicz et al., 2019; Ji et al., 2017; Junczys-Dowmunt et al., 2018; Lichtarge et al., 2019; Xie et al., 2018; Yuan and Briscoe, 2016; Zhao et al., 2019).

However, a challenge in applying EncDec to GEC is that EncDec requires a large amount of training data (Koehn and Knowles, 2017); however, the largest set of publicly available parallel data (Lang-8) in GEC only contains 2 million sentence pairs (Mizumoto et al., 2011). The amount of available data is insufficient for the model to generalize to various grammatical errors. Consequently, there has been much research on methods for augmenting data by incorporating pseudo training data (Choe et al., 2019; Ge et al., 2018; Grundkiewicz et al., 2019; Lichtarge et al., 2019; Xie et al., 2018; Zhao et al., 2019).

When incorporating pseudo data, several decisions must be made regarding the experimental configurations, namely, (i) the method of generating the pseudo data, (ii) the seed corpus for the pseudo data, and (iii) the optimization setting (Section 4.2). However, a consensus on these decisions in the field of GEC has not been reached. For example, Xie et al. (2018) found that a variant of the back-translation (Sennrich et al., 2016b) method (BACKTRANS (NOISY)) outperformed the method of generating pseudo data from raw grammatical sentences (DIRECTNOISE). However, both the current state of the art model (Zhao et al., 2019) and the winner of the BEA-2019 shared task (Bryant et al., 2019; Grundkiewicz et al., 2019) use the DIRECTNOISE-based method.

In this chapter, we investigate the aforementioned decisions regarding pseudo data, with the aim to provide the research community with an improved understanding of the incorporation of pseudo data. Through massive amount of experiments, we explore and determine appropriate settings for GEC. In addition, we validate the reliability of the proposed settings by evaluating their performance on benchmark datasets. Specifically, without any task-specific techniques or architecture, our off-the-shelf EncDec method outperforms not only all previous single-model results but also all ensemble results, with the exception of the ensemble result by Grundkiewicz et al. (2019). By applying additional task-specific techniques, we further improve the model performance and achieved state-of-the-art performance on the CoNLL-2014 test set and the official test set of the BEA-2019 shared task.

4.2 Problem Formulation and Notation

In this section, we formally define the GEC task addressed in this chapter. Let $\mathscr{D}$ be the GEC training data that comprise pairs of an ungrammatical source sentence $\boldsymbol{X}$ and grammatical target sentence $\boldsymbol{Y}$ (i.e., $\mathscr{D} = \{(\boldsymbol{X}_n, \boldsymbol{Y}_n)\}_n$). Here, $|\mathscr{D}|$ denotes the number of sentence pairs in the dataset $\mathscr{D}$.

EncDec is currently the dominant approach to the GEC task (Chollampatt and Ng, 2018; Ji et al., 2017; Junczys-Dowmunt et al., 2018). To describe EncDec, we define $\boldsymbol{X}$ as consisting of a sequence of I tokens, namely, $\boldsymbol{X} = (x_1, \dots, x_I)$, where x_i denotes the i-th token of $\boldsymbol{X}$. Similarly, y_j denotes the j-th token of $\boldsymbol{Y}$. We define $\boldsymbol{Y}$ as always containing two additional special tokens; $\langle bos \rangle$ for y_0 and $\langle eos \rangle$ for y_{J+1}. Thus, $\boldsymbol{Y} = (y_0, y_1, \dots, y_J, y_{J+1})$, that is, the length of $\boldsymbol{Y}$ is always $J+2$. Then EncDec models the following conditional probability:

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{j=1}^{J+1} p(y_j|y_{0:j-1}, \boldsymbol{X}, \boldsymbol{\Theta}), \tag{4.1}$$

where $\boldsymbol{\Theta}$ represent all trainable parameters of the model. Our objective is to find the optimal parameter set $\widehat{\boldsymbol{\Theta}}$ that minimizes the following objective function $\mathscr{L}(\mathscr{D}, \boldsymbol{\Theta})$ for the given training data $\mathscr{D}$:

$$\mathscr{L}(\mathscr{D}, \boldsymbol{\Theta}) = -\frac{1}{|\mathscr{D}|} \sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathscr{D}} \log(p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\Theta})), \tag{4.2}$$

where $p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\Theta})$ denotes the conditional probability of $\boldsymbol{Y}$ given $\boldsymbol{X}$.

In a standard supervised learning setting, the parallel dataset $\mathscr{D}$ comprise only "genuine" parallel data $\mathscr{D}_g$ (i.e., $\mathscr{D} = \mathscr{D}_g$). However, in GEC, it is common to incorporate pseudo data $\mathscr{D}_p$ that are generated from grammatical sentences $\boldsymbol{Y} \in \mathscr{T}$, where $\mathscr{T}$ represents a *seed corpus* (i.e., set of grammatical sentences) (Grundkiewicz et al., 2019; Xie et al., 2018; Zhao et al., 2019).

Our interest lies in the following three nontrivial aspects of (4.2). **Aspect (i)**: There are multiple methods for generating pseudo data $\mathscr{D}_p$ (Section 4.3). **Aspect (ii)**: Options for the seed corpus $\mathscr{T}$ are numerous. To the best of our knowledge, the effect of the seed corpus domain on model performance is yet to be shown. We compare three corpora, namely, Wikipedia, Simple Wikipedia (SimpleWiki) and English Gigaword, as a first trial. Wikipedia and SimpleWiki have similar domains, but different grammatical complexities. Therefore, we can investigate how grammatical complexity affects model performance by comparing these two corpora. We assume that Gigaword contains the smallest amount of noise among the three corpora. We can therefore use Gigaword to investigate whether clean text improves model performance. **Aspect (iii)**: There are at least two major settings for incorporating $\mathscr{D}_p$ into the optimization of Equation 4.2. One is to use two datasets jointly by concatenating them as $\mathscr{D} = \mathscr{D}_g \cup \mathscr{D}_p$, and then solve the following minimization problem:

$$\widehat{\boldsymbol{\Theta}} = \operatorname*{arg\,min}_{\boldsymbol{\Theta}} \{\mathscr{L}(\mathscr{D}_g \cup \mathscr{D}_p, \boldsymbol{\Theta})\}. \tag{4.3}$$

We hereinafter refer to this optimization to as Joint.

The other setting is to use $\mathscr{D}_p$ for pretraining, namely, minimizing $\mathscr{L}(\mathscr{D}_p, \boldsymbol{\Theta})$ to acquire $\boldsymbol{\Theta}'$, and then fine-tuning the model by minimizing $\mathscr{L}(\mathscr{D}_g, \boldsymbol{\Theta}')$. Specifically, the optimization operates as follows:

$$\boldsymbol{\Theta}' = \operatorname*{arg\,min}_{\boldsymbol{\Theta}} \{\mathscr{L}(\mathscr{D}_p, \boldsymbol{\Theta})\} \tag{4.4}$$

$$\widehat{\boldsymbol{\Theta}} = \operatorname*{arg\,min}_{\boldsymbol{\Theta}} \{\mathscr{L}(\mathscr{D}_g, \boldsymbol{\Theta}')\}. \tag{4.5}$$

We refer to this optimization as PRETRAIN. We investigate the aforementioned aspects through extensive experiments (Section 4.4).

4.3 Methods for Generating Pseudo Data

In this section, we describe three methods for generating pseudo data: noisy back-translation (BACKTRANS (NOISY)), direct noizing (DIRECTNOISE), and its variant (DIRECTNOISE (SPELL)). In Section 4.4, we experimentally compare these methods. Examples of each generation method are presented in Figure 4.1.

Original:	`He died there , but the death date is not clear .`
BACKTRANS (NOISY):	`He died at there , but death date is not clear .`
DIRECTNOISE:	⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ , 2 but ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ is not ⟨*mask*⟩ ⟨*mask*⟩
DIRECTNOISE (SPELL):	`H@@ o died there , but the death te is not clear .`
Original:	`Gre@@ en@@ space Information for G@@ rea@@ ter London .`
BACKTRANS (NOISY):	`The information for Gre@@ en@@ space information about G@@ rea@@ ter London .`
DIRECTNOISE:	⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ for ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩
DIRECTNOISE (SPELL):	`Gre@@ en@@ space Information op@@ r@@ ts for G@@ rea@@ ter London .`
Original:	`The cli@@ p is mixed with images of Toronto streets during power failure .`
BACKTRANS (NOISY):	`The cli@@ p is mix with images of Toronto streets during power failure .`
DIRECTNOISE:	The ⟨*mask*⟩ is mixed ⟨*mask*⟩ images si@@ of The ⟨*mask*⟩ streets large ⟨*mask*⟩ power R@@ failure place ⟨*mask*⟩
DIRECTNOISE (SPELL):	`The V@@ li@@ p is mixed with images of Toronto streets during power failure .`
Original:	`At the in@@ stitute , she introduced tis@@ sue culture methods that she had learned in the U.@@ S.`
BACKTRANS (NOISY):	`At in@@ stitute , She introduced tis@@ sue culture method that she learned in U.@@ S.`
DIRECTNOISE:	⟨*mask*⟩ the the ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ tis@@ culture R@@ methods , she P ⟨*mask*⟩ the s U.@@ ⟨*mask*⟩
DIRECTNOISE (SPELL):	`At the in@@ stitute , she introduced tis@@ sue culture methods that she had l@@ Gen@@ em@@ d the U.@@ S.`

Figure 4.1 Examples of sentences generated by BACKTRANS (NOISY), DIRECTNOISE and DIRECTNOISE (SPELL) methods.

4.3.1 Noisy Back-translation: Backtrans (noisy)

Back-translation for the EncDec model was originally proposed for MT by Sennrich et al. (2016b). In back-translation, a reverse model, which generates a source (ungrammatical) sentence $\boldsymbol{X}$ from a given target (grammatical) sentence $\boldsymbol{Y}$, is trained. Then the output of the reverse model is paired with the input and used as pseudo data.

Currently, in the field of MT research, back-translation is considered to be the *de facto* standard method for generating pseudo data due to its strong empirical performance (Edunov et al., 2018; Haddow et al., 2018). However, this is not the case for GEC. Xie et al. (2018) reported that naively applying back-translation leads to only a minor improvement in performance; they demonstrate that in vanilla back-translation, the reverse model is generally too conservative and does not generate a sufficient quantity of grammatical errors. To overcome this issue, Xie et al. (2018) proposed a variant of back-translation[1] called BACKTRANS (NOISY). This method adds $r\beta_{\text{random}}$ to the score of each hypothesis in the beam for every time step. Here, scalar noise r is uniformly sampled from the interval $[0, 1]$, and $\beta_{\text{random}} \in \mathbb{R}_{\geq 0}$ is a hyper-parameter that controls the noise scale. If we set $\beta_{\text{random}} = 0$, then BACKTRANS (NOISY) is identical to vanilla back-translation.

4.3.2 Direct Noizing: DirectNoise and DirectNoise (spell)

Whereas BACKTRANS (NOISY) generates ungrammatical sentences with a reverse model, DIRECTNOISE injects noise "directly" into grammatical sentences (Choe et al., 2019; Edunov et al., 2018; Grundkiewicz et al., 2019; Zhao et al., 2019). Specifically, for each token in a given sentence, this method probabilistically selects one of the following operations: (i) masking with a placeholder token $\langle mask \rangle$, (ii) deletion, (iii) insertion of a random token sampled from unigram distribution, and (iv) keeping the original token. A detailed algorithm is provided in Algorithm 2. For each token, the selection is made based on the categorical distribution $(\mu_{\text{mask}}, \mu_{\text{deletion}}, \mu_{\text{insertion}}, \mu_{\text{keep}})$. The DIRECTNOISE algorithm is described in Algorithm 2.

Recently, Grundkiewicz et al. (2019) proposed another method for generating pseudo data. The central idea of their method is to use an off-the-shelf spell checker to generate a confusion set for a given word. Then, they probabilistically replace words with ones in the confusion sets. This method can be generally interpreted as a variant of DIRECTNOISE, in which the masking operation is discarded and the replacement operation is adopted. Thus, we refer to the method as DIRECTNOISE (SPELL).

[1]referred as "random noizing" in Xie et al. (2018)

Algorithm 2: DirectNoise Algorithm

```
Data: Grammatical sentence $\boldsymbol{Y} \in \mathscr{T}$
Result: Pseudo Corpus $\mathscr{D}_p$
1  $\mathscr{D}_p = \{\}$                                        // create empty set
2  $\boldsymbol{\mu} = (\mu_{\text{mask}}, \mu_{\text{deletion}}, \mu_{\text{insertion}}, \mu_{\text{keep}})$ s.t. $\Sigma\boldsymbol{\mu} = 1$
3  for $Y \in \mathscr{T}$ do
4      $\boldsymbol{X} = (\,)$
5      for $j \in (1, \dots, J)$ do
6          $action \sim Cat(action|\boldsymbol{\mu})$
7          if action is keep then
8              append $y_j$ to $\boldsymbol{X}$
9          else if action is mask then
10             append $\langle mask \rangle$ to $\boldsymbol{X}$
11         else if action is deletion then
12             continue
13         else if action is insertion then
14             append $y_j$ to $\boldsymbol{X}$
15             $w \sim unigram\_distribution(\mathscr{D}_g)$
16             append $w$ to $\boldsymbol{X}$
17     $\mathscr{D}_p = \mathscr{D}_p \cup \{(\boldsymbol{X}, \boldsymbol{Y})\}$
```

4.4 Experiments

The goal of our experiments is to investigate **Aspect (i)–(iii)** introduced in Section 4.2. We design our experiments to ensure that the experimental findings are reproducible and generally applicable to GEC (Bouthillier et al., 2019). Specifically, the experiments are based on the following two strategies. (i) We use an off-the-shelf EncDec model without any task-specific architecture or techniques. (ii) We perform hyperparameter tuning, evaluation, and comparison for each method and setting on a validation set. In Section 4.4.5, we summarize our findings and propose appropriate settings. We then evaluate their performance on multiple benchmark test sets.

4.4.1 Experimental Configurations

Dataset The BEA-2019 workshop official dataset (Bryant et al., 2019) is the origin of the training, validation and test data of our experiments. This dataset consists of the following corpora: the First Certificate in English corpus (Yannakoudakis et al., 2011), Lang-8 Cor-

Table 4.1 Summary of datasets used in our experiments. Dataset marked with “*” is a seed corpus $\mathcal{T}$.

Dataset	#sent (pairs)	#refs.	Split	Scorer
BEA-train	561,410	1	train	-
BEA-valid	4,384	1	valid	ERRANT
CoNLL-2014	1,312	2	test	ERRANT & M^2 scorer
JFLEG	1,951	4	test	GLEU
BEA-test	4,477	5	test	ERRANT
SimpleWiki*	1,369,460	-	-	-
Wikipedia*	145,883,941	-	-	-
Gigaword*	131,864,979	-	-	-

pus of Learner English (Lang-8) (Mizumoto et al., 2011; Tajiri et al., 2012), the National University of Singapore Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013), and W&I+LOCNESS (Granger, 1998; Yannakoudakis et al., 2018)[2]. Hereinafter, we refer to the training data as BEA-train.

The BEA-train is tokenized using the `spaCy` tokenizer[3][4]. We remove sentence pairs that have identical source and target sentences from the training set, following (Chollampatt and Ng, 2018). We then acquire subwords from target sentences through the byte-pair-encoding (BPE) (Sennrich et al., 2016c) algorithm. We use `subword-nmt` implementation[5]. We apply BPE splitting to both source and target text. The number of merge operations is set to 8,000.

As a seed corpus $\mathcal{T}$, we use SimpleWiki[6], Wikipedia[7], or Gigaword[8]. We apply the noizing methods described in Section 4.3 to each corpus and generate pseudo data $\mathcal{D}_p$[9]. The characteristics of each dataset are summarized in Table 4.1.

Evaluation We report results on BEA-valid, the official test set of the BEA-2019 shared task (BEA-test), the CoNLL-2014 test set (CoNLL-2014) (Ng et al., 2014), and the JFLEG test set (JFLEG) (Napoles et al., 2017). All reported results (except for ensemble) are the average of five distinct trials using five different random seeds. We report the scores mea-

[2]The data are publicly available at https://www.cl.cam.ac.uk/research/nl/bea2019st/.

[3]https://spacy.io/

[4]We use the model file `en_core_web_sm-2.1.0` available at https://github.com/explosion/spacy-models/releases/tag/en_core_web_sm-2.1.0

[5]https://github.com/rsennrich/subword-nmt

[6]https://simple.wikipedia.org

[7]We use the `2019-02-25` dump file at https://dumps.wikimedia.org/other/cirrussearch/.

[8]We use English Gigaword Fifth Edition (LDC Catalog No.: LDC2011T07).

[9]The original implementation of DirectNoise (spell) is not publicly available. Instead, we use an in-house re-implementation of the method with the hyperparameters described in the paper (Grundkiewicz et al., 2019).

sured by ERRANT (Bryant et al., 2017; Felice et al., 2016)[10] for BEA-valid, BEA-test, and CoNLL-2014. Because the reference sentences of BEA-test are not publicly available, we evaluate the model outputs on the `CodaLab`[11] platform for BEA-test. We also report results measured by the M^2 scorer (Dahlmeier and Ng, 2012) on CoNLL-2014 for comparison with the results of previous studies. We use the GLEU metric (Napoles et al., 2015, 2016) for JFLEG.

Model We adopt the *Transformer* EncDec model (Vaswani et al., 2017) for each experiment. Specifically, we use the implementation available in the `fairseq` toolkit (Ott et al., 2019) and "Transformer (big)" settings of Vaswani et al. (2017), in which both the encoder and decoder have six layers with 16 attention heads in each layer, a word embedding size d_{model} of 1,024, and a feed-forward network size d_{ff} of 4,096. The dropout rate is set to 0.3.

Optimization We compare two optimization settings, namely, JOINT and PRETRAIN. For the JOINT setting, we optimize the model with Adam (Kingma and Ba, 2015). For the PRETRAIN setting, we pretrain the model with Adam and then fine-tune it on BEA-train using Adafactor (Shazeer and Stern, 2018). The detailed hyperparameters for each setting are provided in Table 4.2 and Table 4.3.

[10]https://github.com/chrisjbryant/errant
[11]https://competitions.codalab.org/competitions/20228

Table 4.2 Hyper-parameter for JOINT optimization

Model Architecture	Transformer (Vaswani et al., 2017) (“big” setting)
Optimizer	Adam (Kingma and Ba, 2015) ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$)
Learning Rate Schedule	Same as described in Section 5.3 of Vaswani et al. (2017)
Number of Epochs	40
Dropout	0.3
Stopping Criterion	Train model for 40 epochs. During the training, save model parameter for every epoch. Then take average of last 20 checkpoints.
Gradient Clipping	1.0
Loss Function	Label smoothed cross entropy (smoothing value: $\epsilon_{ls} = 0.1$) (Szegedy et al., 2016)
Beam Search	Beam size 5 with length-normalization

Table 4.3 Hyper-parameter for PRETRAIN optimization

Pretraining	
Model Architecture	Transformer (Vaswani et al., 2017) ("big" setting)
Optimizer	Adam (Kingma and Ba, 2015) ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$)
Learning Rate Schedule	Same as described in Section 5.3 of Vaswani et al. (2017)
Number of Epochs	10
Dropout	0.3
Gradient Clipping	1.0
Loss Function	Label smoothed cross entropy (smoothing value: $\epsilon_{ls} = 0.1$) (Szegedy et al., 2016)
Fine-tuning	
Model Architecture	Transformer (Vaswani et al., 2017) ("big" setting)
Optimizer	Adafactor (Shazeer and Stern, 2018)
Learning Rate Schedule	Constant learning rate of 3×10^{-5}
Number of Epochs	30
Dropout	0.3
Stopping Criterion	Use the model with the best validation perplexity on BEA-valid
Gradient Clipping	1.0
Loss Function	Label smoothed cross entropy (smoothing value: $\epsilon_{ls} = 0.1$) (Szegedy et al., 2016)
Beam Search	Beam size 5 with length-normalization

Table 4.4 Performance of models on BEA-valid: a value in **bold** indicates the best result within the column. The seed corpus $\mathcal{T}$ is SimpleWiki.

Method	Prec.	Rec.	$F_{0.5}$
Baseline	45.2	22.7	37.7
BACKTRANS (NOISY)	41.7	**31.2**	39.1
DIRECTNOISE	**48.3**	25.4	**40.9**
DIRECTNOISE (SPELL)	44.6	29.5	40.4

4.4.2 Aspect (i): Pseudo Data Generation

We compare the effectiveness of the BACKTRANS (NOISY), DIRECTNOISE, and DIRECTNOISE (SPELL) methods for generating pseudo data. To do this, we first investigate the hyperparameters suitable for BACKTRANS (NOISY) and DIRECTNOISE. Then, we make a comparison of BACKTRANS (NOISY), DIRECTNOISE, and DIRECTNOISE (SPELL). It should be noted that

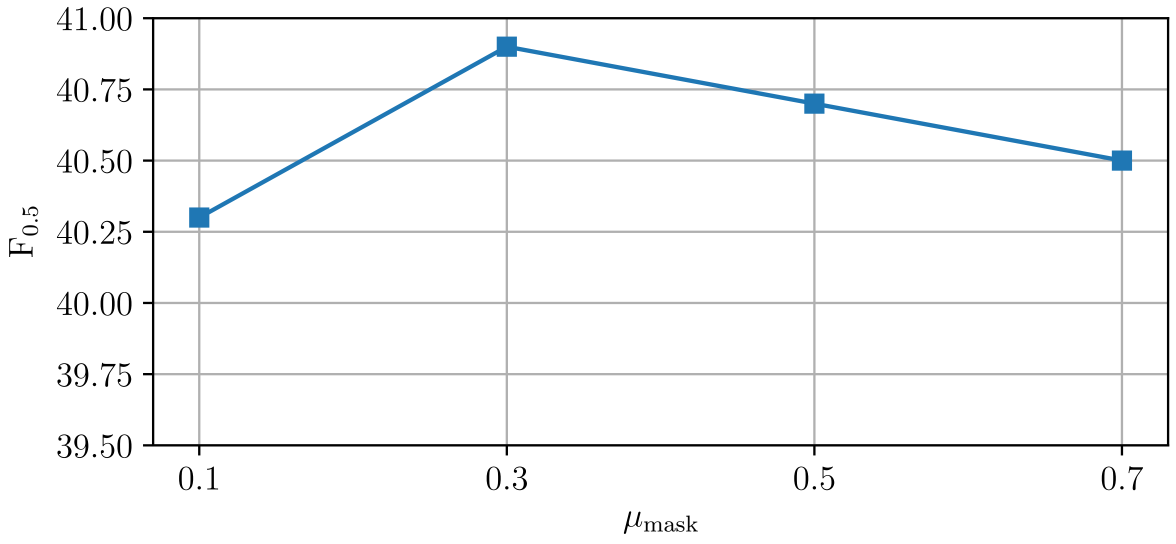


Figure 4.2 Performance of the model on BEA-valid with varying parameters DIRECTNOISE (μ_{mask}).

throughout this section, we use (i) the JOINT setting and (ii) all of SimpleWiki as the seed corpus $\mathscr{T}$.

As described in Section 4.3.2, DIRECTNOISE contains four hyperparameters: ($\mu_{\text{mask}}, \mu_{\text{deletion}}, \mu_{\text{insertion}}, \mu_{\text{keep}}$). Running a grid search over all of these parameters is computationally expensive: thus, in this thesis, we exclusively focus on the effect of μ_{mask}. Therefore we deliberately fix $\mu_{\text{keep}} = 0.2$, and use $\mu_{\text{insertion}} = \mu_{\text{deletion}} = (1 - \mu_{\text{keep}} - \mu_{\text{mask}})/2$. The results are summarized in Figure 4.2. Here, $\mu_{\text{mask}} = 0.3$ exhibits the best performance; therefore, we use $\mu_{\text{mask}} = 0.3$ for the remainder of the experiments.

We also investigate the effect of varying β_{random} of BACKTRANS (NOISY) by evaluating its performance on BEA-valid (Figure 4.3). Here, $\beta_{\text{random}} = 6$ achieves the best performance. It should be noted that according to Figure 4.3, the performance of back-translation without noise ($\beta_{\text{random}} = 0$) is worse than the baseline. We found that given no noise, the reverse model becomes too conservative, and does not generate grammatical errors, which is consistent with the phenomenon reported by Xie et al. (2018). As a result, the pseudo data does not provide useful teaching signals for the model and eventually harm the performance.

Given the suitable hyperparameters for μ_{mask} and β_{random}, we compare the performance of BACKTRANS (NOISY), DIRECTNOISE, and DIRECTNOISE (SPELL). The results are presented in Table 4.4. The table shows that DIRECTNOISE, DIRECTNOISE (SPELL), and BACKTRANS (NOISY) all achieve better $F_{0.5}$ than the baseline, which is consistent with the result reported by previous studies (Grundkiewicz et al., 2019; Xie et al., 2018; Zhao et al., 2019). It is

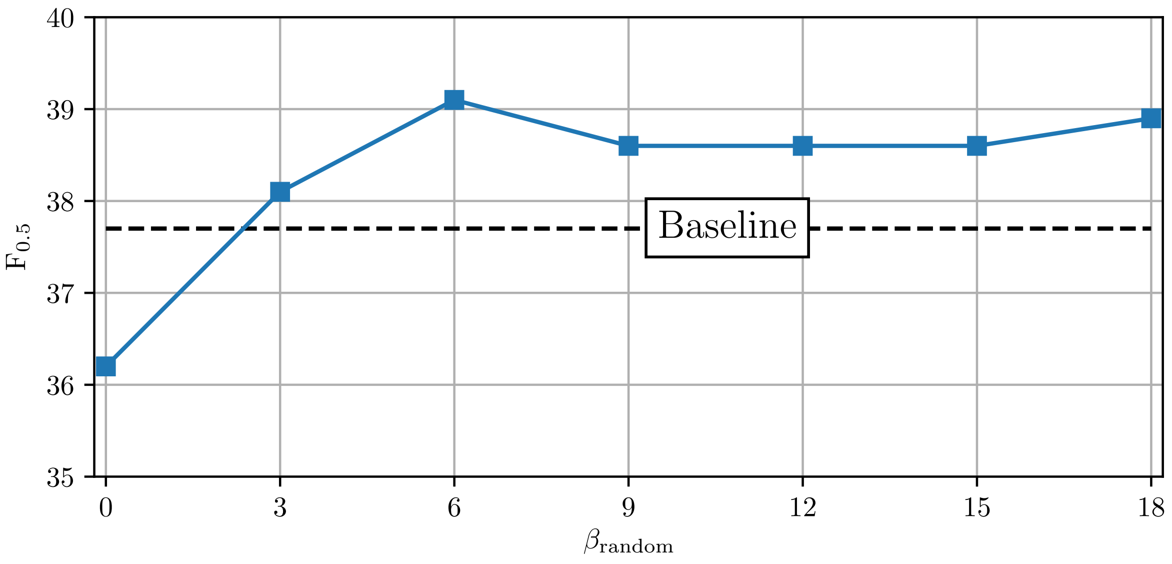


Figure 4.3 Performance of the model on BEA-valid with varying parameters BACKTRANS (NOISY) (β_{random}).

noteworthy that each method improves different metrics; DIRECTNOISE improves precision, whereas BACKTRANS (NOISY) and DIRECTNOISE (SPELL) improve recall. In addition, DIRECTNOISE achieves superior $F_{0.5}$ to DIRECTNOISE (SPELL). This is surprising because DIRECTNOISE does not rely on the external spell checker. We will exclusively use DIRECTNOISE and BACKTRANS (NOISY) for the rest of the experiments, because they achieve the highest precision and recall, respectively in Table 4.4.

Why do different methods improve different metrics? We speculate that this result is related to the quality of the language model of EncDec. In the GEC literature, several studies reported that a better language model leads to improved precision. For example, Junczys-Dowmunt and Grundkiewicz (2016) incorporated an n-gram language model trained from the Web-scale dataset and improved the precision of the vanilla statistical MT model by almost 10 points. In addition, recently, Chollampatt et al. (2019) used the scores computed by a pretrained language model (BERT (Devlin et al., 2019)) as features for re-ranking the outputs of the system; they reported an improved precision over the model without BERT. We speculate that a similar effect is occurring in a model trained with DIRECTNOISE, thanks to the existence of the $\langle mask \rangle$ token. Here, the decoder of the model cannot rely on the encoder's hidden states to generate the sentence, because the $\langle mask \rangle$ token removes information from the source sentence. In other words, the decoder is biased toward developing a better language model, rather than merely copying the source sentence.

Table 4.5 Performance on BEA-valid when changing the seed corpus $\mathscr{T}$ used for generating pseudo data ($|\mathscr{D}_p| = 1.4\text{M}$).

Method	Seed Corpus $\mathscr{T}$	Prec.	Rec.	$F_{0.5}$
Baseline	N/A	45.2	22.7	37.7
BACKTRANS (NOISY)	Wikipedia	42.8	30.5	39.6
BACKTRANS (NOISY)	SimpleWiki	41.7	31.2	39.1
BACKTRANS (NOISY)	Gigaword	42.2	33.1	40.0
DIRECTNOISE	Wikipedia	47.1	25.8	40.4
DIRECTNOISE	SimpleWiki	48.3	25.4	40.9
DIRECTNOISE	Gigaword	47.3	26.7	41.0

4.4.3 Aspect (ii): Seed Corpus $\mathscr{T}$

We investigate the effectiveness of the seed corpus $\mathscr{T}$ on generating pseudo data $\mathscr{D}_p$. The three corpora (Wikipedia, SimpleWiki and Gigaword) are compared in Table 4.5. We set $|\mathscr{D}_p| = 1.4\text{M}$. The difference in $F_{0.5}$ is small, which implies that the seed corpus $\mathscr{T}$ has only a minor effect on the model performance. Nevertheless, Gigaword consistently outperforms the other two corpora. In particular, DIRECTNOISE with Gigaword achieves the best $F_{0.5}$ value among all configurations. This is a positive result for the GEC community, as Gigaword is a collection of news articles, that can be collected in high quantities at a relatively low cost (e.g., News Crawl[12]).

4.4.4 Aspect (iii): Optimization Setting

Here, we compare JOINT and PRETRAIN optimization settings. We are interested in the performance of each setting when the scale of the pseudo data $\mathscr{D}_p$ is (i) approximately the same ($|\mathscr{D}_p| = 1.4\text{M}$) and (ii) substantially larger ($|\mathscr{D}_p| = 14\text{M}$) than that of genuine parallel data $\mathscr{D}_g$ ($|\mathscr{D}_g| \approx 500\text{K}$).

In the case of (ii), we expect that the teaching signal from the pseudo data $\mathscr{D}_p$ becomes dominant in the JOINT setting, and thus, the model may fail to learn from the genuine data $\mathscr{D}_g$. We call this potential problem as **dominant pseudo data**. In order to alleviate this problem, we experiment with upsampling the genuine data $\mathscr{D}_g$ in JOINT, namely, JOINT (UPSAMPLE). Specifically, we search for the appropriate upsampling rate within the values {1, 2, 4, 8, 16, 25} on BEA-valid. Here, 1 is equivalent to JOINT (without upsampling) and 25

[12]http://data.statmt.org/news-crawl/

Table 4.6 Performance of the model with different optimization settings on BEA-valid. The seed corpus $\mathscr{T}$ is Gigaword.

Optimization	Method	$\lvert\mathscr{D}_p\rvert$	Prec.	Rec.	$F_{0.5}$
N/A	Baseline	0	45.2	22.7	37.7
PRETRAIN	BACKTRANS (NOISY)	1.4M	49.2	27.2	42.3
PRETRAIN	DIRECTNOISE	1.4M	47.2	23.3	39.1
JOINT	BACKTRANS (NOISY)	1.4M	42.2	33.1	40.0
JOINT	DIRECTNOISE	1.4M	47.3	26.7	41.0
PRETRAIN	BACKTRANS (NOISY)	14M	50.8	32.6	45.6
PRETRAIN	DIRECTNOISE	14M	48.3	27.7	42.0
JOINT	BACKTRANS (NOISY)	14M	40.8	37.1	40.0
JOINT	DIRECTNOISE	14M	48.0	25.0	40.5
JOINT (UPSAMPLE)	BACKTRANS (NOISY)	14M	41.8	37.8	40.9
JOINT (UPSAMPLE)	DIRECTNOISE	14M	47.0	28.0	41.4

approximately corresponds to a ratio of $\lvert\mathscr{D}_p\rvert : \lvert\mathscr{D}_g\rvert = 1 : 1$. As a result, an upsampling rate of 2 achieved the best $F_{0.5}$ on BEA-valid.

Joint Training or Pretraining

Table 4.6 presents the results. First, let us compare the result of JOINT and JOINT (UPSAMPLE). In BACKTRANS (NOISY), increasing $\lvert\mathscr{D}_p\rvert$ (1.4M → 14M) does not improve $F_{0.5}$ on JOINT (40.0 → 40.0). On the other hand, by upsampling the genuine data $\mathscr{D}_g$, JOINT (UPSAMPLE) improves $F_{0.5}$ (40.0 → 40.9). These results imply that **dominant pseudo data** indeed exists in vanilla JOINT, and upsampling genuine data can alleviate such a problem.

Second, the table shows that PRETRAIN is superior to JOINT, especially in terms of the properties of *more pseudo data and better performance*. For example, in BACKTRANS (NOISY), increasing $\lvert\mathscr{D}_p\rvert$ (1.4M → 14M) improves $F_{0.5}$ on PRETRAIN by more than two points (42.3 → 45.6). This is significantly larger than the improvement achieved by JOINT with upsampling (JOINT (UPSAMPLE)); it only improves by 0.9 point (40.0 → 40.9). An intuitive explanation for this result is that PRETRAIN effectively handles **dominant pseudo data**, because the model is trained only with $\mathscr{D}_g$ during the fine-tuning phase. Therefore, we conclude that PRETRAIN is the best optimization setting in Table 4.6.

Table 4.7 Comparison of our best model and current top models: a **bold** value indicates the best result within the column.

		CoNLL-2014 (M^2 scorer)			CoNLL-2014 (ERRANT)			JFLEG	BEA-test (ERRANT)		
Model	Ensemble	Prec.	Rec.	$F_{0.5}$	Prec.	Rec.	$F_{0.5}$	GLEU	Prec.	Rec.	$F_{0.5}$
Chollampatt and Ng (2018)		60.9	23.7	46.4	-	-	-	51.3	-	-	-
Junczys-Dowmunt et al. (2018)		-	-	53.0	-	-	-	57.9	-	-	-
Grundkiewicz and Junczys-Dowmunt (2018)		66.8	34.5	56.3	-	-	-	61.5	-	-	-
Lichtarge et al. (2019)		65.5	37.1	56.8	-	-	-	61.6	-	-	-
Chollampatt and Ng (2018)	✓	65.5	33.1	54.8	-	-	-	57.5	-	-	-
Junczys-Dowmunt et al. (2018)	✓	61.9	40.2	55.8	-	-	-	59.9	-	-	-
Lichtarge et al. (2019)	✓	66.7	43.9	60.4	-	-	-	**63.3**	-	-	-
Zhao et al. (2019)	✓	71.6	38.7	61.2	-	-	-	61.0	-	-	-
Grundkiewicz et al. (2019)	✓	-	-	64.2	-	-	-	61.2	**72.3**	60.1	69.5
PretLarge		67.9	44.1	61.3	61.2	42.0	56.0	59.7	65.5	59.4	64.2
PretLarge+SSE		68.7	45.2	62.2	62.1	42.7	57.0	60.9	66.1	60.9	65.0
PretLarge+SSE+R2L	✓	**72.4**	**46.1**	**65.0**	**67.3**	**44.0**	**60.9**	61.4	72.1	**61.8**	**69.8**

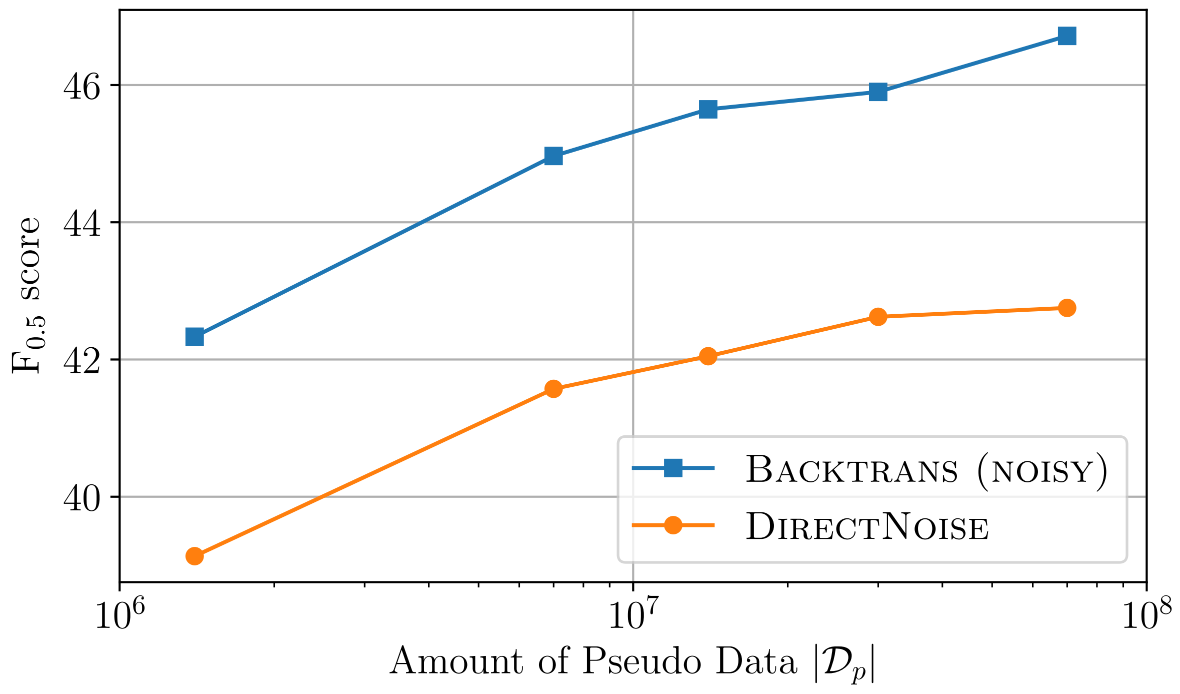


Figure 4.4 Performance on BEA-valid for different amounts of pseudo data ($|\mathscr{D}_p|$). The seed corpus $\mathscr{T}$ is Gigaword.

Amount of Pseudo Data

We investigate the effect of increasing the amount of pseudo data on the PRETRAIN setting. To do this, we pretrain the model with different amounts of pseudo data {1.4M, 7M, 14M, 30M, 70M}. The results in Figure 4.4 demonstrate that the sample efficiency of BACKTRANS (NOISY) is superior to that of DIRECTNOISE. The best model (pretrained with 70M BACKTRANS (NOISY)) achieves $F_{0.5} = 46.7$.

4.4.5 Comparison with Current Top Models

The experimental results thus far indicate that the following configurations improve model performance: (i) the combination of JOINT and Gigaword (Section 4.4.3), (ii) an amount of pseudo data $\mathscr{D}_p$ in JOINT that is not too large (Section 4.4.4), and (iii) PRETRAIN with BACKTRANS (NOISY) using a large amount of pseudo data $\mathscr{D}_p$ (Section 4.4.4). We summarize these findings and attempt to combine PRETRAIN and JOINT. Specifically, we pretrain the model using 70M pseudo data of BACKTRANS (NOISY). We then fine-tune the model by combining BEA-train and a relatively small amount of DIRECTNOISE pseudo data generated from Gigaword (where $|\mathscr{D}_p| = 250\text{K}$). However, the performance does not improve on BEA-valid. Therefore, the best available approach is simply to pretrain the model with a large amount

(70M) of Backtrans (noisy) pseudo data and then fine-tune using BEA-train, which we hereinafter refer to as PretLarge. We use Gigaword for the seed corpus $\mathscr{T}$ because it has the best performance, as illustrated in Table 4.5.

We evaluate the performance of PretLarge on test sets and compared the scores with those of current top models. It is important to note that CoNLL-2014, JFLEG, and BEA-test all involve different domains. For example, CoNLL-2014 consists of essays, while BEA-test contains a much broader type of texts, such as letters, stories, and articles. Therefore, achieving high performance on multiple test sets should ensure that our model's superiority is valid across GEC datasets in general. Table 4.7 reveals that PretLarge achieves $F_{0.5} = 61.3$ on CoNLL-2014, a result that outperforms not only all previous single-model results but also all ensemble results except for that by Grundkiewicz et al. (2019).

To further improve the performance, we incorporate the following two techniques that are widely used in shared tasks such as BEA-2019 (Bryant et al., 2019) and WMT (Barrault et al., 2019):

Synthetic Spelling Error (SSE) Lichtarge et al. (2019) proposed a method of probabilistically injecting character-level noise into a source sentence of pseudo data $\mathscr{D}_p$. Specifically, one of the following operations is applied randomly at a rate of 0.003 per character: deletion, insertion, replacement, or transposition of adjacent characters.

Right-to-left Re-ranking (R2L) Incorporating a right-to-left model into the decoding process was independently proposed by Liu et al. (2016) and Sennrich et al. (2016a), and consistent improvements in performance were reported. Following previous studies (Grundkiewicz et al., 2019; Morishita et al., 2019; Sennrich et al., 2017, 2016a), we train four right-to-left models. The ensemble of four left-to-right models generate n-best candidates and their corresponding scores (i.e., conditional probabilities). We then pass each candidate to the ensemble of the four right-to-left models and compute the scores. Finally, we re-rank the original n-best candidates based on the sum of the two scores. We set $n = 5$.

Table 4.7 presents the results of applying SSE and R2L. PretLarge+SSE+R2L achieves state-of-the-art performance on both CoNLL-2014 ($F_{0.5} = 65.0$) and BEA-test ($F_{0.5} = 69.8$), which is superior to that of the best system in the BEA-2019 shared task (Grundkiewicz et al., 2019).

4.5 Analysis

In Section 4.4.5, we demonstrate that PretLarge and PretLarge+SSE configurations achieve superior performance to that of current top models. In this section, we propose future directions for further improving the performance. We achieve this by analyzing our

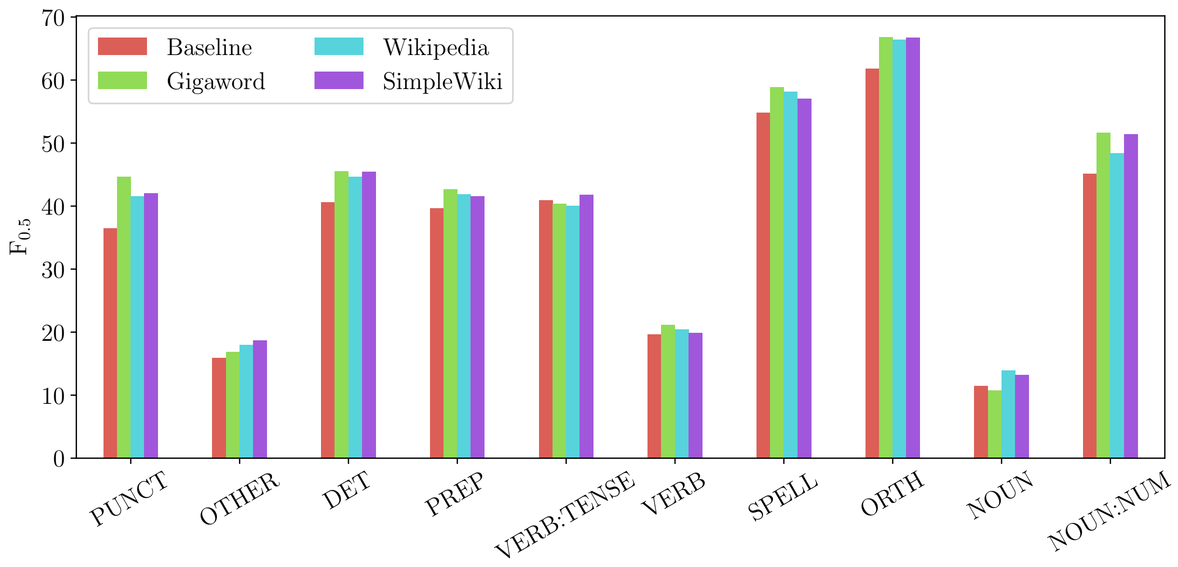


Figure 4.5 Performance of the different seed corpora on BEA-valid across various error types. Only the 10 most frequent error types are presented. For a detailed description of each error type, see (Bryant et al., 2017).

experimental results in terms of grammatical error type performance and proficiency levels. Specifically, we conduct an analysis on the following two aspects: (i) how the effectiveness of the pseudo data varies across different seed corpora (Section 4.5.1), and (ii) the strengths and weaknesses of PretLarge (Section 4.5.2).

4.5.1 Effectiveness of Different Seed Corpora

Error Type Analysis

In this section, we analyze the performance of the models trained with different seed corpora through their performance on each error type on BEA-valid[13]. Here, the question is whether one seed corpus is more effective for correcting certain grammatical error types than the other corpora. To do this, we use the DirectNoise models illustrated in Table 4.5. Figure 4.5 presents the results. The figure shows that different seed corpora indeed have different characteristics. For example, Gigaword outperforms other seed corpora on error types such as PUNCT, PREP, and SPELL. On the other hand, SimpleWiki outperforms Gigaword on the VERB:TENSE error.

[13]BEA-valid contains error type annotation for each edit, that is automatically annotated by ERRANT.

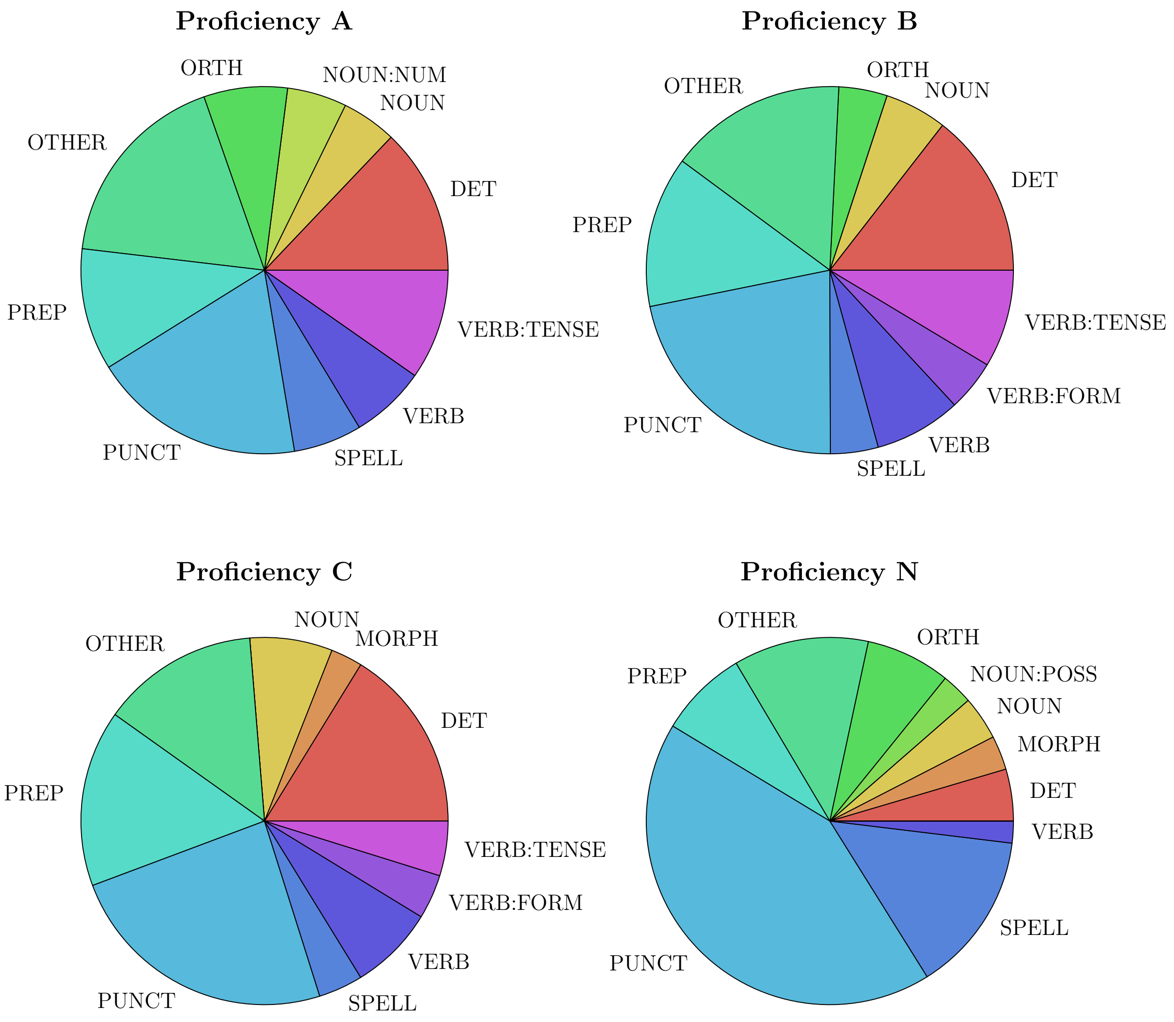


Figure 4.6 Error type distribution across different proficiency levels. Only the 10 most frequent error types are presented.

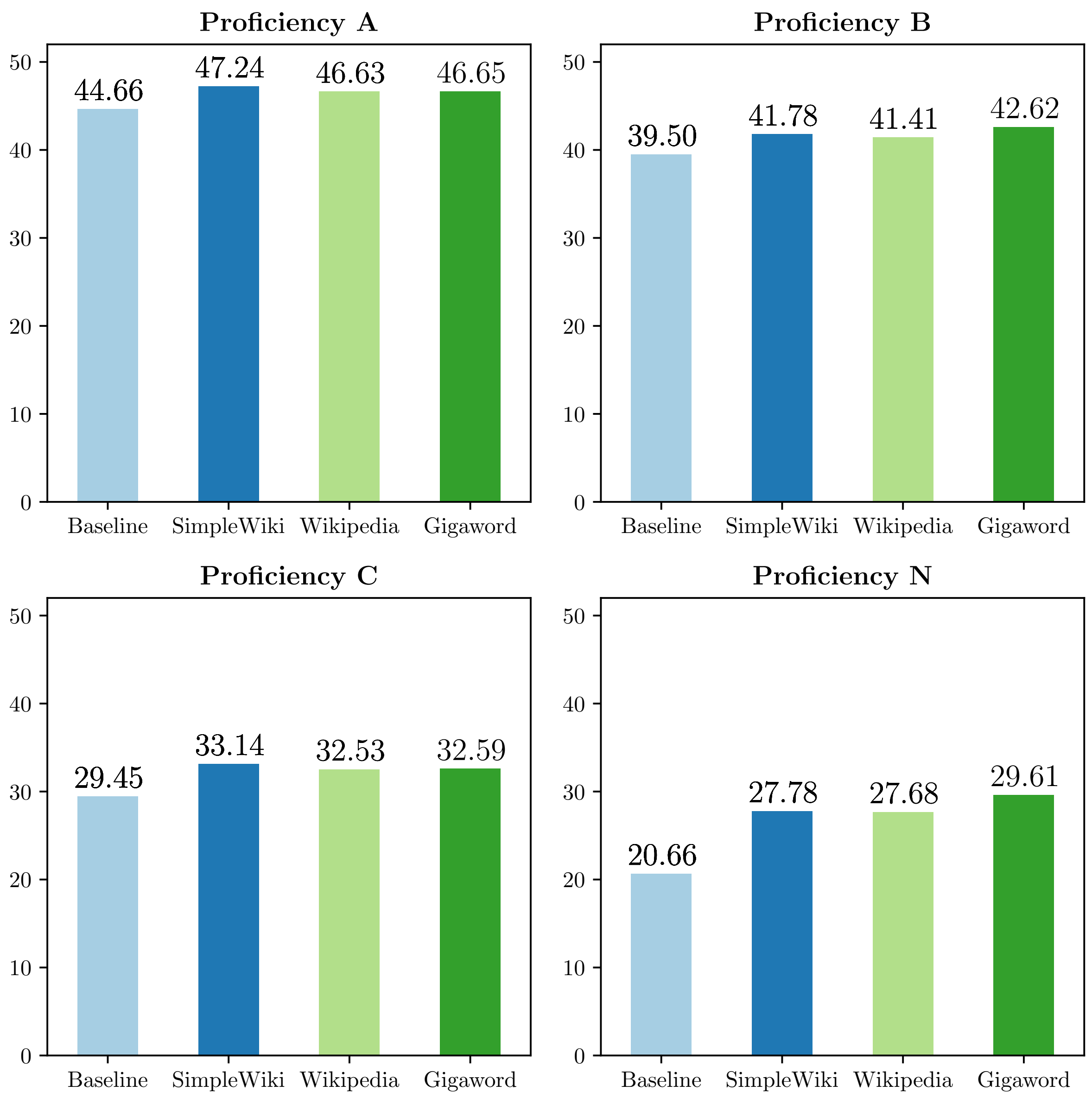


Figure 4.7 Effect of the seed corpus on proficiency A, B, B, and N. The y-axis represents the $F_{0.5}$ score.

Source Sentence:	When the concert finished , we went to cloakroom to get signatures from musicians .
Gold Sentence:	When the concert finished , we went to **the** dressing room to get autographs from musicians .
Model Output:	When the concert finished , we went to **the** cloakroom to get signatures from musicians .

Figure 4.8 NOUN error generated by our model (PRETLARGE+SSE). **Bold text** indicates the grammatical error successfully corrected by the model. Underlined text indicates the grammatical errors not corrected by the model.

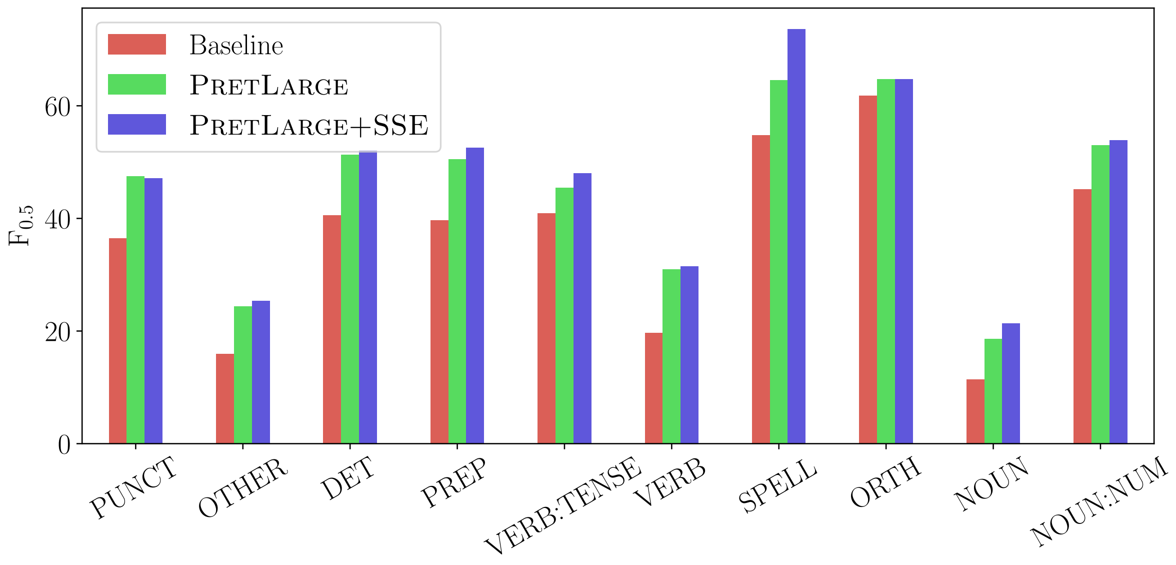


Figure 4.9 Performance of the models on BEA-valid across various error types. Only the 10 most frequent error types are presented. For a detailed description of each error type, see (Bryant et al., 2017).

Proficiency-wise Analysis

One notable characteristic of BEA-valid is that it comprises four sections (A, B, C and N) with different English proficiency levels. Here, A (beginner), B (intermediate), and C (advanced) are derived from CEFR levels (Little, 2006). The remaining level (N) corresponds to text written by native English speakers. These proficiency levels provide us with increased insight into the behavior of the model. This is because, as illustrated in Figure 4.6, the error distribution differs significantly among different proficiency levels. For example, the determiner (DET) error is common in proficiency levels A, B, and C, but not N.

We are interested in the effect of changing the seed corpus on the performance of the model for each proficiency level, as each proficiency level should require a different seed corpus. For example, SimpleWiki should be suitable for proficiency levels A, B, and C because its grammatical complexity is closer to that of English learners. Similarly, Gigaword should be suitable for proficiency level N because its text is written by native English speakers. We analyze this relationship between the seed corpus and proficiency levels using the DirectNoise models illustrated in Table 4.5. Specifically, we evaluated the performance of the model for each proficiency level. Figure 4.7 presents the results. The figure indicates that SimpleWiki has either comparative or superior performance to Gigaword for proficiency levels A, B, and C. However, Gigaword outperforms the other two corpora by almost two points in $F_{0.5}$ score for proficiency level N. These results support our hypothesis that there is a relationship between the grammatical complexity of the seed corpus and the proficiency level.

The fact that a certain seed corpus is more suitable for a certain proficiency level than the other corpora is consistent with our findings in Section 4.5.1. For example, according to Figure 4.6, the VERB:TENSE error is in the 10 most frequent errors in proficiency levels A, B, and C. In Section 4.5.1, we found that SimpleWiki demonstrates the best performance on the correction of the VERB:TENSE error (Figure 4.5). This may be one of the reasons why SimpleWiki shows strong performance on the proficiency levels A, B, and C in Figure 4.7.

4.5.2 Strengths and Weaknesses of PretLarge

Error Type Analysis

We analyze each model through its performance for each error type on BEA-valid (Figure 4.9). Specifically, we are interested in the performance of PRETLARGE and PRETLARGE+SSE compared to that of the baseline model, which is only trained with genuine data $\mathscr{D}_g$. It should be noted that BEA-valid contains error type annotation for each edit, that is automatically annotated by ERRANT.

Figure 4.9 reveals that PRETLARGE improves the performance across all error types compared to the baseline model. In addition, it is surprising that PRETLARGE+SSE improves the performance of PRETLARGE not only for the SPELL error type but also for most other error types. We speculate that incorporating SSE makes the model more robust against noise.

Figure 4.9 also indicates a major weakness of our models, that is, they perform relatively poorly for content word errors, such as NOUN and VERB. This shortcoming is common across GEC models in general; a similar trend is observed in the error type performance of systems participating in the BEA-2019 shared task (Bryant et al., 2019). One reason for this is that there is an insufficient number of content word errors in both genuine data and pseudo data. For example, as illustrated in Figure 4.6, the ratio of NOUN and VERB errors is smaller than the ratio of errors such as PREP, PUNCT, and DET. Thus, developing a method that exclusively generates content word errors is an important direction for future work.

A qualitative analysis through an example in Table 4.8 provides us with other directions for improving the GEC model. Here, the model (PRETLARGE+SSE) successfully corrected the DET error by inserting the missing "the" token. However, the model ignored two NOUN errors in the sentence: one is to replace "cloakroom" with "dressing room", and the other is to replace "signature" with "autograph." For the former error, the model needs to know that the "musicians" are likely to be in the "dressing room" rather than in the "cloakroom." One possible way to do this is to develop a methodology of injecting external commonsense

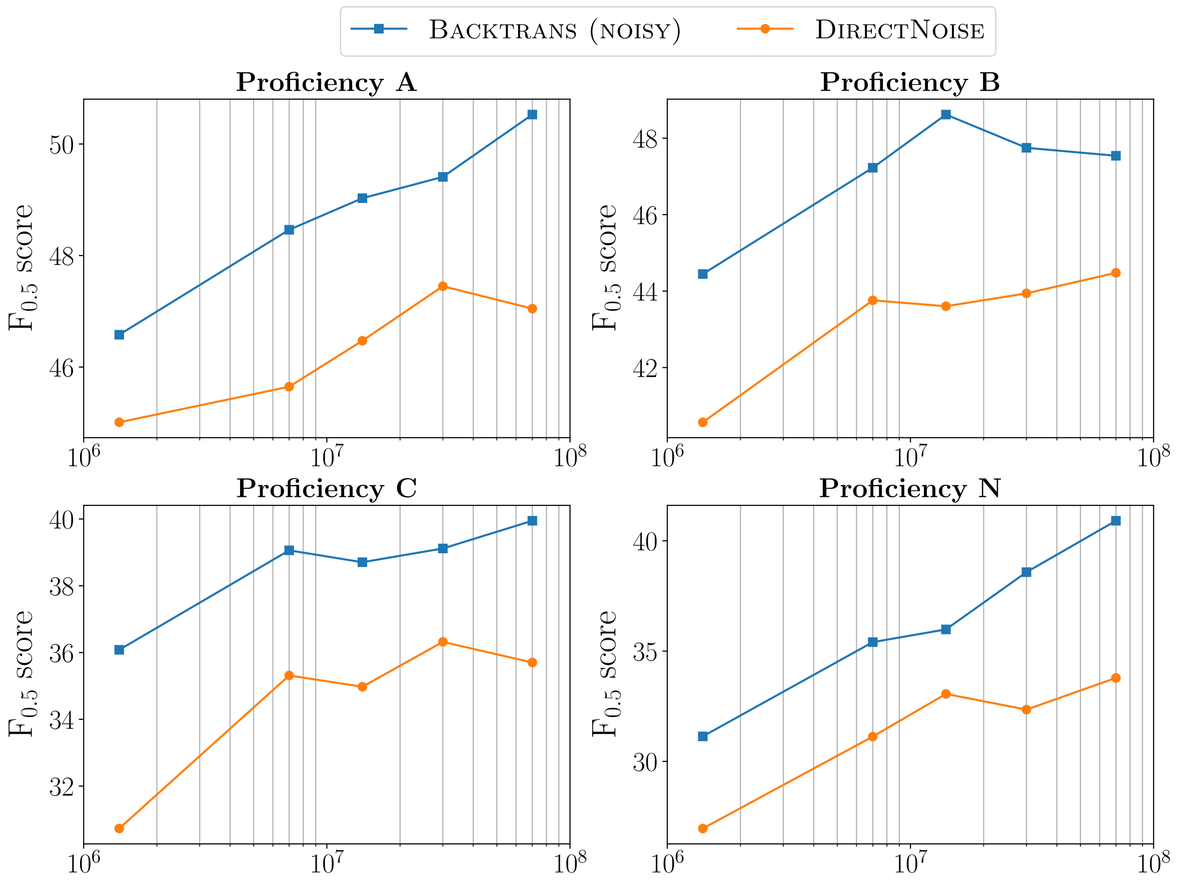


Figure 4.10 Performance of the model on each proficiency level in BEA-valid.

knowledge into the model. For the latter error, the source sentence does not contain enough information for the model to make the correction. This is because "we" in the source sentence cannot be disambiguated: if "we" is the audience, "signature" should be corrected to "autograph." However, supposing that "we" refers to the people from the record label, "signature" seems appropriate (e.g., making a contract). Developing a cross-sentence GEC model (Chollampatt et al., 2019) is a promising approach to overcome such difficulty of disambiguation.

Proficiency-wise Analysis

We investigated the effect of increasing the amount of pseudo data with respect to each proficiency level. If the performance of a certain proficiency level saturated, then an approach other than increasing the amount of pseudo data would be necessary to improve the performance.

The results are presented in Figure 4.10. Here, the performances for proficiency levels A and N scale to the amount of pseudo data, whereas B and C appear to saturate. As discussed in Section 4.5.1, different seed corpora are appropriate for different proficiency levels. Thus, we may be able to improve the performance of B and C by incorporating a seed corpus with lower grammatical complexity. However, the size of SimpleWiki, which contains only approximately 1.4M sentences, is insufficient for this purpose; extracting text from a raw corpus (e.g., Common Crawl[14]) is thus critical.

4.6 Related Work

4.6.1 Methods for Generating Pseudo Data

The lack of genuine data $\mathscr{D}_g$ (i.e., manually annotated error tagged data) has been an ongoing challenge in the field of GEC. The generation of pseudo data has been a central approach for mitigating this problem. There are generally two methods, which can be divided into the following: (i) a rule/probability-based method and (ii) an MT-based method.

A common rule/probability-based method involves applying error templates to external grammatical sentences (i.e., a seed corpus). Here, the templates are generated from a small amount of genuine data. One of the advantages of this method is that a template can be easily manipulated; for example, it can be designed to focus on specific error types, such as mass noun (Brockett et al., 2006), article (Izumi et al., 2003; Rozovskaya and Roth, 2010b; Rozovskaya et al., 2012), and preposition (Cahill et al., 2013; Rozovskaya and Roth, 2010a; Rozovskaya et al., 2012), or it may support grammatical errors in general (Choe et al., 2019; Felice and Yuan, 2014; Yuan and Felice, 2013). More recently, Grundkiewicz et al. (2019) proposed a similar method; they probabilistically replaced words with ones in confusion sets created by an off-the-shelf English spell checker. Their model achieved the best performance in the BEA 2019 shared task (Bryant et al., 2019).

The another rule/probability-based approach is the one proposed by Zhao et al. (2019), which is to inject synthetic noise into grammatical sentences. This is heavily inspired by the concept of both denoizing auto-encoders (Vincent et al., 2008) and pretraining gigantic language models (Devlin et al., 2019; Grundkiewicz et al., 2019; Song et al., 2019; Zhao et al., 2019). We conducted an experiment using a variant of this method (DirectNoise).

A major disadvantage associated with the rule/probability-based is that it is difficult to obtain high-quality data that closely resembles the errors present in genuine data such as the naturally occurring grammatical error. Several studies have reported that pseudo data may

[14]https://commoncrawl.org/

cause performance degradation (Felice and Yuan, 2014; Foster and Andersen, 2009; Yuan and Felice, 2013). Recently, the MT-based method, where a model is used to generate an error from a grammatical sentence, has drawn the attention of the research community. Here, the model is typically either statistical MT (Rei et al., 2017) or neural MT (Kasewa et al., 2018; Lichtarge et al., 2019; Xie et al., 2018). For example, Lichtarge et al. (2019) proposed a round-trip translation method. This method regards translation errors generated by the MT model as pseudo grammatical errors. Here, the method is to first translate a given grammatical sentence to an arbitrary bridge language (e.g., Japanese). Then, a sentence is translated back to the original language and used as a source sentence of pseudo data. Another MT-based method is a variant of the back-translation method proposed by Xie et al. (2018), i.e., BACKTRANS (NOISY) (Section 4.3.1). They demonstrated that BACKTRANS (NOISY) can generate sentences that cannot be distinguished from genuine text containing a grammatical error. In the experiment, we compared BACKTRANS (NOISY) with DIRECTNOISE and demonstrated that the former exhibits superior performance.

Apart from *generating* pseudo data, there exists an approach to *crawl* pseudo data from the Web. Specifically, Lichtarge et al. (2019) extracted Wikipedia's revision histories and constructed pseudo data, namely, Wikipedia Revisions. This approach is orthogonal to ours; one may jointly use generated pseudo data and Wikipedia Revisions.

4.6.2 Seed Corpus

Historically, numerous types of corpora, e.g., collection of news articles (Brockett et al., 2006; Grundkiewicz et al., 2019; Rozovskaya and Roth, 2010a; Xie et al., 2018; Zhao et al., 2019), British National Corpus (Foster and Andersen, 2009), and English Wikipedia (Cahill et al., 2013; Felice and Yuan, 2014; Rozovskaya and Roth, 2010a,b), have been considered to be the seed corpus for generating pseudo data. This fact implies that consensus has not yet been achieved with respect to the choice of seed corpus. Our research question on the effectiveness of seed corpus on the model performance has been formulated based on such a situation.

Felice and Yuan (2014) has the motivation similar to our study; the authors denoted that the variables of seed corpus, including the (i) topic (ii) genre (iii) style (iv) text complexity, and (v) native language of the writer, should be considered. However, they only considered English Wikipedia as the seed corpus; and thus, the no sufficient conclusion could be obtained with respect to these variables. In our study, we conducted controlled experiments and compared three distinctive seed corpora (Section 4.4.3). We found that Gigaword is the best seed corpus, at least in our setting.

4.6.3 Optimization Settings

As discussed in Section 4.2, there are at least two means of optimization, i.e., JOINT and PRETRAIN. The effectiveness of JOINT optimization has been observed in both non-neural models (Felice and Yuan, 2014; Yuan and Felice, 2013) and neural models (Xie et al., 2018). However, the PRETRAIN approach is being actively explored in the GEC field.

The origin of the PRETRAIN approach with respect to the GEC model is “partial pre-training” of EncDec. Junczys-Dowmunt et al. (2018) initialized the embedding matrix and decoder parameters of EncDec with Word2Vec vectors (Mikolov et al., 2013) and pretrained language model respectively. Further, they reported that both procedures consistently improved the performance of EncDec. A similar approach has also been used by Chollampatt et al. (2019). However, more recently, “full pretraining” of EncDec has become dominant (Choe et al., 2019; Grundkiewicz et al., 2019; Lichtarge et al., 2019; Zhao et al., 2019), owing to its strong empirical performance. This approach pretrains the entire EncDec using pseudo data, and subsequently fine-tunes it with genuine data.

One interesting aspect of GEC is that even though it incorporates same model (EncDec) as MT, the dominant optimization setting is different. The JOINT setting is commonly used in MT (Caswell et al., 2019; Edunov et al., 2018), whereas PRETRAIN is used in the existing GEC studies. We compared both settings and confirmed the superiority of PRETRAIN (Section 4.4.4).

4.7 Conclusions

In this chapter, we investigated several aspects of the incorporation of pseudo data in GEC. By conducting a massive amount of experiments, the following procedures are concluded to be effective for training the model and for obtaining a strong performance:

1. utilize Gigaword as the seed corpus (Section 4.4.3);
2. pretrain the model with large amount (e.g., 70M) of BACKTRANS (NOISY) data (Section 4.4.4); and
3. fine-tune the pretrained model using genuine data (Section 4.4.4).

Further, we demonstrated the effectiveness of this proposal by achieving state-of-the-art performance using the CoNLL-2014 and BEA-2019 test sets (Section 4.4.5).

Subsequently, we conducted an in-depth analysis of the experimental results; our findings can be summarized as follows.

1. The suitable seed corpus varies across various grammatical error types and proficiency levels: for example, a seed corpus exhibiting low grammatical complexity is suitable for low-proficiency texts (Section 4.5.1, Section 4.5.1).

2. When compared with the baseline, our proposed setting (PRETLARGE) improves the performance with respect to all the grammatical error types. However, the content word errors remain a challenge (Section 4.5.2).

3. The performance at each proficiency level shows promising scalability with respect to the amount of pseudo data. However, the performances at proficiency levels B and C seem to saturate when using the largest data (Section 4.5.2).

Based on these observations, several possible approaches can be suggested to further improve model performance, especially on content word errors. For example, developing (1) a pseudo data generation method that can exclusively generate content word errors, (2) a means of injecting commonsense knowledge into the model, and (3) more sophisticated cross-sentence GEC are promising approaches.

Chapter 5

The Role of Semi-supervised Learning in the State-of-the-Art Machine Translation System

5.1 Introduction

The joint team of Tohoku University, RIKEN AIP, and NTT (Tohoku-AIP-NTT) participated in the WMT'20 shared news translation task (Barrault et al., 2020) in two language pairs and four language directions: English→German (En→De), German→English (De→En), English→Japanese (En→Ja), and Japanese→English (Ja→En).

At the very beginning of this year's shared task, we planned to employ the following two enhancements at the core of our system. The first enhancement is the noisy synthetic data filtering (Koehn et al., 2018) to better utilize the millions of back-translated synthetic data. However, as we analyze in Section 5.5.1, this filtering turned out to be ineffective. The second enhancement is the reranking of n-best candidates generated a the model. Given a collection of scores from multiple generative/translation models, our reranking module selects the best candidate. We attempted to develop sophisticated machine learning based methods for optimizing the weight of each score. However, we found that those methods are not as effective as the simple grid search on the BLEU score (details in Section 5.3.7 and Section 5.5.3).

Eventually, we designed our system as a combination of techniques that are already widely adopted in the shared task, such as back-translation and fine-tuning. The overview of our system is shown in Figure 5.1. We achieved the first place in De→En on automatic evaluation and obtained strong results in other language directions.

5.2 Dataset and Preprocessing

5.2.1 Bitext

For both En↔De and En↔Ja, we used all bitexts that are available for a constrained system.
En↔De Following Ng et al. (2019), we applied language identification filtering (`langid`)[1] to the bitext. In this filtering, sentence pairs were removed if a supposedly English/German sentence is identified as a non-English/German sentence. Then, we applied the `clean-corpus-n` script available in the Moses toolkit (Koehn et al., 2007) and removed sentence pairs that are either too long and/or their length ratio is too large[2]. These two filtering processes provided us with approximately 44M sentence pairs. Then, we trained and applied the Moses `truecaser` independently for each language. We also trained byte-pair encoding (BPE) (Sennrich et al., 2016c) models using the `sentencepiece` (Kudo and Richardson, 2018) implementation. For BPE training, we used only a subset of the parallel corpus (Europarl, NewsCommentary, and RAPID) to prevent extremely rare characters from contaminating the vocabulary and the subword segmentation.
En↔Ja Similar to En↔De, we applied `langid` to clean bitext, but we did not use `clean-corpus-n` since the Japanese text is not segmented. Instead, we simply removed sentence pairs in which the English sentence is longer than 500 tokens. Eventually, we obtained about 17M sentence pairs. We used `truecaser` for the English side only, because case information does not exist in the Japanese language. We independently trained the BPE merge operation on the bitext. We set the character coverage option[3] of `sentencepiece` to 1.0 and 0.9998 for English and Japanese, respectively.

5.2.2 Monolingual Corpus

The origins of the monolingual corpus in our system are the Europarl, NewsCommentary, and entire NewsCrawl (2008-2019) corpora for English and German, and the Europarl, NewsCommentary and CommonCrawl corpora for Japanese. After bitext preprocessing (Section 5.2.1), we applied `langid` filtering to all monolingual corpora. These corpora are used for large-scale back-translation (Section 5.3.3).

[1]https://github.com/saffsd/langid.py
[2]We set the minimum length to 1, the maximum length to 250, and the maximum ratio to 3.0.
[3]`--character_coverage`

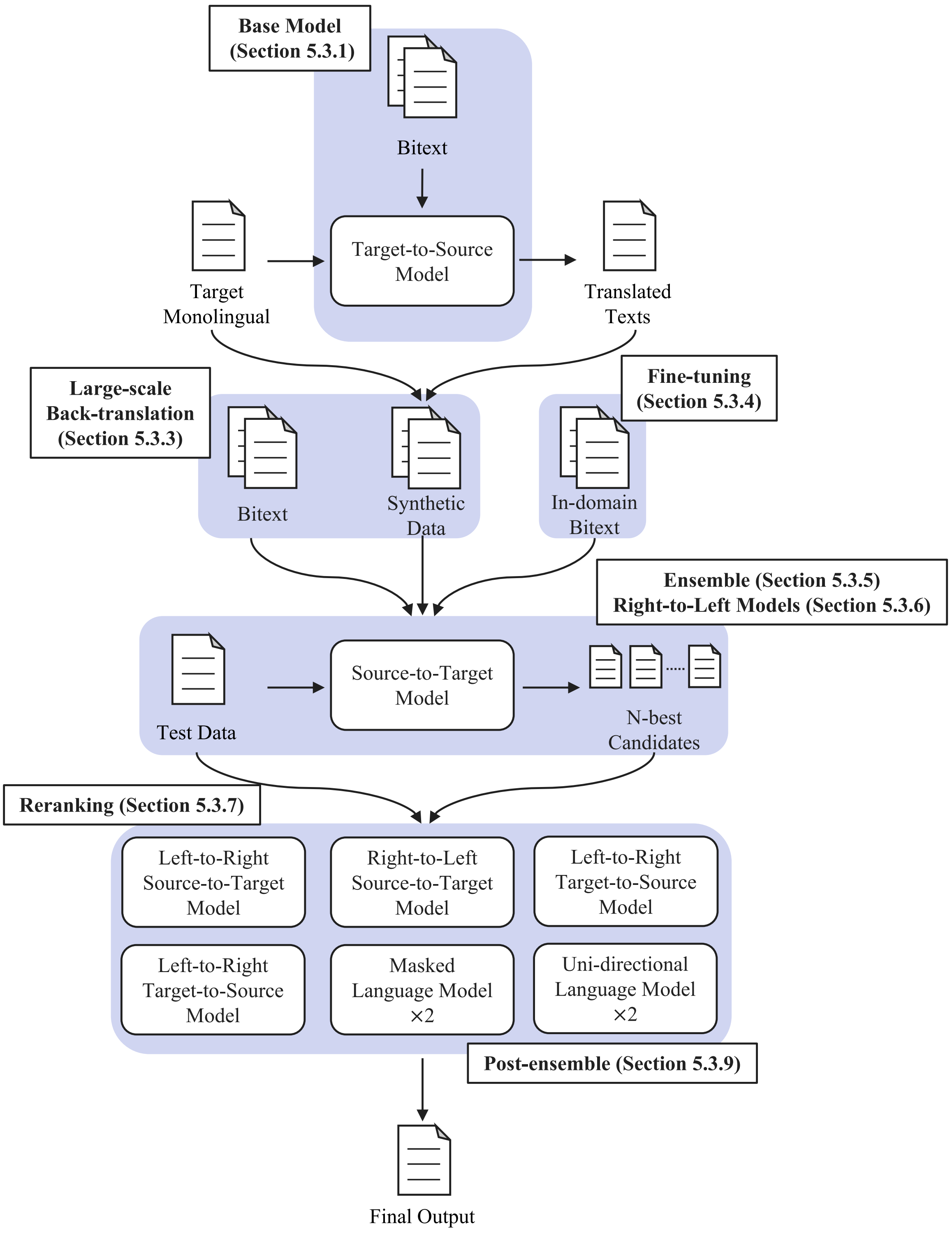

Base Model (Section 5.3.1)
Bitext
Target Monolingual
Target-to-Source Model
Translated Texts
Large-scale Back-translation (Section 5.3.3)
Bitext
Synthetic Data
Fine-tuning (Section 5.3.4)
In-domain Bitext
Ensemble (Section 5.3.5)
Right-to-Left Models (Section 5.3.6)
Test Data
Source-to-Target Model
N-best Candidates
Reranking (Section 5.3.7)
Left-to-Right Source-to-Target Model
Right-to-Left Source-to-Target Model
Left-to-Right Target-to-Source Model
Left-to-Right Target-to-Source Model
Masked Language Model ×2
Uni-directional Language Model ×2
Post-ensemble (Section 5.3.9)
Final Output

Figure 5.1 Overview of our system.

5.3 System Overview

5.3.1 Base Model and Hyperparameter

The well-known Transformer model (Vaswani et al., 2017) is our base Encoder Decoder model. Specifically, we started with the "Transformer (big)" setting described by Vaswani et al. (2017) and increased the feed-forward network (FFN) size from 4,096 to 8,192. Ng et al. (2019) reported that this larger FFN setting slightly improves the performance; we also confirmed it in our preliminary experiment.

Table 5.1 shows a list of hyperparameters for model optimization. We employed an extremely large mini-batch size of 512,000 tokens using the delaying gradient update technique (Bogoychev et al., 2018; Ott et al., 2018). This is because previous studies showed that a large mini-batch size leads to a faster convergence (Ott et al., 2018) and a better generalization (Bawden et al., 2019; Morishita et al., 2019; Popel and Bojar, 2018). We also used a large learning rate of 0.001 to further accelerate the convergence (Goyal et al., 2017; Liu et al., 2019; Ott et al., 2018). We use the `fairseq` toolkit (Ott et al., 2019) for the entire set of experiments. Every reported BLEU score is measured using `SacreBLEU` (Post, 2018).

5.3.2 Subword Size

For En↔De, we used the subword size of 32,000, which is commonly used in previous studies Ng et al. (2019); Vaswani et al. (2017). For En↔Ja, we conducted a hyperparameter search for a suitable subword size; Morishita et al. (2019) empirically showed that a small subword size (e.g., 4,000) is superior to those commonly adopted in the literature (e.g., 16,000 and 32,000). Given their findings, we searched for the subword size in the following range: {4000, 8000, 16000, 32000}.

Table 5.2 shows that the largest subword size achieves the best performance, which is inconsistent with the result of Morishita et al. (2019). One explanation for this result is that Morishita et al. (2019) conducted an experiment on the ASPEC corpus, whose size (approx. 3M) is much smaller than that of the bitext available for the En↔Ja task. That is, the bitext available for the En↔Ja task is sufficiently large for the model to learn a meaningful representation for each subword unit that is close to the word level. Thus, we also used the subword size of 32,000 for En↔Ja.

5.3.3 Large-scale Back-translation

We used the back-translation technique Sennrich et al. (2016b) to generate large-scale synthetic data. First, we trained models on the bitext for all language pairs. Second, for each

Table 5.1 List of hyperparameters for each model.

Base Model	
Architecture	Transformer (big) with FFN size of 8,192
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$)
Learning Rate Schedule	Inverse square root decay
Warmup Steps	4,000
Max Learning Rate	0.001
Dropout	0.3
Gradient Clipping	1.0
Label Smoothing	$\epsilon_{ls} = 0.1$ (Szegedy et al., 2016)
Mini-batch Size	512,000 tokens
Number of Updates	40,000 steps for En↔De and 80,000 steps for En↔Ja
Averaging	Save checkpoint for every 2,000 steps and take an average of last 10 checkpoints
Uni-directional Language Model	
Architecture	`transformer_lm_big` setting available in `fairseq`
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$)
Learning Rate Schedule	Inverse square root decay
Warmup Steps	4,000
Max Learning Rate	0.0005
Dropout	0.1
Gradient Clipping	1.0
Weight Decay	0.0
Mini-batch Size	512,000 tokens
Number of Updates	50,000 steps
Masked Language Model	
Architecture	RoBERTa-base (Liu et al., 2019)
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$)
Learning Rate Schedule	Polynomial decay
Warmup Steps	10,000
Max Learning Rate	0.0005
Dropout	0.1
Gradient Clipping	1.0
Weight Decay	0.01
Mini-batch Size	2,048 sentences
Number of Updates	125,000 steps

language, we fed the monolingual corpus (Section 5.2.2) to the model. Here, we used the beam search of width 6 and length penalty of 1.0. Finally, we applied length and ratio filter-

Table 5.2 Effectiveness of different subword sizes on the validation set of En↔Ja task.

Subword Size	En→Ja
4,000	19.2
8,000	19.6
16,000	19.4
32,000	19.7

Table 5.3 Number of sentence pairs in the synthetic data of each language pair

	En→De	De→En	En→Ja	Ja→En
No filtering	336M	236M	1777M	236M
After filtering	328M	230M	235M	230M

ing to the model outputs[4]. The size of the synthetic data that we generated for each language direction is shown in Table 5.3. The size of the synthetic data for En→Ja, which is generated from CommonCrawl, is extremely large. Thus, we randomly subsampled the synthetic data of En→Ja so that its size roughly matches those of De→En and Ja→En.

We searched for an effective setting for incorporating the synthetic data. As the most straightforward starting point, we simply combined bitext and synthetic data and trained the model. Here, we upsampled the bitext so that the model sees the bitext and synthetic data at a 1:1 ratio (Ng et al., 2019). Table 5.4 shows the result. Here, naively using the synthetic data (BASE+BT) decreased the performance of the model trained with the bitext only (BASE). Given this result, we considered the following two enhancements:

Tagged Back-translation We used the tagged back-translation technique (Tagged-BT) (Caswell et al., 2019), which prepends a special tag token (e.g., ⟨BT⟩) to the source sentence of synthetic data. This simple technique can inform the model about the origin of the given training data, i.e., whether the sentence pair is back-translated. Marie et al. (2020) empirically demonstrated that the model trained with such tagged data can avoid overfitting to the synthetic data. In Table 5.4, the Tagged-BT (BASE+TAGGED-BT) successfully improves the performance from BASE except for the newstest2019. We suspect that the performance does not improve on newstest2019 because it does not contain the "translationese" text, i.e., human-generated translations, which are reported to be the main source of improvement of back-translation (Bogoychev and Sennrich, 2019; Marie et al., 2020).

[4]For En↔De, we removed sentence pairs that contain sentences longer than 250 tokens. For En↔Ja, we removed sentence pairs such that the English sentence is longer than 250 tokens, or the Japanese sentence is longer than 500 characters.

Table 5.4 Effectiveness of using the synthetic data on En→De

Setting	newstest 2014	newstest 2018	newstest 2019
BASE	32.2	47.3	42.2
BASE+BT	32.1	45.9	38.8
BASE+TAGGED-BT	33.0	48.0	42.0
BASE ($l = 9$)+TAGGED-BT	33.1	49.6	42.7
BASE ($l = 12$)+TAGGED-BT	33.4	49.4	42.3

Deeper Model We also considered increasing the model size to take advantage of a massive amount of training data. Specifically, we increased the number of layers l from 6 to 9 and 12 (Wang et al., 2019b). Table 5.4 shows that the performances of BASE ($l = 9$)+TAGGED-BT and BASE ($l = 12$)+TAGGED-BT are almost comparable. We determined that BASE ($l = 9$)+TAGGED-BT is the best option by considering the model performance and training efficiency regarding the GPU memory constraints.

5.3.4 Fine-tuning

Fine-tuning the model with an in-domain news corpus is acknowledged as an extremely important technique for boosting the performance (Bawden et al., 2019; Junczys-Dowmunt, 2019; Ng et al., 2019; Sennrich et al., 2016b). We fine-tuned our models as follows:
En↔De For En↔De, we fine-tuned the model with a collection of newstest2008-2018 and evaluated its performance on newstest2019. For En→De, we only used sentence pairs whose source sentence is originally written in English, i.e., we never used texts with translationese on the source side for fine-tuning. Similarly, for De→En, we used sentence pairs whose source sentence is originally written in German. This way, we ensured that our model does not overfit to the translationese texts; since newstest2019 does not contain translationese texts (Barrault et al., 2019), we expected that newstest2020 does not contain translationese either.

We fine-tuned the model for 200 iterations with a mini-batch size of 20,000 tokens. During the fine-tuning, we fixed the learning rate to 1e-06 for De→En and 1e-05 for En→De. We saved the model every 20 iterations and took an average of the last eight saved models for decoding.
En↔Ja For fine-tuning, we used the Kyoto Free Translation Task (KFTT) corpus and NewsCommentary as the *clean* bitext and NewsCommentary as the *news* bitext. We fine-tuned the models by a two-step procedure, that is, we first fine-tuned with the *clean* bitext

for 2,000 steps. Then we fine-tuned with the *news* bitext for 200 steps. We found that the validation performance of this two-step procedure is slightly better than that of the fine-tuning with the *news* bitext only.

5.3.5 Ensemble

We used the model ensemble method to improve the performance. First, we trained four models with different random seeds. These models were then simultaneously used for computing the score of each candidate during the beam search decoding.

5.3.6 Right-to-Left Models

We used Right-to-Left (R2L) models for reranking the n-best candidates from Left-to-Right (L2R) models. R2L models generate sentences in reverse order. Suppose that conventional L2R models generate sentences from the beginning-of-the-sentence (BOS) to the end-of-the-sentence (EOS); R2L models generate from EOS to BOS. This reranking technique was independently proposed by Liu et al. (2016) and Sennrich et al. (2016a) to mitigate the search error of L2R models, which may occur around EOS. We trained four R2L models and used their scores for reranking the n-best candidates generated by L2R models (Section 5.3.5). Specifically, we computed the score of each candidate with both L2R models and R2L models. Then, we took the sum of the two scores and obtained the final score. We sorted this final score and then selected the candidate with the highest score.

5.3.7 Reranking

We also applied a reranking method based on the scores of several translation (or generative) models, which is closely related to one iteration of Minimum Error Rate Training (MERT) (Och, 2003) often used in Statistical Machine Translation (SMT). The underlying idea is to find the balance of likelihood independently computed from the models.

Suppose we have a set of candidate output sentences for each input in either the validation (training phase) or the test (evaluation phase) sets. In our case, we independently generated n-best candidates using the L2R and R2L models, and obtained $2n$ candidates in total for each. Here, let $\mathscr{C}_i$ represent the set of the obtained $2n$ candidates of the i-th input.

Next, $P_j(e) \in [0, 1]$ denotes the score of the candidate $e \in \mathscr{C}_i$ obtained from the j-th model, where $j \in \{1, \dots, J\}$. Let $w_j \in [0, 1]$ be a weighting factor of the j-th model, and $\mathbf{w} = (w_1, \dots, w_J)$ be the vector representation of the weighting factor. We then obtained the most likely candidate $\hat{e}_{i,\mathbf{w}}$ from $\mathscr{C}_i$ given the i-th input and $\mathbf{w}$ as follows:

$$\hat{e}_{i,\mathbf{w}} = \underset{e \in \mathscr{C}_i}{\arg\max} \left\{ \sum_{j=1}^{J} w_j \log(P_j(e)) \right\}. \tag{5.1}$$

Finally, for the parameter estimation of $\mathbf{w}$, we explored $\widehat{\mathbf{w}}$ by using the following optimization problem:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w} \in \mathscr{G}_{\mathbf{w}}}{\arg\max} \left\{ \texttt{SacreBLEU}(\widehat{\mathscr{E}}_{\mathbf{w}}) \right\}, \tag{5.2}$$

where $\widehat{\mathscr{E}}_{\mathbf{w}} = (\hat{e}_{i,\mathbf{w}})_{i=1}^{I}$ and $\mathscr{G}_{\mathbf{w}}$ represent a set of values that w_j can take, namely, $[0, 1]^J$.

For the reranking experiment, we prepared the following generative and translation models to compute $P_j(e)$.

Source-to-Target L2R and R2L Model The Source-to-Target L2R and R2L models are the same as that used for the candidate generation; the ensemble of four L2R models and four R2L models compute the score of each candidate.

Target-to-Source L2R and R2L Model The Target-to-Source (T2S) model translates a sequence in a reverse direction, that is, it translates a given target sequence to a source sequence. For example, if a candidate sentence is generated by the En→De model, we use the De→En model for computing the T2S score.

Uni-directional Language Model We used the uni-directional language model (UniLM) to compute the likelihood of the decoded target sequence. To do this, we trained the Transformer-based language model (Baevski and Auli, 2019) for all languages on monolingual data. We obtained two distinct scores from two normalization methods: (1) simply dividing by the target sequence length (Yee et al., 2019) and (2) *SLOR* (Lau et al., 2020; Pauls and Klein, 2012). A list of hyperparameters is shown in Table 5.1.

Masked Language Model We also used the pre-trained masked language model (MLM) (Devlin et al., 2019) for computing the score. Specifically, we trained the RoBERTa-base (Liu et al., 2019) setting available in `fairseq` on monolingual data. First, we computed the unnormalized log-probabilities by the method described by Wang and Cho (2019). Then, we normalized the probability by (1) dividing by the sequence length and (2) *PenLP* (Lau et al., 2020; Vaswani et al., 2017). A list of hyperparameters is shown in Table 5.1.

Because the uni-directional language model and MLM both have two distinct variations, we used a total of six models, namely, $J = 6$.

5.3.8 Post-processing

We converted the decoded target sequence from a sequence of subwords to tokens. Then we applied the Moses `detruecaser` to English and German sequences. We also applied language-specific post-processing as follows:

En↔De We observed that the rare tokens such as Greek letters in the source sequence are sometimes translated into ⟨`UNK`⟩. We handled ⟨`UNK`⟩ in the decoded sequence by copying the corresponding token from the source sequence. We determined the corresponding token by finding the token that does not exist in one of the source-side or target-side vocabularies.

En→Ja We did not take any special measures for ⟨`UNK`⟩[5]. We replaced the English style comma “，” and period “．” with the Japanese style “、” and “。” respectively.

Ja→En We observed that the model translates the Japanese vertical bar “｜” to ⟨`UNK`⟩. Thus, we replaced all ⟨`UNK`⟩ with “|”.

5.3.9 Post-ensemble

Kobayashi (2018) proposed the method of taking the ensemble of multiple models *after* decoding the sequence, namely, post-ensemble (POSTENSEMBLE). The underlying idea of POSTENSEMBLE is to choose “majority-like” candidates by comparing the similarities among candidates. He applied POSTENSEMBLE to the abstractive summarization task and reported that the performance is superior to that of the conventional ensemble.

We used POSTENSEMBLE in En→Ja[6]. Specifically, we adopted the `PostCosE` variant in which the cosine similarity is used as a similarity metric. We created 300 dim fasttext word vectors (Bojanowski et al., 2017) on the Japanese monolingual corpus.

5.4 Results

Performance on the Validation Set We show the validation performance of our system in Table 5.5. We used newstest2019 and the official validation set for En↔De and En↔Ja, respectively, for the validation data. The table shows the effectiveness of incorporating each technique described in Section 5.3. Each technique consistently improves the performance in most cases. In addition, it is noteworthy that both En→De and De→En models significantly outperform the performance of the best system from last year’s shared task (WMT’19).

[5]In fact, we never observed ⟨`UNK`⟩ in the decoded test set.

[6]Kobayashi (2018) introduced POSTENSEMBLE as the method that *replaces* the conventional ensemble. Instead, we used two ensemble methods simultaneously.

Table 5.5 Effectiveness of each technique: we use newstest2019 and official validation set for En↔De and En↔Ja respectively. The best result from WMT'19 is unavailable for En↔Ja, because this task has newly appeared this year.

ID	Setting	En→De	De→En	En→Ja	Ja→En
(a)	BASE (Section 5.3.1)	42.4	42.0	19.7	21.6
(b)	BASE ($l = 9$)+TAGGED-BT (Section 5.3.3)	42.7	42.5	22.0	23.9
(c)	(b) + fine-tuning (Section 5.3.4)	44.9	42.3	23.1	24.4
(d)	(c) × 4 (Section 5.3.5)	45.5	42.8	23.9	25.4
(e)	(d) + 4 × (c)-R2L (Section 5.3.6)	45.4	43.6	24.2	25.9
(f)	(e) + reranking (Section 5.3.7)	45.7	43.8	24.9	26.2
-	The best system in WMT'19	44.9	42.8	-	-

Table 5.6 Performance on WMT'20 Test Set: refer to Table 5.5 for model ID.

Direction	Setting / ID	BLEU	chrF
En→De	(f) (Table 5.5)	37.5	0.647
De→En	(f) (Table 5.5)	43.8	0.690
En→Ja	(f) (Table 5.5)	40.1	0.343
Ja→En	POSTENSEMBLE	25.5	0.536

Performance on the Test Set We show the test set performance that we measured in the OCELoT system[7] in Table 5.6. The system provides us with the SacreBLEU score and the chrF score (Popović, 2015).

We used the following models for POSTENSEMBLE of Ja→En: (1) model (f) (Table 5.5), (2) Model (f) with the ensemble of eight models, in which four models are fine-tuned with the *clean* bitext and the other four models are fine-tuned with the *news* bitext, and (3) Model (2) without n-best candidates from the R2L model.

The performance of En→Ja appears significantly better than the validation performance reported in Table 5.5; this is because OCELoT computes the BLEU score with character-level segmentation, whereas we used the MeCab-based word-level segmentation[8]. We also computed the BLEU score with the MeCab-based segmentation for reference and obtained 25.8 points.

[7]https://ocelot.mteval.org/

[8]The use of the MeCab-based segmentation is recommended by SacreBLEU.

5.5 Analysis

In this section, we introduce several negative results from our preliminary experiments. Our attempts include the following: (1) filtering synthetic data, (2) incorporating forward-translation, and (3) developing a more sophisticated reranking method. We also analyzed the issue regarding the use of brackets in the En→Ja task.

5.5.1 Negative Results on Synthetic Data Filtering

We applied corpus filtering to the synthetic data created in Section 5.3.3. The goal of this filtering is to extract and utilize the "clean" subset of synthetic data that may contribute to the model performance. For each of the sentence pairs in the synthetic data, we assigned scores that represent the likelihood of being a sentence pair (Section 5.5.1). Then, we regarded these scores as features for classification; we trained a model classifying clean and noisy sentence pairs (Section 5.5.1). Finally, on the basis of the confidence scores of the classifier, we extracted the presumably clean subset of the synthetic data.

Features

Pointwise HSIC We computed the score for each sentence pair using the pointwise Hilbert–Schmidt independence criterion (PHSIC) (Yokoi et al., 2018), which is a kernel-based co-occurrence measure. Given a set of sentence pairs, PHSIC can assign a high score to a sentence pair that is consistent with the rest of the sentence pairs. To do this, PHSIC utilizes kernel functions and calculates the sentence similarity. Yokoi et al. (2018) applied PHSIC to machine translation corpus filtering and reported promising results. Thus, we also employed PHSIC for corpus synthetic data filtering.

First, we learned the parameters of the PHSIC matrix with a cosine kernel by using all sentence pairs in the bitext, which are represented as sentence embeddings. Then, we used this trained matrix to compute the scores for the synthetic data. We used the following two methods for computing the sentence embeddings: (1) the weighted sum of fasttext vectors (Bojanowski et al., 2017) by smoothed inverse frequency (SIF) weighting (Arora et al., 2017) and (2) the average of final hidden states of the pre-trained MLM. Here, the fasttext vector is the same as the one used for post-ensemble (Section 5.3.9), and MLM is the one from the reranking (Section 5.3.7). The word frequency for SIF weighting is calculated from the monolingual corpus.

Table 5.7 Effectiveness of corpus filtering on En→De.

	newstest		
Amount of Synthetic Data Used: r (%)	2014	2018	2019
100	33.0	48.0	42.0
50	32.9	48.4	42.3
33	33.1	47.9	42.2
25	32.9	48.5	42.4

Table 5.8 Effectiveness of incorporating forward-translation and back-translation on En→De.

	newstest		
Setting	2014	2018	2019
BASE	32.2	47.3	42.2
BASE+TAGGED-BT	33.0	48.0	42.0
BASE+TAGGED-FT	31.7	46.7	42.1
BASE+TAGGED-BT+TAGGED-FT	33.1	48.3	42.4

Cross-entropy from T2S Model We computed the word-normalized conditional cross-entropy using the T2S translation model. For example, the synthetic data generated using the En→De model are scored using the De→En model.

Training a Classifier

We trained a linear support vector machine model that classifies clean and noisy sentence pairs. To train the classifier, we used newstest2009-2019 and the official validation set as clean sentence pairs for En↔De and En↔Ja, respectively. We generated the noisy sentence pairs by randomly adding the noise presented by Wang et al. (2018) to the clean sentence pairs.

After training, we classified each sentence pair in the synthetic data. The confidence score of the classifier was used as an overall score that represents the "cleanness" (i.e., quality) of the sentence pair.

Results

We investigated the effectiveness of the synthetic data filtering. First, we sorted the synthetic data according to the score computed with the classifier (Section 5.5.1). Then, we used the top r% of synthetic data for training.

Table 5.7 shows the results of synthetic data filtering with varying r. We trained the En→De model using the Base+Tagged-BT setting. The results showed that our filtering does not seem to improve the performance over the baseline ($r = 100$). One of the possible reasons for this ineffectiveness is the quality of the sentence embeddings used for PHSIC. That is, the use of fasttext and pre-trained MLM might be inappropriate. Utilizing more powerful sentence encoders such as Sentence-BERT (Reimers and Gurevych, 2019) and Universal Sentence Encoder (Cer et al., 2018) is an interesting option to explore in the future; however, the methods of acquiring such resources in the constrained setting is not trivial.

Input	Only one member of the family, then 15-year-old Cassidy Stay, survived.
Reference	家族の中で、ただ一人、当時 15 歳だったカシディ・ステイさんだけが一命を取り留めた。
Model Output	当時 15 歳のキャシディ・ステイ (Cassidy Stay) だけが生き残った。
Input	Madam Needjan, pledged the association's support to the hospital and called on other associations to emulate the gesture.
Reference	マダム・ニージャンは、協会の当病院への支援を約束し、他の団体もこうした行為に追随するよう呼びかけた。
Model Output	マダム・ニージャン (Madam Needjan) は、協会が病院を支援することを約束し、他の協会にこのジェスチャーを模倣するよう求めた。

Figure 5.2 Error analysis of En→Ja translation.

5.5.2 Effectiveness of Incorporating Forward-Translation

Forward-translation (Burlot and Yvon, 2018) is a technique similar to back-translation; the difference is that while back-translation uses the target-side monolingual data, forward-translation uses the source-side monolingual data to generate synthetic data. Bogoychev and Sennrich (2019) reported that forward-translation is effective for improving the translation of texts that are originally written in the source language (i.e., non-translationese texts).

To determine if we can take the best of the two techniques, namely, forward-translation and back-translation, we combined the synthetic data and trained the model. As described in Section 5.3.3, we prepended a distinct tag to each data source: ⟨FT⟩ and ⟨BT⟩ for data generated by forward-translation and back-translation respectively. Then, we upsampled the bitext, so that the model is fed with the bitext and synthetic data at a 1:0.5:0.5 ratio.

Table 5.8 shows the result. The model incorporating both back-translation and forward-translation (BASE+TAGGED-BT+TAGGED-FT) achieves the best result, however, the improvement was marginal. In addition, the performance of the model with forward-translation only (BASE+TAGGED-FT) was worse than that of the baseline (BASE) in all datasets. Given this result, we only used back-translation and kept the training procedure as simple as possible in our final system.

5.5.3 Negative Result on Reranking

We actually investigated several different types of reranking algorithms other than the standard grid search described in Section 5.3.7. For example, we experiment withed optimizing model weights by machine learning based methods such as those using support vector machines, XGBoost (Chen and Guestrin, 2016), and deep neural networks. Unfortunately, none of them worked well. In this competition, we only used the model scores for the reranking. This setting immediately leads the overfitting to the development sets, and hard to extract meaningful generalized weights (rules) that also work well for unseen test data. The development of the methods that can further and consistently improve the quality of translations is our future work for the next year.

5.5.4 Japanese Text and Brackets

Figure 5.2 shows examples from the validation set of the En→Ja task. These examples illustrate the weakness of our model, in which the named entities are often inappropriately translated. According to the references in the figure, the named entities must be translated from alphabetical characters to *katakana* (カタカナ), e.g., *Cassidy Stay* to カシディ・ステ

イ. Although our model successfully translates the named entities in most of the cases, the model also copies original alphabetical characters into the brackets. For example, the model translates *Madam Needjan* to マダム・ニージャン *(Madam Needjan)*. These alphabetical characters damage the BLEU score. We can remove the extra brackets by the rule-based post-processing; however, we find that this naive operation hurts the brevity penalty.

This extra bracket problem seems to reflect the way that the named entities are written in the En↔Ja training data such as KFTT. We should have considered special preprocessing measures in advance to alleviate this problem.

5.6 Conclusion

In this chapter, we described the submission of the joint team of Tohoku, AIP, and NTT to the WMT'20 news translation task. We participated in the En↔De and En↔Ja translation. In preliminary experiments, we attempted new techniques such as synthetic data filtering, forward-translation, and sophisticated reranking. However, none of them was effective. In the submission, we used several standard techniques such as back-translation and fine-tuning. As a result, we achieved the best BLEU score on De→En and strong results in other directions.

Chapter 6

Shifted Absolute Position Embedding for Transformers

6.1 Introduction

Position representation plays a critical role in self-attention-based encoder-decoder models (Transformers) (Vaswani et al., 2017), enabling the self-attention to recognize the order of input sequences. Position representations have two categories (Dufter et al., 2021): absolute position embedding (APE) (Gehring et al., 2017; Vaswani et al., 2017) and relative position embedding (RPE) (Shaw et al., 2018). With APE, each position is represented by a unique embedding, which is added to inputs. RPE represents the position based on the relative distance between two tokens in the self-attention mechanism.

RPE outperforms APE on sequence-to-sequence tasks Narang et al. (2021); Neishi and Yoshinaga (2019) due to *extrapolation*, i.e., the ability to generalize to sequences that are longer than those observed during training (Newman et al., 2020). Wang et al. (2021) reported that one of the key properties contributing to RPE's superior performance is *shift invariance*[1], the property of a function to not change its output even if its input is shifted. However, unlike APE, RPE's formulation strongly depends on the self-attention mechanism. This motivated us to explore a way to incorporate the benefit of shift invariance in APE.

A promising approach to achieving shift invariance while using absolute positions is to randomly shift positions during training. A similar idea can be seen in several contexts, e.g., computer vision (Goodfellow et al., 2016) and question-answering in NLP (Geva et al., 2020). APE is no exception; a random shift should force Transformer to capture the relative

[1] *Shift invariance* is also known as *translation invariance*.

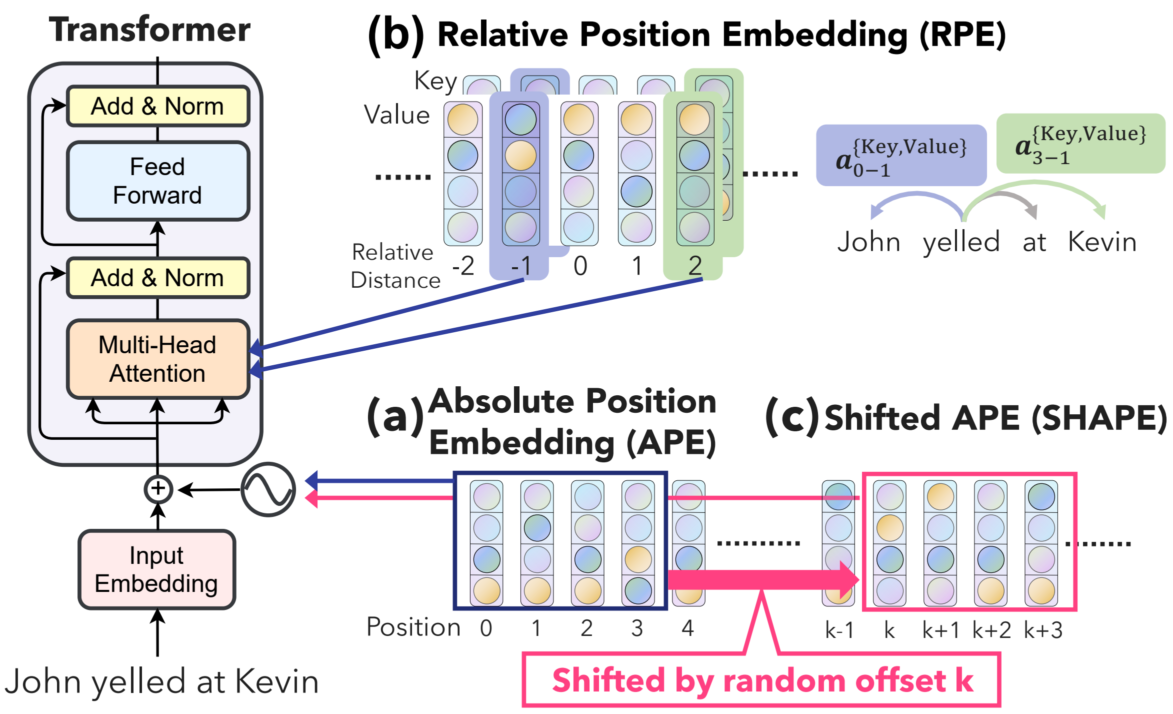


Figure 6.1 Overview of position representations. (a) APE and (c) SHAPE consider *absolute* positions in the *input layer*, whereas (b) RPE considers the *relative* position of a given token pair in the *self-attention mechanism*.

positional information from absolute positions. However, the effectiveness of a random shift for incorporating shift invariance in APE is yet to be demonstrated. Thus, we formulate APE with a random shift as a variant of position representation, namely, *Shifted Absolute Position Embedding* (*SHAPE*; Figure 6.1c), and conduct a thorough investigation. In our experiments, we first confirm that Transformer with SHAPE learns to be shift-invariant. We then demonstrate that SHAPE achieves a performance comparable to RPE in machine translation. Finally, we reveal that Transformer equipped with shift invariance shows not only better extrapolation ability but also better *interpolation* ability, i.e., it can better predict rare words at positions observed during the training.

6.2 Position Representations

Figure 6.1 gives an overview of the position representations compared in this chapter. We denote a source sequence $\boldsymbol{X}$ as a sequence of I tokens, namely, $\boldsymbol{X} = (x_1, \dots, x_I)$. Similarly, let $\boldsymbol{Y}$ represent a target sequence of J tokens $\boldsymbol{Y} = (y_1, \dots, y_J)$.

6.2.1 Absolute Position Embedding (APE)

APE provides each position with a unique embedding (Figure 6.1a). Transformer with APE computes the input representation as the sum of the word embedding and the position embedding for each token $x_i \in \boldsymbol{X}$ and $y_j \in \boldsymbol{Y}$.

Sinusoidal positional encoding (Vaswani et al., 2017) is a deterministic function of the position and the *de facto* standard APE for Transformer[2]. Specifically, for the i-th token, the m-th element of position embedding $\mathrm{PE}(i, m)$ is defined as

$$\mathrm{PE}(i, m) = \begin{cases} \sin\left(\frac{i}{10000^{\frac{2m}{D}}}\right) & m \text{ is even} \\ \cos\left(\frac{i}{10000^{\frac{2m}{D}}}\right) & m \text{ is odd} \end{cases}, \tag{6.1}$$

where D denotes the model dimension.

6.2.2 Relative Position Embedding (RPE)

RPE (Shaw et al., 2018) incorporates position information by considering the relative distance between two tokens in the self-attention mechanism (Figure 6.1b). For example, Shaw et al. (2018) represent the relative distance between the i-th and j-th tokens with relative position embeddings $\mathbf{a}^{\text{Key}}_{i-j}, \mathbf{a}^{\text{Value}}_{i-j} \in \mathbb{R}^D$. These embeddings are then added to key and value representations, respectively.

RPE outperforms APE on out-of-distribution data in terms of sequence length owing to its innate *shift invariance* (Narang et al., 2021; Neishi and Yoshinaga, 2019; Rosendahl et al., 2019; Wang et al., 2021). However, the self-attention mechanism of RPE involves more computation than that of APE[3]. In addition, more importantly, RPE requires the modification of the architecture, while APE does not. Specifically, RPE strongly depends on the self-attention mechanism; thus, it is not necessarily compatible with studies that attempt to replace the self-attention with a more lightweight alternative (Choromanski et al., 2021; Kitaev et al., 2020; Tay et al., 2020).

RPE, which was originally proposed by Shaw et al. (2018), has many variants in the literature (Dai et al., 2019; Huang et al., 2020; Raffel et al., 2020; Wang et al., 2021; Wu et al., 2021). They aim to improve the empirical performance or the computational speed compared with the original RPE. However, the original RPE is still a strong method in terms of the

[2]Learned position embedding (Gehring et al., 2017) is yet another variant of APE; however, we exclusively focus on sinusoidal positional encoding as its performance is comparable (Vaswani et al., 2017).

[3]Narang et al. (2021) reported that Transformer with RPE is up to 25% slower than that with APE.

performance. Narang et al. (2021) conducted a thorough comparison on multiple sequence-to-sequence tasks and reported that the performance of the original RPE is comparable to or sometimes better than its variants. Thus, we exclusively use the original RPE in our experiments.

6.2.3 Shifted Absolute Position Embedding (SHAPE)

Given the drawbacks of RPE, we investigate SHAPE (Figure 6.1c) as a way to equip Transformer with shift invariance without any architecture modification or computational overhead on APE. During training, SHAPE shifts every position index of APE by a random offset. This prevents the model from using absolute positions to learn the task and instead encourages the use of relative positions, which we expect to eventually lead to the learning of shift invariance.

Let k represent an offset drawn from a discrete uniform distribution $\mathcal{U}\{0, K\}$ for each sequence and for every iteration during training, where $K \in \mathbb{N}$ is the maximum shift. SHAPE only replaces $\mathrm{PE}(i, m)$ of APE in Equation 6.1 with

$$\mathrm{PE}(i + k, m). \tag{6.2}$$

We independently sample k for the source and target sequence. SHAPE can thus be incorporated into any model using APE with virtually no computational overhead since only the input is modified. Note that SHAPE is equivalent to the original APE if we set $K = 0$; in fact, we set $K = 0$ during inference. Thus, SHAPE can be seen as a natural extension to incorporate shift invariance in APE.

SHAPE can be interpreted in multiple viewpoints. For example, SHAPE can be seen as a regularizer that prevents Transformer from overfitting to the absolute position; such overfitting is undesirable not only for extrapolation (Neishi and Yoshinaga, 2019) but also for APE with length constraints (Oka et al., 2020, 2021; Takase and Okazaki, 2019). In addition, SHAPE can be seen as a data augmentation method because the randomly sampled k shifts each instance into different subspaces during training.

6.3 Experiments

Using machine translation benchmark data, we first confirmed that Transformer trained with SHAPE learns shift invariance (Section 6.3.2). Then, we compared SHAPE with APE and RPE to investigate its effectiveness (Section 6.3.3).

6.3.1 Experimental Configuration

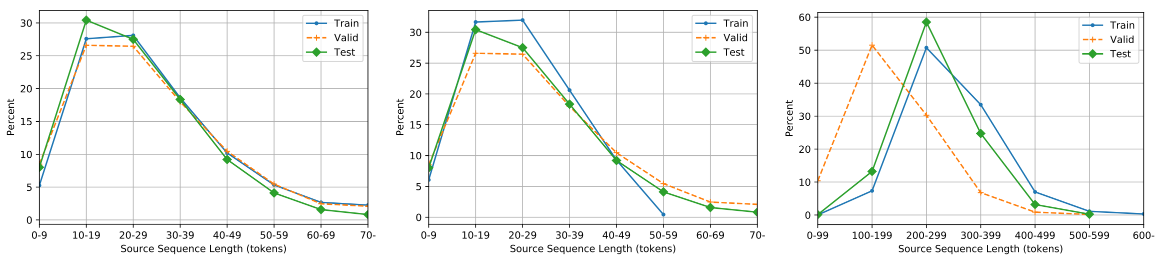
Dataset We used the WMT 2016 English-German dataset for training and followed Ott et al. (2018) for tokenization and subword segmentation (Sennrich et al., 2016c). We used newstest2010-2013 and newstest2014-2016 as the validation and test sets, respectively.

Our experiments consist of the following three distinct dataset settings:
(i) Vanilla: Identical to previous studies (Ott et al., 2018; Vaswani et al., 2017).
(ii) Extrapolate: Shift-invariant models are typically evaluated in terms of extrapolation ability (Newman et al., 2020; Wang et al., 2021). We replicated the settings of Neishi and Yoshinaga (2019); the training set excludes pairs whose source or target sequence exceeds 50 subwords, while the validation and test sets are identical to VANILLA.
(iii) Interpolate: We also evaluate the models from the viewpoint of *interpolation*, which we define as the ability to generate tokens whose lengths are seen during training. Specifically, we evaluate interpolation using long sequences since, first, the generation of long sequences is an important research topic in NLP (Maruf et al., 2021; Zaheer et al., 2020) and second, in datasets with long sequences, the position distribution of each token becomes increasingly sparse. In other words, tokens in the validation and test sets become unlikely to be observed in the training set at corresponding positions; we expect that shift invariance is crucial for addressing such position sparsity.

In this study, we artificially generate a long sequence by simply concatenating independent sentences in parallel corpus. Specifically, given ten neighboring sentences of VANILLA, i.e., $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{10}$ and $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{10}$, we concatenate each sentence with a unique token $\langle sep \rangle$. We also apply the same operation to the validation and test sets.
Evaluation We evaluate the performance with sacreBLEU (Post, 2018). Throughout the experiment, we apply the moses detokenizer to the system output and then compute the *detokenized* BLEU. We summarized the statistics, preprocessing, and evaluation metrics of datasets used in our experiment in Table 6.1. The length statistics are in Figure 6.2.

Table 6.1 Summary of statistics, preprocessing, and evaluation metric of datasets used in our experiment.

Dataset Name	Training Data	# of Sent. Pairs in Training Data	Validation	Test	Evaluation Metric
VANILLA	WMT 2016 English-German	4.5M	newstest2010-2013	newsetst2014-2016	detokenized BLEU via sacre-BLEU
EXTRAPOLATE	WMT 2016 English-German. We removed sequence pairs if the length of the source or target sentence exceeds 50 subwords.	3.9M	newstest2010-2013	newsetst2014-2016	detokenized BLEU via sacre-BLEU
INTERPOLATE	WMT 2016 English-German. Given neighboring ten sentence of VANILLA, i.e., $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{10}$ and $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{10}$, we concatenate each sentence with a special token $\langle sep \rangle$.	450K	newstest2010-2013. We concatenated sentences as in training data.	newstest2014-2016. We concatenated sentences as in training data.	detokenized BLEU via sacre-BLEU



(a) VANILLA dataset (b) EXTRAPOLATE dataset (c) INTERPOLATE dataset

Figure 6.2 Distribution of source sequence length of each dataset.

Models We adopt *transformer-base* (Vaswani et al., 2017) with APE, SHAPE, or RPE, respectively. Our implementations are based on OpenNMT-py (Klein et al., 2017). Unless otherwise stated, we use a fixed value ($K = 500$) for the maximum shift of SHAPE to demonstrate that SHAPE is robust against the choice of K. We set the relative distance limit in RPE to 16 following Shaw et al. (2018) and Neishi and Yoshinaga (2019).

We present the list of hyperparameters used in our experiments in Table 6.2. Hyperparameters for training Transformer follow the recipe available in the official documentation page of OpenNMT-py[4].

[4]https://opennmt.net/OpenNMT-py/FAQ.html#how-do-i-use-the-transformer-model

Table 6.2 List of hyperparameters. †: this corresponds to “learning rate” variable defined in OpenNMT-py framework.

Configurations	**Selected Value**
Encoder-Decoder Architecture	*transformer-base* (Vaswani et al., 2017)
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$)
Learning Rate Schedule	“Noam” scheduler described in (Vaswani et al., 2017)
Warmup Steps	8,000
Learning Rate Scaling Factor†	2
Dropout	0.1
Gradient Clipping	None
Beam Search Width	4
Label Smoothing	$\epsilon_{ls} = 0.1$ (Szegedy et al., 2016)
Mini-batch Size	112k tokens
Number of Gradient Steps	200,000
Averaging	Save checkpoint for every 5,000 steps and take an average of last 10 checkpoints
Maximum Offset K (for SHAPE)	We set $K = 500$ for the most of the experiments. We manually tuned K on validation BLEU for EXTRAPOLATE from following range: $\{10, 20, 30, 40, 100, 500\}$, and report the score of $K = 40$ in addition to $K = 500$. We used a single random seed for the tuning.
Relative Distance Limit (for RPE)	16 following (Neishi and Yoshinaga, 2019)
GPU Hardware Used	DGX-1 and DGX-2

Table 6.3 BLEU score on the sub-sampled *training* data of INTERPOLATE (10,000 pairs). In **Original** and **Swapped**, the order of input sequence is $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{10}$ and $\boldsymbol{X}_2, \ldots, \boldsymbol{X}_{10}, \boldsymbol{X}_1$, respectively.

	Original	**Swapped**	**Performance Drop**
APE	28.81	20.74	8.07
SHAPE	28.51	27.06	1.45

6.3.2 Experiment 1: Shift Invariance

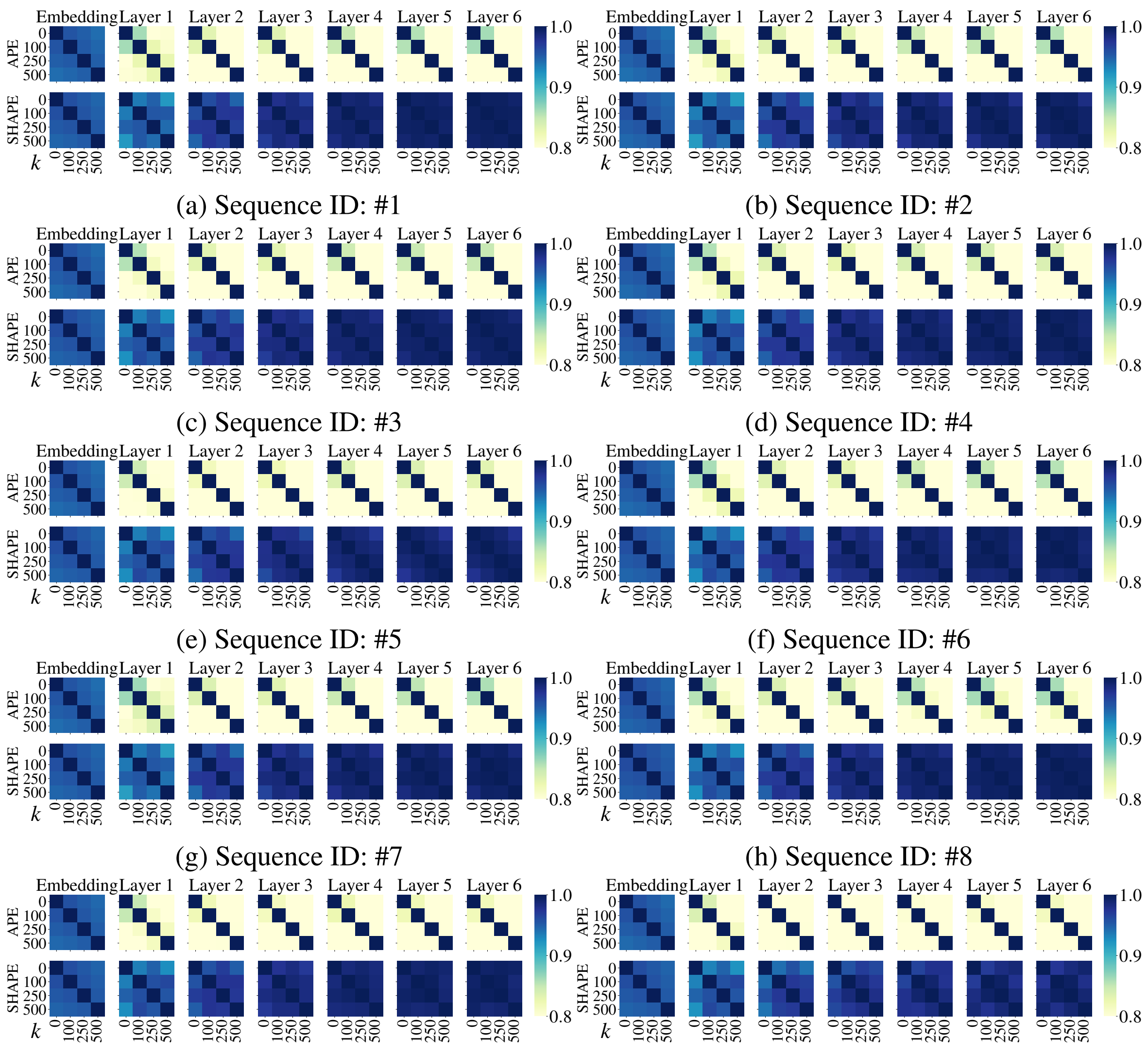
We confirmed that SHAPE learns shift invariance by comparing APE and SHAPE trained on INTERPOLATE.

Quantitative Evaluation: BLEU on Training Data We first evaluated if the model is robust to the order of sentences in each sequence. We used the sub-sampled training data (10k pairs) of INTERPOLATE to eliminate the effect of unseen sentences; in this way, we can isolate the effect of sentence order. Given a sequence in the original order (**Original**), $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{10}$, we generated a *swapped* sequence (**Swapped**) by moving the first sentence to the end, i.e., $\boldsymbol{X}_2, \ldots, \boldsymbol{X}_{10}, \boldsymbol{X}_1$. The model then generates two sequences $\boldsymbol{Y}'_1, \ldots, \boldsymbol{Y}'_{10}$ and $\boldsymbol{Y}'_2, \ldots, \boldsymbol{Y}'_{10}, \boldsymbol{Y}'_1$. Finally, we evaluated the BLEU score of $\boldsymbol{Y}'_1$. The result is shown in Table 6.3. Here, SHAPE has a much smaller performance drop than APE when evaluated on different sentence ordering. This result indicates the shift invariance property of SHAPE.

Qualitative Evaluation: Similarities of Representations We also qualitatively confirmed the shift invariance as shown in Figure 6.3. The figure illustrates how the offset k changes the encoder representations of trained models APE and SHAPE. Given the two models and an input sequence $\boldsymbol{X}$, we computed the encoder hidden states of the given input sequence for each $k \in \{0, 100, 250, 500\}$. For each position i, we computed the cosine similarity (sim) of the hidden states from two offsets, i.e., $\boldsymbol{h}_i^{k_1}, \boldsymbol{h}_i^{k_2} \in \mathbb{R}^D$, and computed its average across the positions as

$$\frac{1}{I}\sum_{i=1}^{I} \mathrm{sim}(\boldsymbol{h}_i^{k_1}, \boldsymbol{h}_i^{k_2}). \tag{6.3}$$

As shown in Figure 6.3, SHAPE builds a shift-invariant representation; regardless of the offset k, the cosine similarity is almost always 1.0. Such invariance is nontrivial because the similarity of APE does not show similar characteristics.



(a) Sequence ID: #1 (b) Sequence ID: #2

(c) Sequence ID: #3 (d) Sequence ID: #4

(e) Sequence ID: #5 (f) Sequence ID: #6

(g) Sequence ID: #7 (h) Sequence ID: #8

(i) Sequence ID: #9 (j) Sequence ID: #10

Figure 6.3 Cosine similarities of encoder hidden states with different offsets $k \in \{0, 100, 250, 500\}$. Only the representation of SHAPE is invariant with k.

6.3.3 Experiment 2: Performance Comparison

We compared the overall performance of position representations on the validation and test sets as shown in Table 6.4. Figure 6.4 shows the BLEU improvement of RPE and SHAPE from APE with respect to the source sequence length.

On **Vanilla**, the three models show comparable results. APE being comparable to RPE is inconsistent with the result reported by Shaw et al. (2018); we assume that this is due to a difference in implementation. In fact, Narang et al. (2021) have recently reported that improvements in Transformer often do not transfer across implementations.

Table 6.4 BLEU scores on newstest2010-2016. **Valid** is the average of newstest2010-2013. **Test** is the average of newstest2014-2016. †: the values are averages of five distinct trials with five different random seeds. $*$: not available as the implementation was very slow. **Speed** is the relative speed to APE (larger is faster).

Dataset	Model	Valid	Test	Speed
VANILLA	APE†	23.61	30.46	x1.00
	RPE†	23.67	30.54	x0.91
	SHAPE†	23.63	30.49	x1.01
EXTRAPOLATE	APE	22.18	29.22	x1.00
	RPE	22.97	29.86	x0.91
	SHAPE	22.96	29.80	x0.99
INTERPOLATE	APE	31.40	38.23	x1.00
	RPE*	-	-	-
	SHAPE	32.50	39.09	x0.99

On **Extrapolate**, RPE (29.86) outperforms APE (29.22) by approximately 0.6 BLEU points on the test set; this is consistent with the result reported by Neishi and Yoshinaga (2019). Moreover, SHAPE achieves comparable test performance to RPE (29.80). According to Figure 6.4a, both RPE and SHAPE have improved extrapolation ability, i.e., better BLEU scores on sequences longer than those observed during training. In addition, Figure 6.4a shows the performance of SHAPE with the maximum shift $K = 40$ that was chosen on the basis of the BLEU score for the validation set. This model outperforms RPE, achieving BLEU scores of 23.12 and 29.86 on the validation and test sets, respectively. These results indicate that SHAPE can be a better alternative to RPE.

On **Interpolate**, we were unable to train RPE because its training was prohibitively slow[5]. Similarly to EXTRAPOLATE, SHAPE (39.09) outperforms APE (38.23) on the test set. Figure 6.4b shows that SHAPE consistently outperformed APE for every sequence length. From this result, we find that the shift invariance also improves the *interpolation ability* of Transformer.

[5]A single gradient step of RPE took about 5 seconds, which was 20 times longer than that of APE and SHAPE. We assume that the RPE implementation available in OpenNMT-py has difficulty in dealing with long sequences.

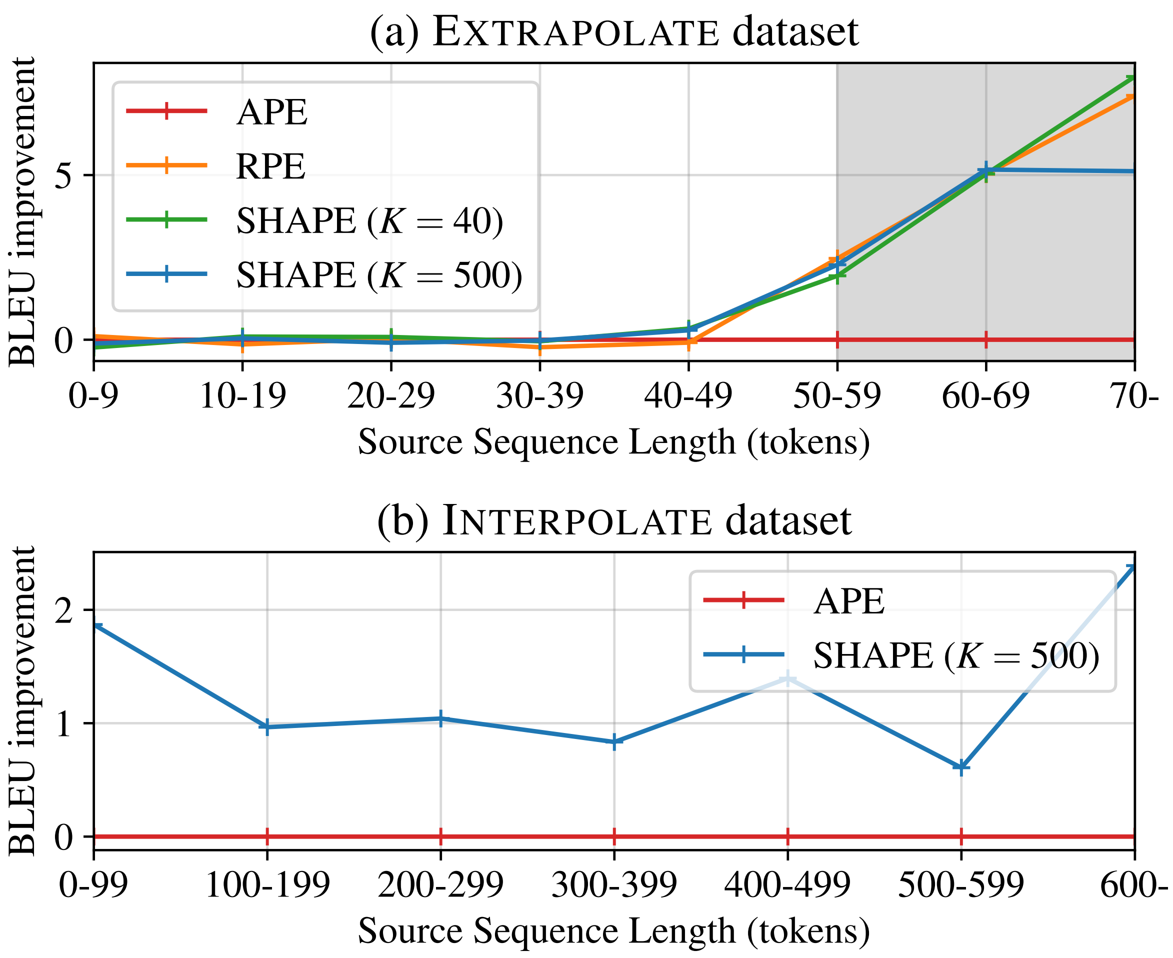
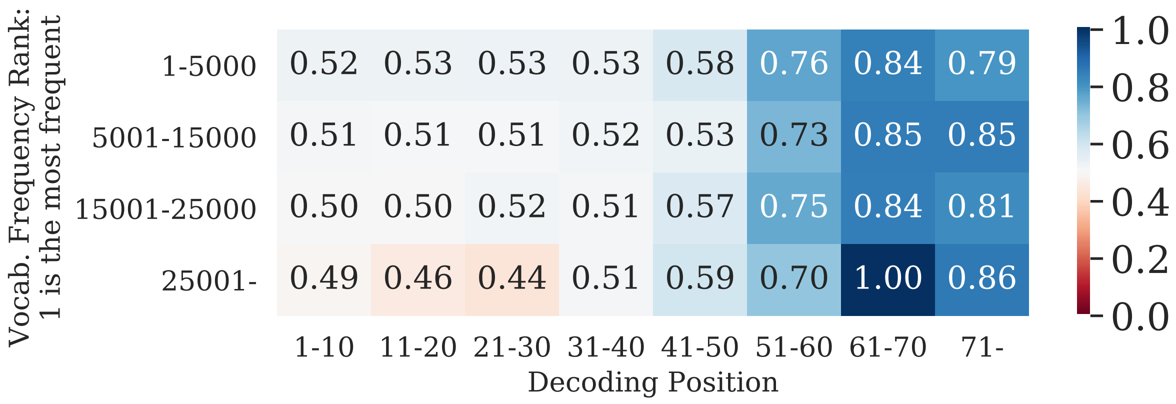


Figure 6.4 BLEU score improvement from APE on validation and test sets with respect to the source sequence length. The gray color means no training data.

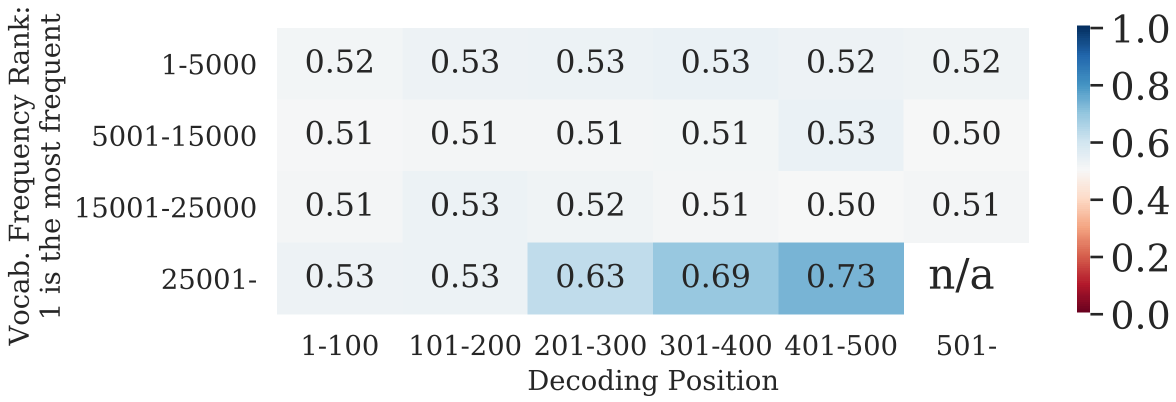
6.4 Analysis

This section provides a deeper analysis of how the model with translation invariance improves the performance. We hereinafter exclusively focus on APE and SHAPE because SHAPE achieves comparable performance to RPE, and we were unable to train RPE on the INTERPOLATE dataset as explained in footnote 5.

As discussed in Section 6.3.3, Figure 6.4 demonstrated that SHAPE outperformed APE in terms of BLEU score. However, BLEU evaluates two concepts simultaneously, that is, the token precision via n-gram matching and the output length via the brevity penalty (Papineni et al., 2002). Thus, the actual source of improvement remains unclear. We hereby exclusively analyzed the precision of token prediction. Specifically, we computed tokenwise scores assigned for gold references, and we then compared them across the models; given a sequence pair ($\boldsymbol{X}$, $\boldsymbol{Y}$) and a trained model, we computed a score (i.e., log probability) s_j for



(a) EXTRAPOLATE dataset



(b) INTERPOLATE dataset

Figure 6.5 Tokenwise analysis on gold references: the value in each cell represents the ratio that SHAPE assigns a higher score to a gold token than APE.

each token y_j in a teacher-forcing manner. Here, a higher score to gold token means better model performance. We used the validation set for comparison.

Figure 6.5 shows the ratio that SHAPE assigns a higher score to a gold token than APE, compared across for each position of the decoder.

Better extrapolation means better token precision Figure 6.5a shows that SHAPE outperforms APE, especially in the right part of the heat map. This area corresponds to sequences longer than those observed during training. This result indicates that better extrapolation in terms of BLEU score means better token precision.

Interpolation is particularly effective for rare tokens As shown in Figure 6.5b, SHAPE consistently outperforms APE and the performance gap is especially significant in the low-frequency region (bottom part). This indicates that SHAPE predicts rare words better than APE. One plausible explanation for this observation is that SHAPE carries out data augmentation in the sense that in each epoch, the same sequence pair is assigned a different position depending on the offset k. Rare words typically have sparse position distributions in training data and thus benefit from the extra position assignment during training.

6.5 Conclusion

We investigated SHAPE, a simple variant of APE with shift invariance. We demonstrated that SHAPE is empirically comparable to RPE yet imposes almost no computational overhead on APE. Our analysis revealed that SHAPE is effective at extrapolation to unseen lengths and interpolating rare words. SHAPE can be incorporated into the existing codebase with a few lines of code and no risk of a performance drop from APE; thus, we expect SHAPE to be used as a drop-in replacement for APE and RPE.

Chapter 7

Conclusion

In this thesis, we investigated the scalable task-oriented semi-supervised learning method for deep neural networks. Specifically, we have addressed the following research issues:

What makes task-oriented SSL scalable?: Findings on the performance scalability of task-oriented SSL are mixed in the research field. Thus, the properties that make the underlying method scalable (or unscalable) are unclear.

Scalable task-oriented SSL method applicable for arbitrary tasks: The task-oriented SSL method often makes the assumption on the target task. Thus, it is unclear if a finding on certain task can transfer to other tasks. In other words, the task-oriented SSL method applicable for arbitrary tasks is demanded.

What are the limits of task-oriented SSL methods?: Suppose that we have a scalable SSL method at hand; is incorporating more unlabeled data all we need for improving model performance? What are limitations or remaining challenges of scalable task-oriented SSL?

The key contributions of this thesis are summarized as follows:

Building a scalable task-oriented SSL method: We proposed a novel task-oriented SSL method. We demonstrated that our method scales to the amount of unlabeled data using the text classification benchmark data. Through the analysis, we investigated the requirements for a scalable task-oriented SSL method. Finally, we demonstrated that our method can be combined with the state-of-the-art generic SSL method to improve the performance.

Expanding the applicability of scalable task-oriented SSL method: We expanded the applicability of scalable task-oriented SSL by tackling one of sequence-to-sequence problems, namely, grammatical error correction (GEC). We conducted a controlled empirical comparison of existing task-oriented SSL methods on GEC. We identified the best scalable method through the experiment and achieved the state-of-the-art performance on multiple GEC benchmark datasets.

Exploring the limits of task-oriented SSL methods: For MT and GEC, we analyzed the models trained on a massive amount of unlabeled data. We revealed the limitations of the current best SSL methods for each task through the analysis.

Minimal architecture modification as a means of compensating the lack of data with desired property: We enhanced the state-of-the-art Transformer model to address the limitations of current SSL methods. Using the issue of length extrapolation, we demonstrated that an enhancement in Transformers' position representation can improve the model's generalization to the sequences that are longer than those observed during the training.

References

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. In *Proceedings of the 33th International Conference on Machine Learning (ICML 2016)*, pages 173–182.

Arora, S., Liang, Y., and Ma, T. (2017). A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Ba, J. and Caruana, R. (2014). Do Deep Nets Really Need to be Deep? In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems (NIPS 2014)*, volume 27. Curran Associates, Inc.

Baevski, A. and Auli, M. (2019). Adaptive Input Representations for Neural Language Modeling. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Barrault, L., Biesialska, M., Bojar, O., Costa-jussà, M. R., Federmann, C., Graham, Y., Grundkiewicz, R., Haddow, B., Huck, M., Joanis, E., Kocmi, T., Koehn, P., Lo, C.-k., Ljubešić, N., Monz, C., Morishita, M., Nagata, M., Nakazawa, T., Pal, S., Post, M., and Zampieri, M. (2020). Findings of the 2020 Conference on Machine Translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation (WMT 2020)*, pages 1–55.

Barrault, L., Bojar, O., Costa-jussà, M. R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Malmasi, S., Monz, C., Müller, M., Pal, S., Post, M., and Zampieri, M. (2019). Findings of the 2019 Conference on Machine Translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pages 1–61.

Bawden, R., Bogoychev, N., Germann, U., Grundkiewicz, R., Kirefu, F., Miceli Barone, A. V., and Birch, A. (2019). The University of Edinburgh's Submissions to the WMT19 News Translation Task. In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pages 103–115.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research (JMLR)*, 3(Feb):1137–1155.

Bennett, K. and Demiriz, A. (1999). Semi-Supervised Support Vector Machines. In Kearns, M., Solla, S., and Cohn, D., editors, *Advances in Neural Information Processing Systems (NIPS 1999)*, volume 11, pages 368–374. MIT Press.

Bogoychev, N., Heafield, K., Aji, A. F., and Junczys-Dowmunt, M. (2018). Accelerating Asynchronous Stochastic Gradient Descent for Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 2991–2996.

Bogoychev, N. and Sennrich, R. (2019). Domain, Translationese and Noise in Synthetic Data for Neural Machine Translation. *arXiv preprint arXiv:1911.03362*.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics (TACL 2017)*, 5:135–146.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the Opportunities and Risks of Foundation Models. *arXiv preprint arXiv:2108.07258*.

Bonial, C., Babko-Malaya, O., Choi, J. D., and Hwang, J. D. (2010). PropBank Annotation Guidelines.

Bouthillier, X., Laurent, C., and Vincent, P. (2019). Unreproducible Research is Reproducible. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, pages 725–734.

Bradbury, J., Merity, S., Xiong, C., and Socher, R. (2017). Quasi-Recurrent Neural Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Brockett, C., Dolan, W. B., and Gamon, M. (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 (ACL 2006)*, pages 249–256.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33, pages 1877–1901. Curran Associates, Inc.

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 (BEA 2019)*, pages 52–75.

Bryant, C., Felice, M., and Briscoe, T. (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 (ACL 2017)*, pages 793–805.

Burlot, F. and Yvon, F. (2018). Using Monolingual Data in Neural Machine Translation: a Systematic Study. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pages 144–155.

Cahill, A., Madnani, N., Tetreault, J., and Napolitano, D. (2013). Robust Systems for Preposition Error Correction Using Wikipedia Revisions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2013)*, pages 507–517.

Caswell, I., Chelba, C., and Grangier, D. (2019). Tagged Back-Translation. In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pages 53–63.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., and Kurzweil, R. (2018). Universal Sentence Encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2014). One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association*, pages 2635–2639.

Chen, J., Yang, Z., and Yang, D. (2020a). MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 2147–2157.

Chen, L., Garcia, F., Kumar, V., Xie, H., and Lu, J. (2021). Industry Scale Semi-Supervised Learning for Natural Language Understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, pages 311–318.

Chen, L., Ruan, W., Liu, X., and Lu, J. (2020b). SeqVAT: Virtual Adversarial Training for Semi-Supervised Sequence Labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8801–8811.

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'16, pages 785–794. ACM.

Choe, Y. J., Ham, J., Park, K., and Yoon, Y. (2019). A Neural Grammatical Error Correction System Built On Better Pre-training and Sequential Transfer Learning. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2019)*, pages 213–227.

Chollampatt, S. and Ng, H. T. (2018). A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, pages 5755–5762.

Chollampatt, S., Wang, W., and Ng, H. T. (2019). Cross-Sentence Grammatical Error Correction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 435–445.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, Ł., et al. (2021). Rethinking Attention with Performers. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Clark, K., Luong, M.-T., Manning, C. D., and Le, Q. (2018). Semi-Supervised Sequence Modeling with Cross-View Training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 1914–1925.

Dahlmeier, D. and Ng, H. T. (2012). Better Evaluation for Grammatical Error Correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2012)*, pages 568–572.

Dahlmeier, D., Ng, H. T., and Wu, S. M. (2013). Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of the 8th Workshop on Building Educational Applications Using NLP (BEA 2013)*, pages 22–31.

Dai, A. M. and Le, Q. V. (2015). Semi-supervised Sequence Learning. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems (NIPS 2015)*, volume 28, pages 3079–3087. Curran Associates, Inc.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 2978–2988.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, pages 248–255.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2019)*, pages 4171–4186.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. (2019). DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. 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 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Dufter, P., Schmitt, M., and Schütze, H. (2021). Position Information in Transformers: An Overview. *arXiv preprint arXiv:2102.11090*.

Edunov, S., Ott, M., Auli, M., and Grangier, D. (2018). Understanding Back-Translation at Scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 489–500.

Felice, M., Bryant, C., and Briscoe, T. (2016). Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835.

Felice, M. and Yuan, Z. (2014). Generating artificial errors for grammatical error correction. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL-SRW 2014)*, pages 116–126.

Foster, J. and Andersen, O. (2009). GenERRate: Generating Errors for Use in Grammatical Error Detection. In *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2009)*, pages 82–90.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. (2021). The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027.*

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. (2018). AllenNLP: A Deep Semantic Natural Language Processing Platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6.

Ge, T., Wei, F., and Zhou, M. (2018). Fluency Boost Learning and Inference for Neural Grammatical Error Correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 1055–1065.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional Sequence to Sequence Learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, pages 1243–1252.

Geva, M., Gupta, A., and Berant, J. (2020). Injecting Numerical Reasoning Skills into Language Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 946–958.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. In Gordon, G., Dunson, D., and DudÃk, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*, chapter 7.4, pages 233–234. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015).*

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. (2017). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv preprint arXiv:1706.02677.*

Granger, S. (1998). The computer learner corpus: A versatile new source of data for SLA research. In Granger, S., editor, *Learner English on Computer*, pages 3–18. Addison Wesley Longman, London and New York.

Grundkiewicz, R. and Junczys-Dowmunt, M. (2018). Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 284–290.

Grundkiewicz, R., Junczys-Dowmunt, M., and Heafield, K. (2019). Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2019)*, pages 252–263.

Haddow, B., Bogoychev, N., Emelin, D., Germann, U., Grundkiewicz, R., Heafield, K., Miceli Barone, A. V., and Sennrich, R. (2018). The University of Edinburgh's Submissions to the WMT18 News Translation Task. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pages 399–409.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., et al. (2018). Achieving Human Parity on Automatic Chinese to English News Translation. *arXiv preprint arXiv:1803.05567*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 770–778.

He, P., Liu, X., Gao, J., and Chen, W. (2020). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *arXiv preprint arXiv:2006.03654*.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Huang, Z., Liang, D., Xu, P., and Xiang, B. (2020). Improve Transformer Models with Better Relative Position Embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3327–3335.

Izumi, E., Uchimoto, K., Saiga, T., Supnithi, T., and Isahara, H. (2003). Automatic Error Detection in the Japanese Learners' English Spoken Data. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 145–148.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87.

Ji, J., Wang, Q., Toutanova, K., Gong, Y., Truong, S., and Gao, J. (2017). A Nested Attention Neural Hybrid Model for Grammatical Error Correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 753–762.

Johnson, R. and Zhang, T. (2015). Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems (NIPS 2015)*, volume 28. Curran Associates, Inc.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P.-l., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T. V., Gottipati, R., Gulland, W., Hagmann, R., Ho, C. R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., and Yoon, D. H. (2017). In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 1–12, New York, NY, USA. Association for Computing Machinery.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the Limits of Language Modeling. *arXiv preprint arXiv:1602.02410*.

Junczys-Dowmunt, M. (2019). Microsoft Translator at WMT 2019: Towards Large-Scale Document-Level Neural Machine Translation. In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pages 225–233.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2016). Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1546–1556.

Junczys-Dowmunt, M., Grundkiewicz, R., Guha, S., and Heafield, K. (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 (NAACL 2018)*, pages 595–606.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*.

Kasewa, S., Stenetorp, P., and Riedel, S. (2018). Wronging a Right: Generating Better Errors to Improve Grammatical Error Detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 4977–4983.

Kiela, D., Bartolo, M., Nie, Y., Kaushik, D., Geiger, A., Wu, Z., Vidgen, B., Prasad, G., Singh, A., Ringshia, P., Ma, Z., Thrush, T., Riedel, S., Waseem, Z., Stenetorp, P., Jia, R., Bansal, M., Potts, C., and Williams, A. (2021). Dynabench: Rethinking Benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124.

Kingma, D. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Kitaev, N., Kaiser, Ł., and Levskaya, A. (2020). Reformer: The Efficient Transformer. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*.

Kiyono, S., Suzuki, J., and Inui, K. (2019). Mixture of Expert/Imitator Networks: Scalable Semi-Supervised Learning Framework. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4073–4081. AAAI Press.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72.

Kobayashi, H. (2018). Frustratingly Easy Model Ensemble for Abstractive Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 4165–4176.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

Koehn, P., Khayrallah, H., Heafield, K., and Forcada, M. L. (2018). Findings of the WMT 2018 Shared Task on Parallel Corpus Filtering. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pages 726–739.

Koehn, P. and Knowles, R. (2017). Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Lau, J. H., Armendariz, C., Purver, M., Shu, C., and Lappin, S. (2020). How Furiously Can Colourless Green Ideas Sleep? Sentence Acceptability in Context. *Transactions of the Association for Computational Linguistics*, 8:296–310.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al. (2015). DBpedia–A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web*, 6(2):167–195.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research (JMLR)*, 5:361–397.

Lichtarge, J., Alberti, C., Kumar, S., Shazeer, N., Parmar, N., and Tong, S. (2019). Corpora Generation for Grammatical Error Correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)*.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A Structured Self-attentive Sentence Embedding. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

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

Liu, L., Utiyama, M., Finch, A., and Sumita, E. (2016). Agreement on Target-bidirectional Neural Machine Translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*, pages 411–416.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1412–1421.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*.

Marie, B., Rubino, R., and Fujita, A. (2020). Tagged Back-translation Revisited: Why Does It Really Work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 5990–5997.

Maruf, S., Saleh, F., and Haffari, G. (2021). A Survey on Document-Level Neural Machine Translation: Methods and Evaluation. *ACM Computing Survey*, 54(2).

McAuley, J. and Leskovec, J. (2013). Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 165–172, New York, NY, USA. Association for Computing Machinery.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in Translation: Contextualized Word Vectors. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems (NIPS 2017)*, volume 30, pages 6294–6305. Curran Associates, Inc.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems (NIPS 2013)*, volume 26. Curran Associates, Inc.

Miyato, T., Dai, A. M., and Goodfellow, I. (2017). Adversarial Training Methods For Semi-Supervised Text Classification. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Mizumoto, T., Komachi, M., Nagata, M., and Matsumoto, Y. (2011). Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pages 147–155.

Morishita, M., Suzuki, J., and Nagata, M. (2019). NTT Neural Machine Translation Systems at WAT 2019. In *Proceedings of the 6th Workshop on Asian Translation (WAT 2019)*, pages 99–105.

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 (ACL & IJCNLP 2015)*, pages 588–593.

Napoles, C., Sakaguchi, K., Post, M., and Tetreault, J. (2016). GLEU Without Tuning. *arXiv preprint arXiv:1605.02592*.

Napoles, C., Sakaguchi, K., and Tetreault, J. (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 (EACL 2017)*, pages 229–234.

Narang, S., Chung, H. W., Tay, Y., Fedus, W., Fevry, T., Matena, M., Malkan, K., Fiedel, N., Shazeer, N., Lan, Z., et al. (2021). Do Transformer Modifications Transfer Across Implementations and Applications? *arXiv preprint arXiv:2102.11972*.

Neishi, M. and Yoshinaga, N. (2019). On the Relation between Position Information and Sentence Length in Neural Machine Translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL 2019)*, pages 328–338.

Newman, B., Hewitt, J., Liang, P., and Manning, C. D. (2020). The EOS Decision and Length Extrapolation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP (BlackboxNLP 2020)*, pages 276–291.

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*, pages 1–14.

Ng, N., Yee, K., Baevski, A., Ott, M., Auli, M., and Edunov, S. (2019). Facebook FAIR's WMT19 News Translation Task Submission. In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pages 314–319.

Och, F. J. (2003). Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 160–167.

Oka, Y., Chousa, K., Sudoh, K., and Nakamura, S. (2020). Incorporating Noisy Length Constraints into Transformer with Length-aware Positional Encodings. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING 2020)*, pages 3580–3585.

Oka, Y., Sudoh, K., and Nakamura, S. (2021). Using Perturbed Length-aware Positional Encoding for Non-autoregressive Neural Machine Translation. *arXiv preprint arXiv:2107.13689*.

Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I. (2018). Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems (NeurIPS 2018)*, volume 31. Curran Associates, Inc.

Ortiz Suárez, P. J., Sagot, B., and Romary, L. (2019). Asynchronous Pipelines for Processing Huge Corpora on Medium to Low Resource Infrastructures. Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019, pages 9–16, Mannheim. Leibniz-Institut für Deutsche Sprache.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Ott, M., Edunov, S., Grangier, D., and Auli, M. (2018). Scaling Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pages 1–9.

Pang, B. and Lee, L. (2005). Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 115–124.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 311–318.

Pauls, A. and Klein, D. (2012). Large-Scale Syntactic Language Modeling with Treelets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 959–968.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543.

Peters, M. E., Ammar, W., Bhagavatula, C., and Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep Contextualized Word Representations. 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 2227–2237.

Popel, M. and Bojar, O. (2018). Training Tips for the Transformer Model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Popović, M. (2015). chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

Post, M. (2018). A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers (WMT 2018)*, pages 186–191.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research (JMLR)*, 21(140):1–67.

Rei, M., Felice, M., Yuan, Z., and Briscoe, T. (2017). Artificial Error Generation with Machine Translation and Syntactic Patterns. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2017)*, pages 287–292.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. 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 2019)*, pages 3982–3992.

Rosendahl, J., Tran, V. A. K., Wang, W., and Ney, H. (2019). Analysis of Positional Encodings for Neural Machine Translation. In *Proceedings of 16th International Workshop on Spoken Language Translation 2019 (IWSLT 2019)*.

Rozovskaya, A. and Roth, D. (2010a). Generating Confusion Sets for Context-Sensitive Error Correction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 961–970.

Rozovskaya, A. and Roth, D. (2010b). Training Paradigms for Correcting Errors in Grammar and Usage. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2010)*, pages 154–162.

Rozovskaya, A., Sammons, M., and Roth, D. (2012). The UI System in the HOO 2012 Shared Task on Error Correction. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP (BEA 2012)*, pages 272–280.

Sato, M., Suzuki, J., and Kiyono, S. (2019). Effective Adversarial Regularization for Neural Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 204–210.

Sato, M., Suzuki, J., Shindo, H., and Matsumoto, Y. (2018). Interpretable Adversarial Perturbation in Input Embedding Space for Text. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 4323–4330.

Saunshi, N., Malladi, S., and Arora, S. (2021). A Mathematical Exploration of Why Language Models Help Solve Downstream Tasks. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Sennrich, R., Birch, A., Currey, A., Germann, U., Haddow, B., Heafield, K., Miceli Barone, A. V., and Williams, P. (2017). The University of Edinburgh's Neural MT Systems for WMT17. In *Proceedings of the Second Conference on Machine Translation (WMT 2017)*, pages 389–399.

Sennrich, R., Haddow, B., and Birch, A. (2016a). Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation (WMT 2016)*, pages 371–376.

Sennrich, R., Haddow, B., and Birch, A. (2016b). Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 86–96.

Sennrich, R., Haddow, B., and Birch, A. (2016c). Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1715–1725.

Sennrich, R. and Zhang, B. (2019). Revisiting Low-Resource Neural Machine Translation: A Case Study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 211–221.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers) (NAACL 2018)*, pages 464–468.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Shazeer, N. and Stern, M. (2018). Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pages 4603–4611.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. (2019). MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, pages 5926–5936.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems (NIPS 2014)*, volume 27, pages 3104–3112. Curran Associates, Inc.

Suzuki, J. and Isozaki, H. (2008). Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data. In *Proceedings of ACL-08: HLT*, pages 665–673.

Suzuki, J., Isozaki, H., Carreras, X., and Collins, M. (2009). An Empirical Study of Semi-supervised Structured Conditional Models for Dependency Parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 551–560.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 2818–2826.

Tajiri, T., Komachi, M., and Matsumoto, Y. (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 (ACL 2012)*, pages 198–202.

Takase, S. and Kiyono, S. (2021). Rethinking Perturbations in Encoder-Decoders for Fast Training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2021)*, pages 5767–5780.

Takase, S. and Okazaki, N. (2019). Positional Encoding to Control Output Sequence Length. 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 3999–4004.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2020). Efficient Transformers: A Survey. *arXiv preprint arXiv:2009.06732*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems (NIPS 2017)*, volume 30, pages 5998–6008. Curran Associates, Inc.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine learning (ICML 2008)*, pages 1096–1103.

Wang, A. and Cho, K. (2019). BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 30–36.

Wang, A., Hula, J., Xia, P., Pappagari, R., McCoy, R. T., Patel, R., Kim, N., Tenney, I., Huang, Y., Yu, K., Jin, S., Chen, B., Van Durme, B., Grave, E., Pavlick, E., and Bowman, S. R. (2019a). Can You Tell Me How to Get Past Sesame Street? Sentence-Level Pre-training Beyond Language Modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 4465–4476.

Wang, B., Shang, L., Lioma, C., Jiang, X., Yang, H., Liu, Q., and Simonsen, J. G. (2021). On Position Embeddings in BERT. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D. F., and Chao, L. S. (2019b). Learning Deep Transformer Models for Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 1810–1822.

Wang, R., Marie, B., Utiyama, M., and Sumita, E. (2018). NICT's Corpus Filtering Systems for the WMT18 Parallel Corpus Filtering Task. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pages 963–967.

Wu, C., Wu, F., and Huang, Y. (2021). DA-Transformer: Distance-aware Transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2021)*, pages 2059–2068.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*.

Xie, Z., Genthial, G., Xie, S., Ng, A., and Jurafsky, D. (2018). Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 619–628.

Yang, X., Song, Z., King, I., and Xu, Z. (2021). A Survey on Deep Semi-supervised Learning. *arXiv preprint arXiv:2103.00550*.

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.

Yannakoudakis, H., Briscoe, T., and Medlock, B. (2011). A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pages 180–189.

Yee, K., Dauphin, Y., and Auli, M. (2019). Simple and Effective Noisy Channel Modeling for Neural Machine Translation. 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 2019)*, pages 5696–5701.

Yokoi, S., Kobayashi, S., Fukumizu, K., Suzuki, J., and Inui, K. (2018). Pointwise HSIC: A Linear-Time Kernelized Co-occurrence Norm for Sparse Linguistic Expressions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 1763–1775.

Yuan, Z. and Briscoe, T. (2016). Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2016)*, pages 380–386.

Yuan, Z. and Felice, M. (2013). Constrained Grammatical Error Correction using Statistical Machine Translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task (CoNLL 2013)*, pages 52–61.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020). Big Bird: Transformers for Longer Sequences. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33. Curran Associates, Inc.

Zhao, W., Wang, L., Shen, K., Jia, R., and Liu, J. (2019). Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)*.

List of Publications

Journal Papers (Refereed)

1. Shun Kiyono, Jun Suzuki, Tomoya Mizumoto and Kentaro Inui. Massive Exploration of Pseudo Data for Grammatical Error Correction. In IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 28, pp. 2134–2145, 2020, doi: 10.1109/TASLP.2020.3007753.

International Conference Papers (Refereed)

1. Shun Kiyono, Sosuke Kobayashi, Jun Suzuki, Kentaro Inui. SHAPE: Shifted Absolute Position Embeddings for Transformers. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021), pp.3309–3321, November 2021.

2. Shun Kiyono, Takumi Ito, Ryuto Konno, Makoto Morishita and Jun Suzuki. Tohoku-AIP-NTT at WMT 2020 News Translation Task. In Proceedings of the Fifth Conference on Machine Translation (WMT 2020), pp.144–154, November 2020.

3. Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto and Kentaro Inui. An Empirical Study of Incorporating Pseudo Data into Grammatical Error Correction. In Proceedings of 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019), pp. 1236–1242, November 2019.

4. Shun Kiyono, Jun Suzuki, Kentaro Inui. Mixture of Expert/Imitator Networks: Scalable Semi-supervised Learning Framework. In Proceedings of The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19), pp.4073–4081, January 2019.

5. Shun Kiyono, Sho Takase, Jun Suzuki, Naoaki Okazaki, Kentaro Inui and Masaaki Nagata. Reducing Odd Generation from Neural Headline Generation. In Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation (PACLIC32), pp.290–303, December 2018.

6. Shun Kiyono, Sho Takase, Jun Suzuki, Naoaki Okazaki, Kentaro Inui and Masaaki Nagata. Unsupervised Token-wise Alignment to Improve Interpretation of Encoder-Decoder Models. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pp.74–81, October 2018.

Awards

1. Asia-Pacific Association for Machine Translation (AAMT) 第 16 回長尾賞
2. 言語処理学会第 26 回年次大会 (NLP2020) 優秀賞
3. 言語処理学会第 25 回年次大会 (NLP2019) 優秀賞
4. 言語処理学会第 24 回年次大会 (NLP2018) 優秀賞