論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Addressing Text Degeneration of Discriminative Models with Re-ranking Methods
著者(和文)	ZhangYing
Author(English)	Ying Zhang
出典(和文)	学位:博士（工学）, 学位授与機関:東京工業大学, 報告番号:甲第12621号, 授与年月日:2023年12月31日, 学位の種別:課程博士, 審査員:奥村 学,熊澤 逸夫,中山 実,篠崎 隆宏,船越 孝太郎
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第12621号, Conferred date:2023/12/31, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Tokyo Institute of Technology
Department of Information and Communications Engineering

Academic Year 2023

Addressing Text Degeneration of Discriminative Models with Re-ranking Methods

Zhang Ying

Department of Information and Communications Engineering
School of Engineering

Supervisor Okumura Manabu, Professor

Abstract

Natural language processing (NLP) involves tasks that enable computers to process and understand human languages, whether in text or speech, with the goal of improving communications between humans and machines, as well as among humans themselves. The origins of NLP can be traced back to the 1950s when Alan Turing introduced the Turing Test as a tool to determine whether a computer is as skillful as a human being in processing human languages. With the rapid development of the Internet and computer hardware in recent decades, the field of NLP has experienced significant growth, and many NLP applications are now widely used in our daily lives, such as Google Translate and Apple Siri. One notable advancement in recent years is the emergence of ChatGPT, a large-scale intelligent language model, that has gained global attention for its impressive conversational capabilities.

In NLP, discriminative models are often preferred over generative models owing to their direct computation and proven success in various applications such as machine translation, text summarization, and named entity recognition, despite challenges such as unbalanced data distribution and limited training data. However, recent studies have highlighted two significant issues that arise when using discriminative models: text degeneration and exposure bias, which can hurt the quality of the models' predictions.

The primary objective of this thesis is to improve the prediction results from discriminative models by presenting two re-ranking-based methods: the *language model-based reranker* (LMR) and the *bidirectional Transformer reranker* (BTR), to counter the decoding issues of text degeneration and exposure bias. The aim of this study is to propose general ideas that can be easily implemented in NLP tasks, including discourse parsing and grammatical error correction, without requiring complex modifications to the existing discriminative model structure or excessive computational resources.

The proposed LMR is designed as a generative model that leverages information from the top-α candidates generated by a discriminative model. In addition to the source text, the LMR also treats each candidate as the input. For each candidate, the LMR employs a masked language model and pseudo-log-likelihood scores (PLL) to estimate the sequence probability based on the bidirectional representation of the concatenated sequence comprising the source text and the candidate. This thesis uses contrastive learning during training to increase the probability difference between the true sequence and unmatched sequences, expecting to enhance the ability of the LMR to distinguish between them, thereby mitigating issues such as text degeneration. Experimental results on the RST-DT dataset demonstrate the effectiveness of LMR in discourse segmentation and sentence-level discourse parsing tasks.

Building upon the success of the LMR, this thesis extends it to a seq2seq architecture, resulting in the BTR. The BTR preserves the seq2seq-style Transformer architecture and incorporates a masked language model in the decoder component to re-estimate the probability of each candidate through PLL based on the corresponding bidirectional representations. During inference, the BTR compares the re-ranked top-1 results with the original ones using an acceptance threshold β to determine the final results. Experimental results on three publicly available grammatical error correction datasets demonstrates the effectiveness of BTR in overcoming exposure bias and text degeneration.

The experimental results presented in this study provide strong evidence for the effectiveness of the proposed re-ranking-based methods in addressing decoding problems. Moreover, these results suggest that future research could explore the potential benefits of combining PLL-based and discriminative models. Building upon these findings, this thesis offers valuable insights into potential directions, including label embedding, masking, and negative sampling strategies, to benefit the NLP community.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis focuses on the objective function f in natural language processing tasks, given a source text $X = (x_1, \ldots, x_I)$ of length I over the source language vocabulary $\mathcal{V}_s$. The objective is to generate an appropriate text or classify the source text X into a sequence of labels $Y = (y_1, \ldots, y_J)$, composed of J tokens or labels from the target language vocabulary or the label set $\mathcal{V}_t$ with respect to the tasks [Jurafsky, 2000, Liu et al., 2022]. The function f can be expressed as:

$$f : X \to Y. \tag{1.0.1}$$

Here, the tokens x_i and y_j are the minimum units in the tokenized or chopped text, and could be characters, words, or any other relevant units. The label y_j indicates what have been perceived by our senses from the source tokens [Jurafsky, 2000].

Previous research has commonly used probability theories to construct a function f that allows machines to learn from data and make predictions for new inputs. These functions can be categorized into two types: discriminative models and generative models. Discriminative models make predictions by directly modeling the posterior probability $p(Y|X)$, i.e. they learn a direct map from input text X to predicted text Y. Conversely, given all possible Y, generative models compute the joint probability $p(X, Y)$ for each Y and use Bayes rules [Webb, 2010] to calculate $p(Y|X)$ as:

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} \propto p(X, Y), \tag{1.0.2}$$

and then select the most likely Y as the predicted text [Ng and Jordan, 2001].

To develop a text processing system, it is crucial to determine which model to use and how to use it to make accurate predictions involving multiple discrete tokens.

1.1 Successful Discriminative Models

Despite issues such as unbalanced data distribution and insufficient data for training, discriminative models have been preferred in most cases over generative models owing to their direct computation without an intermediate step [Ng and Jordan, 2001, Yogatama et al., 2017] and their success in many applications.

In particular, to recognize the coherent structure of a natural language text, Wang et al. [2018c] utilized a discriminative model that combines bidirectional long short-term memory networks (LSTM) [Hochreiter and Schmidhuber, 1997] with conditional random fields (CRF) [Lafferty et al., 2001], called BiLSTM-CRF [Huang et al., 2015], to directly detect a set of rhetorical structure boundaries from the source text. Given the source text and corresponding boundaries, Wang et al. [2017] and Kobayashi et al. [2020] also modeled parsers with conditional probabilities to link these rhetorical structures. Chapter 3 details the proposals of Huang et al. [2015] and Wang et al. [2017].

In addition, Luong et al. [2015] introduced a global attention-based sequence-to-sequence (seq2seq) model that decomposes the conditional probability for neural machine translation. Two years later, Vaswani et al. [2017] proposed a variant of the seq2seq model, Transformer, which used a multi-head self-attention mechanism and significantly outperformed previous models, including the global attention-based seq2seq model, to become the dominant paradigm in natural language processing tasks. Both the seq2seq model and Transformer are discussed in detail in Chapter 2.

With the popularity of Transformer, several large pre-trained seq2seq-style Transformer models, including BART [Lewis et al., 2020], mBART [Liu et al., 2020], ProphetNet [Qi et al., 2020], PEGASUS [Zhang et al., 2020], T5 [Raffel et al., 2020], and mT5 [Xue et al., 2021], have been developed to learn general representations and have demonstrated effectiveness in making high-quality predictions, reducing time- and resource-cost for downstream tasks. Specifically, as reported by Rothe et al. [2021], after successively fine-tuning with the cleaned LANG-8 corpus (cLang-8) [Rothe et al., 2021], the pre-trained T5 and mT5 models achieved state-of-the-art results on grammatical error correction (GEC) benchmarks for four languages. Besides, the experimental results in Liu et al. [2022] demonstrated that simply fine-tuning pre-trained PEGASUS and BART models could achieve comparable results on two abstractive summarization datasets: CNN/DM [Hermann et al., 2015] and XSum [Narayan et al., 2018]. mBART has also shown its effectiveness in several high- and low-resource machine translation tasks, including English ⇔ German and English ⇔ Romanian.

Given the success of applying discriminative models in natural language processing, this thesis is concerned with improving the prediction results of such models.

1.2 Training and Decoding in Discriminative Models

Let $p_*(Y|X)$ be the true probability distribution. Given a sampled corpus $\mathcal{D} = \{(X_n, Y_n)\}_{n=1}^{N}$ with N input-output pairs from $p_*(Y|X)$, we expect to model a probability distribution $p(Y|X,\theta)$ with parameter θ, and find a most likely model $p(Y|X,\theta^*)$ to resemble $p_*(Y|X)$, s.t. $p(Y|X,\theta^*) \approx p_*(Y|X)$ for all pairs of (X,Y) in $\mathcal{D}$ [Welleck et al., 2020]. This task of finding the most likely parameter θ^* to fit in the corpus $\mathcal{D}$ is known as the training or optimization problem, and the *de facto* approach is to use maximum likelihood estimation (MLE) [Murphy, 2012] to update initialized θ as:

$$\theta^* := \arg\max_{\theta} \prod_{(X,Y)\in\mathcal{D}} p(Y|X;\theta) = \arg\max_{\theta} \sum_{(X,Y)\in\mathcal{D}} \log p(Y|X;\theta), \tag{1.2.1}$$

where $p(Y|X;\theta)$ could be decomposed according to the chain rule as an auto-regressive language model that sequentially predicts the next token by giving previous tokens [Yang et al., 2018] as:

$$p(Y|X;\theta) = \prod_{j=1}^{J} p(y_j|Y_{<j}, X;\theta), \tag{1.2.2}$$

where $Y_{<j} = (y_1, \ldots, y_{j-1})$ and y_J denotes the end token <eos>. Chapter 2 details the language model.

Given the source X and the trained model with optimized θ^*, finding the most likely prediction Y^* is referred to as the decoding or inference problem, which is usually solved using maximum a posterior (MAP) estimation [Smith, 2011, Eikema and Aziz, 2020] to decode the posterior distribution $p(Y|X;\theta^*)$ as:

$$Y^* = \arg\max_{Y} \log p(Y|X;\theta^*). \tag{1.2.3}$$

Once we decompose $p(Y|X;\theta)$ as Equation (1.2.2), $p(Y|X;\theta^*)$ could be decomposed in the same way as:

$$p(Y|X;\theta^*) = \prod_{j=1}^{J} p(y_j|Y_{<j}, X;\theta^*), \tag{1.2.4}$$

and it will continue generating tokens until emitting the end token <eos>.

We can obtain the prediction Y^* by finding the sequence with the highest probability $p(Y^*|X;\theta^*)$ after enumerating all possible sequences through exhaustive search. However, the computational complexity of this procedure grows exponentially as $O(|\mathcal{V}_t|^J)$ when using Equation (1.2.4), where $|\mathcal{V}_t|$ is the target vocabulary size. For example, if $|\mathcal{V}_t| = 10000$ and $J = 10$, the number of possible sequences would be 10^{40}, making complete enumeration impractical [Stahlberg and Byrne, 2019]. Conversely, the simplest algorithm is greedy decoding with computational complexity $O(|\mathcal{V}_t| \times J)$. This algorithm picks one symbol y_j^* with the highest probability $p(y_j^*|Y_{<j}^*, X;\theta^*)$ at a time and could result in sub-optimal results, because only one hypothesis is tracked [Gu et al., 2017]. Therefore, the natural choice is to use a heuristic search algorithm, such as beam search, to track k hypotheses or beams with computational complexity $O(|\mathcal{V}_t| \times J \times k)$, where k is the beam size. Contrary to greedy decoding, beam search keeps track of the top-k highest scoring partial hypotheses at each time step and returns the hypothesis with the highest log probability from the retained k hypotheses.

1.3 Problems

Although discriminative models use MLE and MAP as their objective function and decoding strategy, respectively, and can generate high-quality predictions for a wide range of tasks, they are still criticized as suffering from several problems. 1) The first is **exposure bias**, which occurs when the model is only exposed to the training data distribution without considering its own predictions. This can cause brittle generation during inference, especially when using MAP with limited search space, such as beam search [Ranzato et al., 2016]. In details, we usually adopt teacher forcing [Williams and Zipser, 1989] in Equation (1.2.2) to feed the true previous tokens into the model for predicting the next token. However, in Equation (1.2.4), the predicted previous tokens are fed into the model with no future time steps factored in, unlike dynamic programming [Bellman, 1952]. This can cause cumulative errors in inference, where wrong decisions made at one time step lead to incorrect predictions for future tokens [Bengio et al., 2015]. 2) The second problem is **text degeneration**, that even a successfully pre-trained model [Radford et al., 2019] can assign a higher probability to generic, repetitive, and awkward beam-search-decoded texts than to grammatical or natural text. Consequently, the output text may appear bland, incoherent, or strangely repetitive, while human-like text has a more diverse and surprising probability distribution [Holtzman et al., 2020]. This problem is also evident in the GEC task, as demonstrated by Liu et al. [2021], where even though a fully pre-trained seq2seq model [Kiyono et al., 2019] can generate several high-quality grammatical candidates using beam search for an ungrammatical sentence, there may be still a gap between the selected hypothesis and the most grammatical one within these candidates. 3) The third problem is known as **search error**, which arises in inference when using Equation (1.2.4) with a more comprehensive search, such as beam search with a larger beam size. This can result in larger conditional probabilities but shorter predictions with worse text quality [Sountsov and Sarawagi, 2016, Stahlberg and Byrne, 2019, Meister et al., 2020]. As demonstrated by Yang et al. [2018], an increased search space with more tracks could make it easier to explore the end token <eos>, resulting in shorter hypotheses and worse evaluation results in the tokenized BLEU metric [Papineni et al., 2002]. Furthermore, Sountsov and Sarawagi [2016] demonstrated that using MLE for the training objective in Equation (1.2.2) underestimates the margin of separating long sequences from short ones, thereby resulting in a preference for shorter sentences than longer ones in inference when using MAP. 4) The fourth problem is the issue of **low diversity**, where the tracks retained by beam search for Equation (1.2.4) tend to have a common part, resulting in generic or "safe" outputs with low diversity [Li and Jurafsky, 2016, Vijayakumar et al., 2018, Ippolito et al., 2019].

To address these problems, existing methods can be classified into four categories: **decoding-based**, **optimization-based**, **scoring-based**, and **re-ranking-based** methods. 1) **Decoding-based methods** aim to solve these problems by extending standard beam search algorithms [Li and Jurafsky, 2016, Vijayakumar et al., 2018, Kulikov et al., 2019] or replacing MAP with stochastic sampling strategies [Fan et al., 2018, Holtzman et al., 2020]. For example, Vijayakumar et al. [2018] extended beam search by dividing k beams into groups and incorporating diversity constraints within groups to promote both diversity and quality of decoded sequences in image captioning and visual question generation tasks. Fan et al. [2018] introduced a top-k sampling strategy to sample the next word from the top-k most likely candidates for the neural story generation task. However, selecting a suitable k for this strategy can be a problem, as generation according to this strategy could have high variance in likelihood with larger k, which leads to high diversity but usually results in incoherency issues, as indicated by Holtzman et al. [2020]. Therefore, Holtzman et al. [2020] proposed a dynamical sampling strategy, nucleus sampling (or top-p sampling), to sample the next word from the top-p portion of the probability mass. Sountsov and Sarawagi [2016] also indicated that these post-processing methods could reduce the appearances of <eos> and indirectly alleviate search errors. It is important to note that while these post-processing methods alleviate the decoding problems, they do not address the core issue that the model assign too high probabilities to generic and awkward sequences instead of generating human-like texts, and they still suffer from exposure bias with no future time steps being factorized. 2) Previous research [Welleck et al., 2020, Liu et al., 2022] has argued that MLE optimization is the main cause of dull and repetitive outputs, and **optimization-based methods** have been utilized to tackle it by introducing new optimizations that work with MLE. Specifically, Welleck et al. [2020] proposed unlikelihood training, which minimizes the probability of unlikely predictions (or negative candidates) while maximizing the probability of true targets. In contrast, to alleviate exposure bias, Liu et al. [2022] assumed that sentence probabilities should be correlated with their quality as evaluated by an automatic metric, and modified contrastive learning [Hadsell et al., 2006] to encourage the model to assign higher probabilities to higher-quality candidates. However, these two methods compared two or more candidates to normalize $p(X;\theta)$ and $p(Y|X;\theta)$ for optimizing θ, which increases the memory requirements for GPU or CPU computation. 3) Previous research [Tu et al., 2016, Mi et al., 2016, Suzuki and Nagata, 2017, Yang et al., 2018, Kiyono et al., 2018, Meister et al., 2020] has suggested that the objective function itself may cause previous decoding problems. In particular, according to Tu et al. [2016] and Mi et al. [2016], the global attention-based seq2seq model did not explicitly consider which source-side tokens had already been covered in past attentions, resulting in repeated attention to translated source tokens and generating degenerated text with redundant tokens. To address decoding issues, based on Equation (1.0.1), researchers have proposed different objective functions to score each pair of (X, Y), denoted as **scoring-based methods**. Tu et al. [2016] and Mi et al. [2016] estimated coverage normalization when computing $p(y_j|Y_{<j}, X;\theta)$ in Equation (1.2.2) and Equation (1.2.4) to consider the coverage of the attention distribution and prevent generating redundant outputs. However, this method is not applicable to seq2seq models with multiple attentions, such as the Transformer (details explained in Chapter 2). To address the redundant generation without modifying the standard attention mechanism, a simple idea is to adopt multi-task learning [Sener and Koltun, 2018] for models to simultaneously learn multiple tasks: an original task that predicts Y from X, and M

added tasks aim to different problems. Let λ_m denote the weight for the t^{th} added task, $R_m(\cdot)$ the m^{th} added task-specific loss function. Based on Equation (1.2.1), when using multi-task learning to score (X, Y), the model parameter θ can be updated as follows:

$$\theta^* := \arg\max_{\theta} \sum_{(X,Y)\in\mathcal{D}} \left(\log p(Y|X;\theta) + \sum_{m=1}^{M} \lambda_m \cdot R_m(Y|X;\theta) \right). \tag{1.3.1}$$

For instance, Kiyono et al. [2018] integrated a frequency constraint $R(Y|X;\theta)$ on input tokens to control redundancy in text summarization. However, their approach assumes that input sentences contain more tokens than the output, which limits its applicability to other tasks. Moreover, designing different computations for $p(y_j|Y_{<j}, X;\theta)$ is straightforward and intuitive, but the lack of factorization for future steps during inference still results in exposure bias. 4) To address this issue, similar to Equation (1.0.2), Sountsov and Sarawagi [2016] modeled a globally conditional probability $p(Y|X;\theta)$ as:

$$p(Y|X;\theta) = \frac{S(X,Y;\theta)}{\sum_{Y'\in\mathcal{Y}} S(X,Y';\theta)}, \tag{1.3.2}$$

where $S(X,Y;\theta)$ computes the score of (X,Y) with factorization for future steps, instead of using $p(y_j|Y_{<j},X;\theta)$. Given a set $\mathcal{Y}$ of possible outputs, the item $\sum_{Y'\in\mathcal{Y}} S(X,Y';\theta)$ acts as a normalizer to normailize $S(X,Y;\theta)$ for $Y \in \mathcal{Y}$. However, due to the exponentially growing computational complexity, exhaustive search for generating $\mathcal{Y}$ is practically impossible. To reduce computation while taking a glimpse of the possible future steps, they employed a re-ranking strategy, denoted as **re-ranking-based methods**, that first generates a set $\mathcal{Y}_k$ of k-best candidates using a base model, such as Equation (1.2.4), and then re-scores these candidates whose future steps are visible using the proposed score function. Li and Jurafsky [2016] also used a similar re-ranking strategy to enhance the diversity and quality of translation. Their function $S(X,Y;\theta)$ considers $p(X|Y)$, $p(Y)$, and length of targets with factorization for future steps, in addition to computing the sequential probability $p(y_j|Y_{<j},X;\theta)$. As the two work relied on LSTM structures for re-ranking, it remains unclear how well the re-ranking approach with factorization for future steps would perform on the current dominant model structure, Transformer.

1.4 Goals and Organization

This thesis presents two **re-ranking-based** methods to alleviate two major problems faced by discriminative models, **text degeneration** and **exposure bias**. The goal of this study is to propose general ideas that can be easily implemented in natural language processing tasks, such as grammatical error correction and discourse parsing, to enhance the quality of predictions without making complex modifications on the dominant discriminative model structure or using excessive computational resources. The proposed methods also have the potential to relieve the **search error** by carefully considering the probability distribution, which may reject the unsuitable appearances of <eos>.

The proposals in this study are based on the Transformer architecture, which is one of the most widely used architectures in the field of natural language processing, as mentioned previously. Chapter 2 provides an overview of the Transformer architecture and other essential information related to the proposed model structures, including seq2seq models, language models, and masked language modeling [Devlin et al., 2019].

To alleviate **text degeneration** and improve prediction quality by countering the **exposure bias** problem, the first part of this study proposes a function $S(X, Y; \theta)$ that considers the joint probability $p(X, Y)$ with unlikelihood training. In inference, it follows Sountsov and Sarawagi [2016] and Li and Jurafsky [2016] to implement the re-ranking strategy to make future steps visible for each possible $Y \in \mathcal{Y}_\alpha$. This setting is introduced in Chapter 3 and denoted as a *language model-based reranker* (LMR) that uses information from target labels by treating the labels as an input and enhancing label representations by embedding descriptions for each label, which allows LMR to effectively use a pre-trained language model. The LMR is evaluated on the RST-DT [Carlson et al., 2002] dataset and achieves state-of-the-art performance in discourse segmentation and sentence-level discourse parsing tasks.

The second method in Chapter 4 presents an extension of LMR, a *bidirectional Transformer reranker* (BTR), which can be easily applied to pre-trained seq2seq models. The BTR preserves the seq2seq-style Transformer architecture while incorporating a BERT-style self-attention mechanism [Devlin et al., 2019] in the decoder to compute the probability of each target token. During inference, the BTR compares the re-ranked top-1 result with the original top-1 one using an acceptance threshold β to give the final results. Experimental results on three publicly available GEC datasets demonstrate the effectiveness of BTR in overcoming exposure bias, with significant improvements in $F_{0.5}$ scores compared to the fine-tuned T5-base model.

Chapter 5 summarizes the main contributions and findings of this study, highlighting the effectiveness of the proposed **re-ranking-based methods** in addressing the **exposure bias** and **text degeneration** problems in natural language processing tasks. It suggests that future research could explore combining discriminative models with representations of target sequences, and focus on more robust models to domain shifts and other challenges in real-world applications, providing valuable insights into the potential directions for advancing predictions from the state-of-the-art discriminative models in natural language processing.

Chapter 2

Preliminary

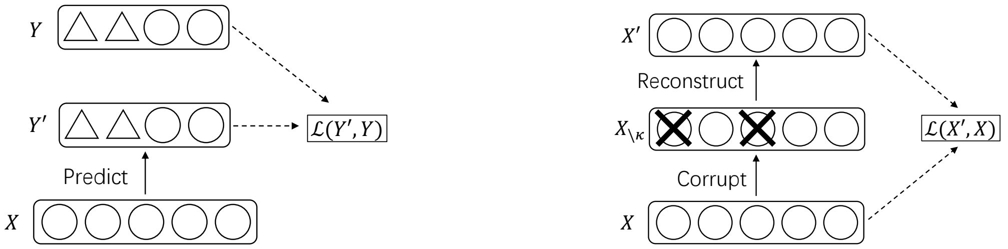
This chapter introduces the fundamental concepts of language models and encoder-decoder architectures, which are extensively employed for decomposing probability and decoding time-specific probability distribution, respectively, throughout this thesis. Specifically, it focuses on the strategy for decomposing probabilities, which involves a comparison between masked language modeling and auto-regressive language modeling. However, detailed information on neural networks, such as self-attention and feed-forward mechanisms, are presented in the references. Please note that in the following sections, the end token <eos> is used to indicate the end of each sequance, denoted as x_I and y_J for X and Y, respectively.

2.1 Supervised Learning

Given a set of input observations, each associated with some correct outputs, the goal of the supervised learning algorithm is to learn how to map from a new observation to a correct output. Natural language generation is such a task using supervised learning to generate an appropriate text Y from a source text X, as shown in Figure 2.1(a). Formally, taking an input X and a fixed target language vocabulary $\mathcal{V}_t$, natural language generation task returns a sequence consisting of tokens belonging to $\mathcal{V}_t$. Natural language processing tasks also involve classification. For example, in discourse segementation, the objective is to assign label to each source token to indicate whether the token represents the start of an elementary discourse unit or not, when given the input X and a label set $\mathcal{V}_t = \{0, 1\}$. This task is described in detail in Chapter 3. In the supervised situation, we use a training set $\mathcal{D}$ of N source texts that have been hand-labeled with targets as: $\{(X_1, Y_1), \ldots, (X_N, Y_N)\}$. Our goal is to learn a model f that is capable of mapping from a new text X to its correct target Y [Jurafsky, 2000].

2.1.1 Self-supervised Learning

While supervised learning is limited by the scarcity of hand-labeled data, self-supervised learning that leverages the abundance of unlabeled data, readily available at web-scale, has gained attention as a promising approach in natural language processing [Balestriero



(a) Supervised learning

(b) Self-supervised learning

Figure 2.1: An example of supervised learning (a) and self-supervised learning (b), where Y' represents the predicted text derived from X, X' represents the reconstructed text derived from $X_{\backslash\kappa}$. The loss $\mathcal{L}(Y', Y)$ quantifies the dissimilarity between the true text Y and the predicted text Y', and the model is optimized by minimizing the loss to reduce this discrepancy.

et al., 2023]. As presented in Figure 2.1(b), self-supervised learning performs as a denoising object: given an unlabeled input text X, a fraction of its elements are corrupted, typically discarded or replaced based on a noise strategy, models are then trained to recover these corruptions by only using the corrupted input $X_{\backslash\kappa}$, where κ denotes the set of corrupted positions in the input X. This process enables the models to learn descriptive and meaningful representations (§2.2), making self-supervised learning serves as a pretext task to generate high-quality representations that benefit various downstream tasks in natural language processing, including machine translation, text summarization, and text generation [Raffel et al., 2020, Balestriero et al., 2023].

2.2 Representations

In natural language processing tasks, tokens $x_i \in X$ are typically represented as $|\mathcal{V}_s|$-dimensional vectors in the form of $[0, 1]^{|\mathcal{V}_s|}$, where $|\mathcal{V}_s|$ is the size of the source language vocabulary $\mathcal{V}_s$. While this atomic unit is simple and natural for representing x_i as an index in a vocabulary, it ignores the precise syntactic and semantic relationships between tokens, resulting in suboptimal generation performance. Moreover, many natural languages have vocabularies with tens of thousands of words, making it challenging to train an effective model with limited parameters [Mikolov et al., 2013a,b].

To address these limitations, distributed word representations, known as word embeddings, are utilized as under-complete representations for x. These word embeddings are dense, low-dimensional, and real-valued vectors $e \in \mathbb{R}^{d_e}$ with each dimension captures a latent feature of the word (or token), where $d_e < |\mathcal{V}_s|$ and $\mathbb{R}$ denotes the real coordinate space [Turian et al., 2010]. To store and access these word embeddings efficiently, a lookup table $\mathcal{E} \in R^{d_e \times |\mathcal{V}_s|}$ is constructed, where each row represents the embedding vector for a particular token in the vocabulary. Another commonly mentioned representation is hidden representation, referred to as hidden states, which is learned from hidden layers of a neural network. These hidden states, denoted as $h \in \mathbb{R}^{d_h}$, are also dense, low-dimensional, and real-valued vectors without directly correspond to the desired output for a specific task, where $d_h < |\mathcal{V}_s|$ [Goodfellow et al., 2016]. Naturally, to enable the system quickly achieve effective performance on tasks, we expect the extracted hidden state h_i from x_i to preserve a significant amount of information about the input X [Vincent et al., 2010]. To illustrate word embeddings and hidden states, let's consider an example of using a neural network language model [Bengio et al., 2003] to estimate the conditional probability of the next word based on the previous context. The network computes the probability distribution $p(\cdot|X_{<i})$ over the source vocabulary $\mathcal{V}_s$ at the i^{th} timestep as follows [Derby et al., 2020]:

$$\begin{aligned} e_i &= \mathcal{E}x_{i-1} \\ \tilde{h}_i &= g(e_i, \tilde{h}_{i-1}) \\ p(\cdot|X_{<i}) &= \text{softmax}(W\tilde{h}_i + b). \end{aligned} \tag{2.2.1}$$

Here, W is a weight matrix, b is a bias term, g represents one or multiple temporally compatible layers such as LSTM or Transformer, and softmax denotes the softmax function (§2.4). Given the one-hot vector x_{i-1}, we first extract its embedding e_i from the lookup table $\mathcal{E}$. Then, the function g takes e_i and the previous hidden state $\tilde{h}_{i-1}$ to produce a new hidden state $\tilde{h}_i$. After passing $\tilde{h}_i$ through a fully-connected layer, we obtain the probability distribution $p(\cdot|X_{<i})$ for generating x_i. We compute the cross-entropy loss (§2.6.1) between the predicted distribution and the actual distribution and minimize the loss using gradient descent. Through this process, the embedding matrix $\mathcal{E}$ can be optimized as a by-product of the main task [Mikolov et al., 2013b, Devlin et al., 2019]. The representations discussed in this thesis refer to the distributed word representations.

2.3 Transfer Learning

Data dependence is one of the most serious problems in learning deep features for a neural network model f. This is because such models consist of a large number of parameters and require a substantial amount of data to effectively capture the underlying patterns in supervised learning tasks. However, in reality, data collection can be complex and expensive. To address this issue, transfer learning assumes that the training data does not need to be independent and identically distributed from the test data. Under this assumption, the training process is divided into two steps: the first step involves pre-training the model parameters θ on upstream tasks, while the second step focuses on fine-tuning the optimized model parameters θ^* to learn a new set of parameters ϕ for downstream tasks [Tan et al., 2018, Zhang and Hashimoto, 2021]. The current dominant approach in pre-training methods leverages the abundance of unlabeled data using self-supervised learning, as demonstrated in pre-trained models like BERT, GPT2, BART, and T5. After pre-training, the optimized model is fine-tuned on specific downstream tasks, such as machine translation and text summarization. Here, we present an example of fine-tuning a pre-trained model $p(Y|X;\theta^*)$ for the grammatical error correction task. Given a corpus $\mathcal{D}' = \{(X_n, Y_n)\}_{n=1}^{N'}$, where X represents an ungrammatical sentence and Y represents the corresponding grammatical sentence, during fine-tuning, we modify the optimized parameter θ^* to ϕ for the pre-trained model using the following objective:

$$\phi^* := \arg\max_{\phi} \prod_{(X,Y)\in\mathcal{D}'} p(Y|X;\phi,\theta^*) = \arg\max_{\phi} \sum_{(X,Y)\in\mathcal{D}'} \log p(Y|X;\phi,\theta^*), \qquad (2.3.1)$$

2.4 Softmax Function

When x_i is the m^{th} element of $\mathcal{V}_s$, the softmax function [Bridle, 1990] in Equation (2.2.1) could be expressed as [Bishop, 1995]:

$$\begin{aligned} p(\cdot|X_{<i}) &= \text{softmax}(W\tilde{h}_i + b), \\ \text{where } p(x_i = m|X_{<i}) &= \frac{\exp(\mathbf{w}_m\tilde{h}_i + b_m)}{\sum_{m'=1}^{|\mathcal{V}_s|}\exp(\mathbf{w}_{m'}\tilde{h}_i + b_{m'})}, \end{aligned} \qquad (2.4.1)$$

$\mathbf{w}_m \in \mathcal{R}^{|d_h|}$ denotes the m^{th} row of $W \in \mathcal{R}^{|\mathcal{V}_s \times d_h|}$, b_m denotes the m^{th} element of $b \in \mathcal{R}^{|\mathcal{V}_s|}$, and $\exp(\cdot)$ refers to the exponential function. The softmax function is widely used in neural networks as a normalizer to produce a probability distribution. It non-linearly normalizes a set of numbers to the interval $[0, 1]$ while ensuring that all components sum up to 1, thanks to the exponential function.

The reason why we select the exponential function for the softmax could be tracked back to the work of Ludwig Boltzmann in 1868 [Boltzmann, 1868]. Boltzmann first formulated the Boltzmann distribution, which is often used to describe the distribution of particles in a system, such as atoms or molecules. In this distribution, Boltzmann assumed that each particle has a finite number of velocities and, consequently, a finite number of values for its kinetic energy. Let M be the number of available states accessible to a system, o_m be the number of particles in state m, $O = \sum_{m=1}^{M} o_m$ be the total number of particles in this system, and μ_m be the kinetic energy of a particle in state m. For a system consisting of many particles, the probability of state m represents the likelihood of randomly selecting a particle that is in state m. This probability is determined by the number of particles in state m divided by the total number of particles in the system as:

$$p(m) = \frac{o_m}{O}, \qquad (2.4.2)$$

o_m is estimated as described below.

Let $\Omega = \sum_{m=1}^{M} o_m\mu_m$ be the total energy of a system. By considering a group of constants (M, O, Ω), the total number of available systems, denoted as δ, is given by $\frac{O!}{\prod_{m=1}^{M} o_m!}$. Boltzmann established a relationship between the thermodynamic entropy τ and δ, expressed as $\tau = \rho \ln \delta$, where ρ is the Boltzmann constant. And thus, to construct a “best” system, we could maximize τ according to the maximum entropy theory. Considering previous constrains, we could form a Lagrangian function as:

$$\mathcal{L}_{\text{energy}} = \ln O! - \sum_{m=1}^{M} \ln o_m! - \lambda \left((\sum_{m=1}^{M} o_m) - O \right) - \gamma \left((\sum_{m=1}^{M} o_m \mu_m) - \Omega \right), \tag{2.4.3}$$

where λ and γ are Lagrange multipliers. To maximize the entropy τ, all partial derivatives of this equation should be zero. Using the Stirling’s approximation, $\ln x! \approx x \ln x - x$, o_m can be derived as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{energy}}}{\partial o_m} \approx \frac{\partial}{\partial o_m}(-o_m \ln o_m + o_m - \lambda o_m - \gamma o_m \mu_m) &= 0 \\ -\ln o_m - \lambda - \gamma \mu_m &= 0 \\ o_m &= \exp(-\lambda - \gamma \mu_m). \end{aligned} \tag{2.4.4}$$

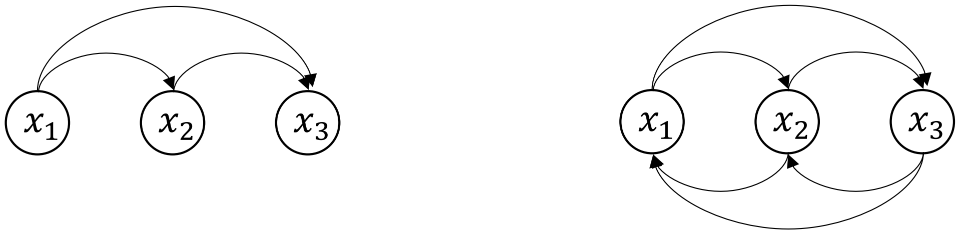
Consequently, Equation (2.4.2) can be rewritten as:

$$p(m) = \frac{\exp(-\lambda - \gamma \mu_m)}{\sum_{m'=1}^{M} \exp(-\lambda - \gamma \mu_{m'})} = \frac{\exp(-\gamma \mu_m)}{\sum_{m'=1}^{M} \exp(-\gamma \mu_{m'})}, \tag{2.4.5}$$

where the denominator $\sum_{m'=1}^{M} \exp(-\gamma \mu_{m'})$ is referred to as the partition function [Boltzmann, 1868, Atkins et al., 2014, Sharp and Matschinsky, 2015]. Assuming γ is an arbitrary positive constant, resulting in the states with lower energy will always have a higher probability of being occupied. For neural networks, μ can be defined as a function $\mu(X, Y, \theta)$ to measure the “goodness (or badness)” of each possible configuration of Y and X. The output number of μ can be interpreted as the degree of compatibility between the values of Y and X. During training, to find optimal model parameter θ^*, we can associate low energies to “desired” paired of data (i.e. highly compatible configurations of the variables), and high energies to “undesired” configurations (i.e. highly incompatible configurations of the variables) [LeCun and Huang, 2005, LeCun et al., 2006].

2.5 Denoising Autoencoder

To guarantee the extraction of useful features, Vincent et al. [2010] assumed that a good hidden state should exhibit stability and robustness when faced with corrupted input, while being capable of capturing the structure of the corrupted input to facilitate the recovery of the corresponding clean input, akin to a denoising task. Building upon this assumption, Vincent et al. [2010] proposed the denoising autoencoder algorithm to train a stochastic operator $p(X|X_{\setminus\kappa})$ to reconstruct the input X from its corrupted version $X_{\setminus\kappa}$ through self-supervised learning. Let $q(X_{\setminus\kappa}|X)$ denote a stochastic operator, $\mathcal{B}(\mu)$ the Bernoulli distribution with mean μ. The initial input X could be corrupted into $X_{\setminus\kappa}$ by means of a stochastic mapping $X_{\setminus\kappa} \sim q(X_{\setminus\kappa}|X)$. Please note that, each time a training



(a) acyclic **(b)** cyclic

Figure 2.2: Directed graphs in the auto-regressive language model (a) and in the masked language model (b).

example X is presented, a different corrupted version $X_{\setminus\kappa}$ of it is generated according to $q(X_{\setminus\kappa}|X)$. For optimization, the loss $\mathcal{L}(X', X)$ estimates an associated reconstruction error $\mathcal{L}(X', X)$ between the reconstructed input X' and the original input X. If x_i is binary, that is $x_i \in [0, 1]^{|\mathcal{V}_s|}$, $\mathcal{L}(X', X)$ could be the cross-entropy loss.

2.6 Language Model

Reconstructing the original input sequence X from its corrupted version $X_{\setminus\kappa}$ can be achieved using a language model. A language model represents the probability distribution over sequences of tokens in natural languages. For a sequence $X = (x_1, \ldots, x_I)$, the language model assigns the joint probability $p(x_1, \ldots, x_I)$ to the marginal probability $p(X)$ [Goodfellow et al., 2016].

2.6.1 Auto-regressive Language Model

A commonly used type of language model is auto-regressive language model, also called as causal language model. It utilizes the principles of auto-regressive modeling to predict the next token based on the previously generated tokens [Hyndman and Athanasopoulos, 2018], which defines an acyclic directed graph (or Bayes net) [Murphy, 2012] as Figure 2.2(a) shows [Yamakoshi et al., 2022]. Then, the marginal probability $p(X)$ can be factorized using the chain rule as the product of conditional probabilities [Radford et al., 2019] as follows:

$$p(X) = p(x_1, \ldots, x_I) = \prod_{i=1}^{I} p(x_i|X_{<i}). \tag{2.6.1}$$

Given a corpus $\mathcal{D}_X$, the optimal parameter θ^* of the auto-regressive language model can be estimated by employing the MLE approach. This involves updating the initial parameter θ as follows:

$$\begin{aligned}\theta^* := \arg\max_{\theta} \prod_{X \in \mathcal{D}_X} p(X;\theta) &= \arg\max_{\theta} \sum_{X \in \mathcal{D}_X} \log p(X;\theta) \\ &= \arg\max_{\theta} \sum_{X \in \mathcal{D}_X} \sum_{i=1}^{I} \log p(x_i|X_{<i};\theta). \end{aligned} \tag{2.6.2}$$

Assume the probability distribution $p(\cdot|X_{<i};\theta)$ is computed by a neural network such as Equation (2.2.1). Because x_i is a binary variable, optimizing Equation (2.6.2) is equivalent to minimizing the cross-entropy loss [BRIER, 1950], that is

$$\mathcal{L}_{\text{CrossEntropy}} = -\sum_{X \in \mathcal{D}_X} \log p(X;\theta) = -\sum_{X \in \mathcal{D}_X} \left(\sum_{i=1}^{I} \sum_{v \in \mathcal{V}_s} x_{i,v} \log p(v|X_{<i};\theta) \right), \tag{2.6.3}$$

where $p(v|X_{<i};\theta)$ denotes the probability at the i^{th} time step for vocab $v \in \mathcal{V}_s$, and

$$x_{i,v} := \begin{cases} 1 & \text{if } x_i = v \\ 0 & \text{if } x_i \neq v. \end{cases} \tag{2.6.4}$$

Given the trained model with optimized θ^*, finding the most likely prediction X^* is solved using MAP to decode $p(X;\theta^*)$ as:

$$X^* = \arg\max_{X} \log p(X;\theta^*) = \arg\max_{X} \sum_{i=1}^{I} \log p(x_i|X_{<i};\theta^*), \tag{2.6.5}$$

it will continue generating tokens until emitting the end token <eos>.

Although the auto-regressive language model is optimized through self-supervised learning, it has gained popularity in text generation tasks, rather than focusing on generating high-quality representations for other downstream tasks. In particular, it is commonly used as the decoder component in encoder-decoder models, which will be discussed in Section 2.7. One of the most widely recognized auto-regressive language models is GPT-2 [Radford et al., 2019], renowned for its effectiveness in text generation tasks owing to a pre-training step. Leveraging the achievements of GPT-2, extensive development efforts over several years led in a remarkable addition to the GPT series. This refined version, known as ChatGPT, has captured worldwide attention with its impressive capabilities and advancements in conversational interactions with humans.

2.6.2 Masked Language Modeling

Different from the auto-regressive language model, the pretext task: masked language modeling (MLM), that introduced in BERT was designed to learn bidirectional representations for a given sentence X through a cyclic directed graph or dependency

Symbol	Item
X	Thank you for inviting me to your party last week .
κ	$\{3, 4, 7, 11\}$
X_κ	{for, inviting, your, last}
$X_{\setminus\kappa}$	Thank you so <M> me to your party <M> week .

Table 2.1: Example of a sentence of length 11 before and after corruption.

net [Heckerman et al., 2001, Devlin et al., 2019, Yamakoshi et al., 2022]. Figure 2.2(b) illustrates an example. During training, for $X \in \mathcal{D}_X$, BERT follows the denosing autoencoder algorithm to utilize a masking noise strategy to corrupt a fraction of elements of X, resulting in a corrupted version $X_{\setminus\kappa}$. Note that, BERT also ensures that each time a training example X is presented, a different corrupted version $X_{\setminus\kappa}$ is generated. Specifically, 15% of the elements in X are randomly selected by the training data generator. If x_i is chosen, it is replaced with (1) the mask token <M> 80% of the time, (2) a random token from $\mathcal{V}_s$ 10% of the time, and (3) the unchanged x_i 10% of the time. An example of sentence corruption is presented in Table 2.1. While using self-supervised learning, MLM aims to predict the original, uncorrupted tokens X_κ for a corrupted $X_{\setminus\kappa}$, instead of reconstructing the original input X, which is different from the denoising autoencoder. The model parameter θ is optimized by minimizing the cross-entropy loss as:

$$\begin{aligned} &- \log p(X_\kappa | X_{\setminus\kappa}; \theta) \\ \approx &- \sum_{i \in \kappa} \log p(x_i | X_{\setminus\kappa}, i; \theta) = - \sum_{i \in \kappa} \sum_{v \in \mathcal{V}_s} x_{i,v} \log p(v | X_{\setminus\kappa}, i; \theta), \end{aligned} \tag{2.6.6}$$

where $p(v|X_{\setminus\kappa}, i; \theta)$ denotes the probability at the i^{th} position of $X_{\setminus\kappa}$ for vocab $v \in \mathcal{V}_s$, and $x_{i,v}$ is computed by Equation (2.6.4) [Song et al., 2020]. Assume the probability distribution $p(\cdot|X_{\setminus\kappa}, i; \theta)$ is also computed by a neural network. Compared with $p(x_i|X_{<i}, \theta)$, $p(x_i|X_{\setminus\kappa}, i; \theta)$ utilizes both the left and right sides of the context of the masked token x_i to capture deeper relationship between tokens.

As claimed by Salazar et al. [2020], $\log p(X; \theta)$ in Equation (2.6.2) could be approximated with the pseudo-log-likelihood scores (PLL) [Besag, 1975] as follows:

$$\log p(X; \theta) \approx \mathrm{PLL}(X; \theta) := \sum_{i=1, \kappa=\{i\}}^{I} \log p(x_i | X_{\setminus\kappa}, i; \theta). \tag{2.6.7}$$

Besides, when using PLL to estimate the cross-entropy loss, the loss of $x_i|X_{\setminus\kappa}, i$ versus i from BERT is flatter than GPT-2, that uses the chain rule. Considering the candidate sentences might have different lengths, PLL is ideal for reranking.

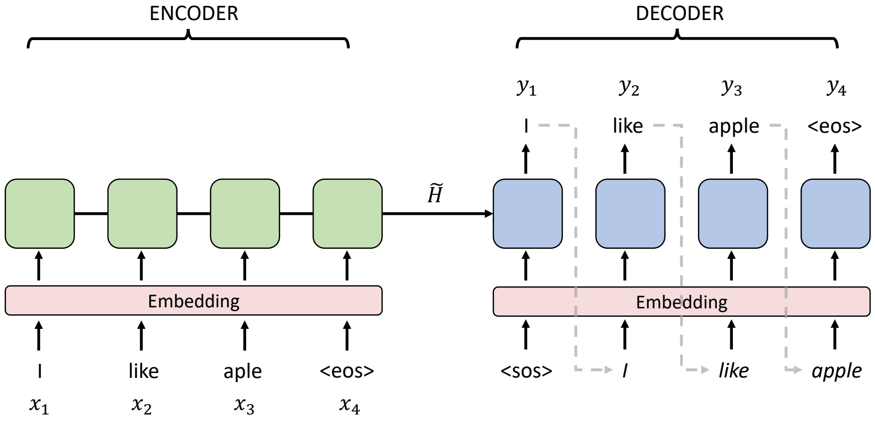


Figure 2.3: An example of using the encoder-decoder model to correct grammatical errors. The input text X =(I, like, aple, <eos>) contains a grammatical error, the word "aple". The target text Y =(I, like, apple, <eos>) represents the corrected version. In the decoder, the predicted token y_j at the j^{th} time step is utilized as an input in the $(j+1)^{\text{th}}$ time step and is highlighted in *italics*. The input sequence to the decoder begins with the special token <sos>, indicating the start of the prediction. This sequence is referred to as shifted right outputs, where each token in Y is shifted one position to the right.

2.7 Encoder-Decoder Sequence-to-Sequence Architectures

While a language model is typically designed to predict the probability distribution of each token in the given sequence, there are tasks that require to generate a new sequence with respect to the given one. In such cases, the sequence-to-sequence (seq2seq) model, also known as the encoder-decoder model, is commonly used. The seq2seq model consists of two primary components: encoder and decoder. The encoder calculates the hidden representations $\widetilde{H}$ of a source sentence X, while the decoder generates a target sentence Y from $\widetilde{H}$. An example is presented in Figure 2.3. Here, the decoder operates as an auto-regressive language model. Therefore, similar to Equation (2.6.2), the seq2seq model with parameters θ could decompose the conditional probability $p(Y|X;\theta)$ according to the chain rule, and update initialized θ as follows:

$$\begin{aligned}\theta^* := \arg\max_{\theta} \prod_{(X,Y)\in\mathcal{D}} p(Y|X;\theta) &= \arg\max_{\theta} \sum_{(X,Y)\in\mathcal{D}} \log p(Y|X;\theta) \\ &= \arg\max_{\theta} \sum_{(X,Y)\in\mathcal{D}} \sum_{j=1}^{J} \log p(y_j|Y_{<j},X;\theta). \qquad (2.7.1)\end{aligned}$$

At the j^{th} decoding step, to estimate $p(y_j|Y_{<j},X;\theta)$, the seq2seq model predicts the probability distribution $p(\cdot|Y_{<j},X;\theta)$ by

$$p(\cdot|Y_{<j},X;\theta) = \text{softmax}(W\tilde{s}_j + b), \qquad (2.7.2)$$

where $\tilde{s}_j$ is the final hidden state from the decoder. Let $p(v|Y_{<j}, X; \theta)$ denotes the predicted probability at the j^{th} decoding step for vocab $v \in \mathcal{V}_t$. Similar to the auto-regressive language model, the seq2seq model parameter θ is optimized by minimizing the cross-entropy loss as:

$$-\log p(Y|X;\theta) = -\sum_{j=1}^{J} \sum_{v \in \mathcal{V}_t} y_{j,v} \log p(v|Y_{<j}, X; \theta), \tag{2.7.3}$$

$$\text{where } y_{j,v} := \begin{cases} 1 & \text{if } y_j = v \\ 0 & \text{if } y_j \neq v. \end{cases} \tag{2.7.4}$$

Given the source X and the trained model with optimized θ^*, similar to Equation (2.6.5), finding the most likely prediction Y^* is solved as:

$$Y^* = \arg\max_{Y} \log p(Y|X;\theta^*) = \arg\max_{Y} \sum_{j=1}^{J} p(y_j|Y_{<j}, X; \theta^*), \tag{2.7.5}$$

it will continue generating tokens until emitting the end token <eos>.

2.7.1 Seq2seq-style Transformer

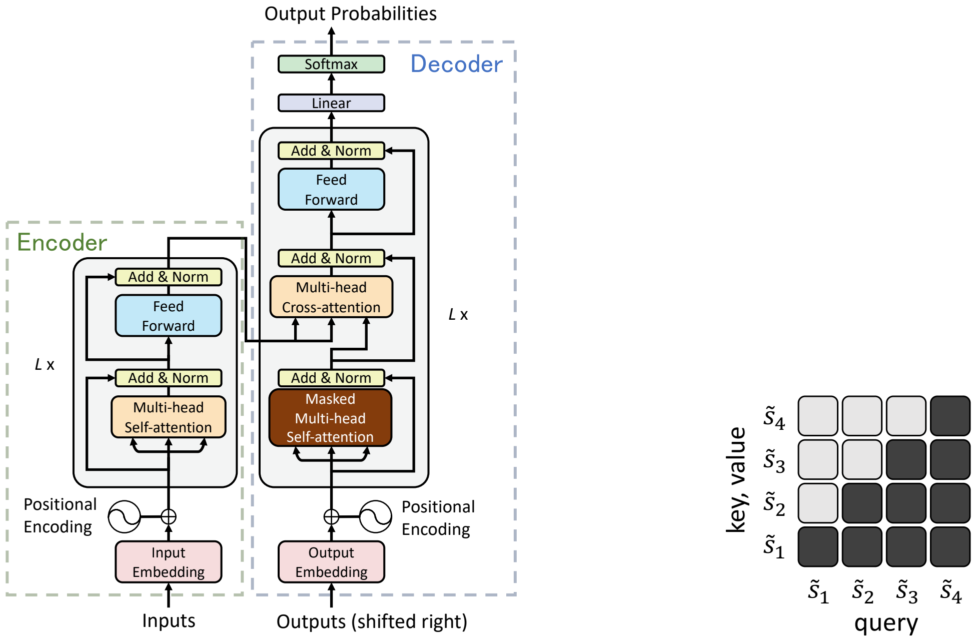
The current most popular seq2seq-style nerual network is the Transformer, whose architecture is depicted in Figure 2.4(a). Let FNN denote a feed-forward layer and $\text{Attn}(q, K, V)$ the attention layer, where q, K, and V indicate the query, key, and value, respectively. The "Add & Norm" layer is combined with the FNN or Attn layers and is not explicitly shown in the following formulas. We assume both the encoder and decoder in the Transformer architecture consist of L layers. To compute $\tilde{s}_j$, the encoder first encodes the source sentence X into its last hidden representations $\widetilde{H}^L = (\tilde{h}_1^L, \ldots, \tilde{h}_I^L)$. For each layer $\ell \in L$, the computation for the hidden state $\tilde{h}_i^\ell$ in the encoder is as follows:

$$\begin{aligned} h_i^\ell &= \text{Attn}_s(\tilde{h}_i^{\ell-1}, \widetilde{H}^{\ell-1}, \widetilde{H}^{\ell-1}) \\ \tilde{h}_i^\ell &= \text{FNN}(h_i^\ell), \end{aligned} \tag{2.7.6}$$

where $\tilde{h}_i^0$ is the word embedding of the token x_i, Attn_s indicates the self-attention layer.

Then, the decoder uses the hidden representations $\widetilde{H}^L$ to compute the final hidden state $\tilde{s}_j^L$. For $\ell \in L$, the hidden state $\tilde{s}_j^\ell$ of the ℓ^{th} layer in the decoder is computed by

$$\begin{aligned} s_j^\ell &= \text{Attn}_s(\tilde{s}_j^{\ell-1}, \widetilde{S}_{\leq j}^{\ell-1}, \widetilde{S}_{\leq j}^{\ell-1}), \\ \hat{s}_j^\ell &= \text{Attn}_c(s_j^\ell, \widetilde{H}^L, \widetilde{H}^L), \\ \tilde{s}_j^\ell &= \text{FNN}(\hat{s}_j^\ell), \end{aligned} \tag{2.7.7}$$



(a) Transformer

(b) Causal

Figure 2.4: The Transformer architecture (a) utilizes a causal masked self-attention mechanism (b) in the decoder.

where $\tilde{s}_j^0$ is the word embedding of the token y_{j-1} and $\tilde{s}_1$ is the state for the start token <sos>. $\widetilde{S}_{\leq j}^{\ell-1}$ denotes a set of hidden states $(\tilde{s}_1^{\ell-1}, \ldots, \tilde{s}_j^{\ell-1})$. Attn_c indicates the cross-attention layer. A causal attention mask can be used to compute S^ℓ in parallel, as in Figure 2.4(b).

The seq2seq-style Transformer has demonstrated excellent performance, leading to the popularity of encoder-only and decoder-only Transformer models for various tasks.

2.7.2 Encoder-only Transformer

One prominent encoder-only Transformer model is BERT, which was introduced in Section 2.6.2. Unlike auto-regressive language models, the encoder-only structure of BERT is not specifically designed for text generation tasks. Therefore, except for the cross-entropy loss, the optimization for the encoder-only Transformer model can involve mean square error or other loss functions in relation to downstream tasks.

In the case of MLM to predict the probability distribution $p(\cdot|X_{\setminus\kappa}, i; \theta)$ at each position i in the set of corrupted positions κ, similar to Equation (2.7.2), a linear transformation is applied to the hidden state $\tilde{h}_{i\setminus\kappa}^L$, followed by a softmax function as follows:

$$p(\cdot|X_{\setminus\kappa}, i; \theta) = \text{softmax}(W\tilde{h}_{i\setminus\kappa}^L + b), \quad i \in \kappa. \tag{2.7.8}$$

And for $\ell \in L$, the hidden state $\tilde{h}^{\ell}_{i\backslash\kappa}$ of the ℓ^{th} layer in the model is computed by

$$\begin{aligned} h^{\ell}_{i\backslash\kappa} &= \text{Attn}_s(\tilde{h}^{\ell-1}_{i\backslash\kappa}, \widetilde{H}^{\ell-1}_{\backslash\kappa}, \widetilde{H}^{\ell-1}_{\backslash\kappa}) \\ \tilde{h}^{\ell}_{i\backslash\kappa} &= \text{FNN}(h^{\ell}_{i\backslash\kappa}) \end{aligned} \tag{2.7.9}$$

where $\tilde{h}^{0}_{i\backslash\kappa}$ is the embedding of the i^{th} token in $X_{\backslash\kappa}$ and $\widetilde{H}^{\ell-1}_{\backslash\kappa} = (\tilde{h}^{\ell-1}_{1\backslash\kappa}, \ldots, \tilde{h}^{\ell-1}_{I\backslash\kappa})$ denotes a set of hidden states for $X_{\backslash\kappa}$.

2.7.3 Decoder-only Transformer

The decoder-only Transformer architecture functions as an auto-regressive language model. For each layer $\ell \in L$, the hidden state $\tilde{s}^{\ell}_{i}$ is computed by excluding the cross-attention layer Attn_c from Equation (2.7.7) as follows:

$$\begin{aligned} s^{\ell}_{i} &= \text{Attn}_s(\tilde{s}^{\ell-1}_{i}, \widetilde{S}^{\ell-1}_{\leq i}, \widetilde{S}^{\ell-1}_{\leq i}), \\ \tilde{s}^{\ell}_{i} &= \text{FNN}(s^{\ell}_{i}), \end{aligned} \tag{2.7.10}$$

where $\tilde{s}^{0}_{i}$ is the word embedding of the token x_{i-1} and $\tilde{x}_1$ is the state for the start token <sos>. The main difference between Equation (2.7.9) and Equation (2.7.10) lies in the usage of hidden states in the self-attention layer Attn_s. The encoder-only model utilizes both left and right side representations of the the i^{th} token in $X_{\backslash\kappa}$ to compute $h_{i\backslash\kappa}$, while the decoder-only model only utilizes the left side representations of x_i to compute s_i.

As mentioned in Section 2.6.1, the parameter θ of the auto-regressive language model is optimized by minimizing the cross-entropy loss. Similar to Equation (2.7.2), the probability distribution $p(\cdot|X_{<i};\theta)$ can be estimated from $\tilde{s}^{L}_{i}$ using the softmax function:

$$p(\cdot|X_{<i};\theta) = \text{softmax}(W\tilde{s}^{L}_{i} + b). \tag{2.7.11}$$

2.8 Contrastive Learning

As mentioned in Section 2.2, a good representation should accurately preserve the syntactic and semantic relationships between tokens. Based on this assumption, Hadsell et al. [2006] anticipated that a good representation would ensure that “similar” points in a high-dimensional space remain mapped to nearby points in a lower-dimensional manifold. Considering using MLE alone for this assumption only reduces the distance among similar points, Hadsell et al. [2006] proposed contrastive learning to pull neighbors (similar points) together while pushing non-neighbors (dissimilar points) apart.

2.8.1 Unlikelihood Training

Based on the principles of MLE and contrastive learning, Welleck et al. [2020] proposed the unlikelihood training to reduce the predicted probability for specific tokens, referred to as negative samples. When decomposing the marginal probability $p(X)$ with an auto-regressive language model, at the i^{th} time step, the target token $x_i \in X$ is also denoted as the positive sample, while the corresponding negative samples consist of a set of candidate tokens $C^i = \{c^i_1, \ldots, c^i_M\}$, where each candidate token $c^i_m \in \mathcal{V}_s$ and $c^i_m \neq x_i$. The unlikelihood loss for the i^{th} time step is defined as:

$$\mathcal{L}^i_{\text{unlikelihood}} = -\sum_{c \in C^i} \log\left(1 - p(c|X_{<i};\theta)\right), \tag{2.8.1}$$

where the unlikelihood loss decreases as the probability $p(c|X_{<i};\theta)$ decreases. Then, Welleck et al. [2020] combined the unlikelihood loss with the likelihood loss to augment the maximum likelihood training in Equation (2.6.2) as:

$$\theta^* := \arg\max_{\theta} \sum_{X \in \mathcal{D}_X} \sum_{i=1}^{I} \underbrace{\log p(x_i|X_{<i};\theta)}_{\text{likelihood}} + \underbrace{\lambda \sum_{c \in C^i} \log\left(1 - p(c|X_{<i};\theta)\right)}_{\text{unlikelihood}}, \tag{2.8.2}$$

where λ represents a scaling factor that balances the contributions of likelihood and unlikelihood. Maximizing the likelihood term increases $p(x_i|X_{<i};\theta)$ while decreasing $p(c|X_{<i};\theta)$ and the unlikelihood term. In other words, this estimation aims to maximize likelihood while minimizing unlikelihood.

The combination of likelihood and unlikelihood training has demonstrated its effectiveness in improving model performance in various natural language processing tasks, such as response generation [Song et al., 2021], natural language inference [Ding et al., 2020], and text generation [Welleck et al., 2020].

Chapter 3

Language Model-based Reranker

Textual coherence is essential for writing a natural language text that is comprehensible to readers. To recognize the coherent structure of a natural language text, Rhetorical Structure Theory (RST) is applied to describe an internal discourse structure for the text as a constituent tree [Mann and Thompson, 1988]. A discourse tree in RST consists of elementary discourse units (EDUs), spans that describe recursive connections between EDUs, and nuclearity and relation labels that describe relationships for each connection. Figure 3.1 (a) shows an example RST discourse tree. A span including one or more EDUs is a node of the tree. Given two adjacent non-overlapping spans, their nuclearity can be either *nucleus* or *satellite*, denoted by N and S, where the *nucleus* represents a more salient or essential piece of information than the *satellite*. Furthermore, a relation label, such as *Attribution* and *Elaboration*, is used to describe the relation between the given spans [Mann and Thompson, 1988, Carlson and Marcu, 2001]. To build such trees, RST parsing consists of discourse segmentation, a task to detect EDU boundaries in a given text, and discourse parsing, a task to link spans for detected EDUs.

This chapter focus on discourse segmentation and sentence-level discourse parsing, which are indispensable in RST parsing [Joty et al., 2013, Feng and Hirst, 2014a, Joty et al., 2015, Wang et al., 2017, Kobayashi et al., 2020] and are applicable to many downstream tasks, such as machine translation [Guzmán et al., 2014, Joty et al., 2017] and sentence compression [Sporleder and Lapata, 2005]. In discourse segmentation, Carlson et al. [2001] proposed a method for using lexical information and syntactic parsing results. Many researchers [Fisher and Roark, 2007, Xuan Bach et al., 2012, Feng and Hirst, 2014b] utilized these clues as features in a classifier although automatic parsing errors degraded segmentation performance. To avoid this problem, Wang et al. [2018c] used BiLSTM-CRF [Huang et al., 2015] to handle an input without these clues in an end-to-end manner. Lin et al. [2019] jointly performed discourse segmentation and sentence-level discourse parsing in their pointer-network-based model. They also introduced multi-task learning for both tasks and reported the state-of-the-art results for discourse segmentation and sentence-level discourse parsing in terms of F_1 scores. Despite these achievements, there is still room for improvement for both tasks owing to the scarcity of labeled data. It is important to extract more potential information from the current dataset for further performance improvement.

Under this motivation, this chapter proposes a *language model-based reranker* (LMR)

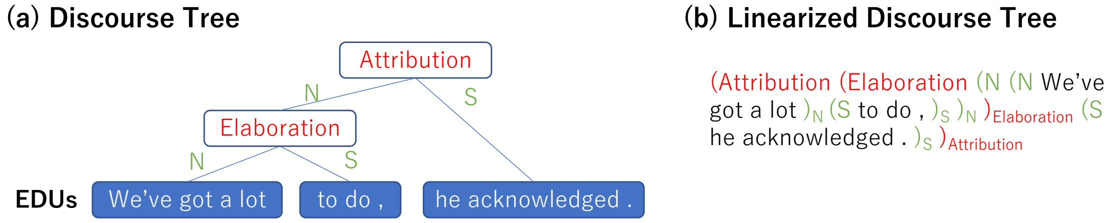

(a) Discourse Tree
Attribution
N
S
Elaboration
N
S
EDUs
We've got a lot
to do ,
he acknowledged .
(b) Linearized Discourse Tree
(Attribution (Elaboration (N (N We've got a lot)N (S to do ,)S)N)Elaboration (S he acknowledged .)S)Attribution

Figure 3.1: An example discourse tree structure.

for both discourse segmentation and sentence-level discouse parsing. The LMR can jointly predict text and label probabilities by treating a text and labels as a single sequence, like Figure 3.1 (b). Therefore, different from conventional methods, the LMR can use more information from labels by treating the labels as an input. Furthermore, the LMR can enhance label representations by embedding descriptions of each label defined in the annotation manual [Carlson and Marcu, 2001], that allows us to use a pre-trained language model such as MPNet [Song et al., 2020] effectively, because we can already have the representations for labels, that were unseen in the pre-training step.

Experimental results on the RST-DT dataset [Carlson et al., 2002] show that the LMR can achieve the state-of-the-art scores in both discourse segmentation and sentence-level discourse parsing. The LMR utilizing the proposed enhanced label embeddings achieves the best F_1 score of 96.72 in discourse segmentation. Furthermore, in sentence-level discourse parsing, when utilizing the enhanced relation label embeddings, the LMR achieves the best relation F_1 scores of 84.69 with gold EDU boundaries and 81.18 with automatically segmented boundaries, respectively.

3.1 Related Work on Discourse Segmentation and Parsing

Discourse segmentation is a fundamental task for building an RST discourse tree from a text. Carlson et al. [2001] proposed a method for using lexical information and syntactic parsing results for detecting EDU boundaries in a sentence. Fisher and Roark [2007], Xuan Bach et al. [2012], Feng and Hirst [2014b] utilized these clues as features in a classifier, while Wang et al. [2018c] utilized BiLSTM-CRF in an end-to-end manner to avoid performance degradation caused by syntactic parsing errors.

Sentence-level discourse parsing is also an important task for parsing an RST discourse tree, as used in many RST parsers [Joty et al., 2013, Feng and Hirst, 2014a, Joty et al., 2015, Wang et al., 2017, Kobayashi et al., 2020]. Recently, Lin et al. [2019] tried to jointly perform discourse segmentation and sentence-level discourse parsing with pointer-networks and achieved the state-of-the-art F_1 scores in both discourse segmentation and sentence-level discourse parsing.

In spite of the performance improvement of these models, a restricted number of labeled RST discourse trees is still a problem. In the discourse segmentation and parsing tasks, most prior work is on the basis of discriminative models, which learn mapping from input texts to predicted labels. Thus, there still remains room for improving model performance by considering mapping from predictable labels to input texts to exploit more label information. To consider such information in a model, Mabona et al. [2019] introduced a generative model-based parser, RNNG [Dyer et al., 2016], to document-level RST discourse parsing. Different from the proposed LMR, this model unidirectionally predicts action sequences as an auto-regressive language model.

This chapter models the LMR for the discourse segmentation and sentence-level discourse parsing tasks. The LMR utilizes a BERT-style bidirectional encoder-only Transformer to avoid prediction bias caused by using different decoding directions. Because the LMR is on the basis of generative models, it can jointly consider an input text and its predictable labels, and map the embeddings of both input tokens and labels onto the same space. Owing to this characteristic, the LMR can effectively use the label information through constructing label embeddings from the description of a label definition [Carlson and Marcu, 2001]. Furthermore, recent strong pre-trained models such as MPNet are available for any input tokens in the LMR.

3.2 Learning Rhetorical Structure with Discriminative Models

The LMR reranks the results from a conventional discourse segmenter and parser, which can be constructed as discriminative models. This section explains these base models and introduces related mathematical notations.

3.2.1 Discourse Segmenter

In discourse segmentation, given an input text $X = (x_1, \ldots, x_I)$, where x_i is a word, a segmenter detects EDUs $Y = (y_1, \ldots, y_J)$ from X as a supervised learning task. Because there is no overlap or gap between EDUs, discourse segmentation can be considered as a kind of sequential labeling task, which assigns labels $L = (\ell_1, \ldots, \ell_I)$, where each $\ell_i \in \{0, 1\}$ indicates whether the word is the start of an EDU or not. By using a discriminative model, such as BiLSTM-CRF and pointer-networks [Lin et al., 2019], the probability of predicting EDUs from X can be $p(L|X)$ or $p(Y|X)$. Because of its simple structure and extensibility, this chapter follows Wang et al. [2018c] to choose BiLSTM-CRF as the base model for discourse segmentation. In BiLSTM-CRF, $p(L|X)$ is formulated through a CRF layer over all possible label sequences as follows:

$$p(L|X) = \frac{\prod_{i=1}^{I} \psi_i(\ell_i, \ell_{i-1}, \tilde{h}_i)}{\sum_{L' \in \mathcal{Y}} \prod_{i=1}^{I} \psi_i(\ell_i{'}, \ell'_{i-1}, \tilde{h}_i)}, \quad (3.2.1)$$

where $\psi_i(\ell_i, \ell_{i-1}, \tilde{h}_i) = \exp(B_{l_{i-1},l_i} + W\tilde{h}_i)$ is the potential function consisting of two parts: the parameter B_{ℓ_{i-1},ℓ_i} represents the transition probability from label ℓ_{i-1} to label ℓ_i, and $W\tilde{h}_i$ represents the emission score, indicating how likely the label is ℓ_i at the i^{th} time step given the input x_i. Here, $\tilde{h}_i$ is the final hidden state at the i^{th} time step, and $\mathcal{Y}$ is the set of possible label sequences. Let LSTM denote the LSTM layer, Concat be the concatenation layer, and e_i be the word embedding of x_i. The hidden state $\tilde{h}_i$ is computed by using the bidirectional LSTM [Schuster and Paliwal, 1997] as follows:

$$\begin{aligned} \tilde{h}_i^{(1)} &= \mathrm{LSTM}(e_i, \tilde{h}_{i-1}^{(1)}) \\ \tilde{h}_i^{(2)} &= \mathrm{LSTM}(e_i, \tilde{h}_{i+1}^{(2)}) \\ \tilde{h}_i &= \mathrm{Concat}(\tilde{h}_i^{(1)}, \tilde{h}_i^{(2)}). \end{aligned} \tag{3.2.2}$$

Equation (3.2.2) and Equation (3.2.1) clearly present the advantages of using BiLSTM-CRF. These equations illustrate how bidirectional representations of the input X can be leveraged to compute hidden states, and how the probability of $p(L|X)$ is normalized over all possible label sequences, unlike auto-regressive methods that rely on a limited set of possible sequences obtained through heuristic search methods like beam search. However, it should be noted that the CRF layer in BiLSTM-CRF only considers local transitions between consecutive labels, which does not capture global transition probabilities. As mentioned earlier, there is still room for improvement by considering the representations of the target L to alleviate text degeneration. Building upon this observation, this thesis proposes the LMR to inherit the top-α Viterbi [Forney, 1973] results of Wang et al. [2018c], scored by Equation (3.2.1), as described in Section 3.3.

3.2.2 Discourse Parser

In discourse parsing, given an input text X and its EDUs Y, we can build a binary tree $A = (a_1, \ldots, a_{2I-1})$, where each node $a_i \in A$ has three kinds of labels: span d_i, nuclearity u_i, and relation r_i. The sequences of span D and nuclearity U can be predicted simultaneously, as in 2-stage Parser [Wang et al., 2017], or span D can be predicted in advance for labeling nuclearity U and relation R, as in pointer-networks [Lin et al., 2019] and span-based Parser [Kobayashi et al., 2020]. Because of its better performance, this chapter adopts the 2-stage Parser as the base model for sentence-level discourse parsing. 2-stage Parser extracts several features and does classification with SVMs in two stages. In the first stage, it identifies the span and nuclearity simultaneously to construct a tree based on the transition-based system with four types of actions: Shift, Reduce-NN, Reduce-NS, and Reduce-SN. In the second stage, for a given node a_i, r_i is predicted as the relation between the left and right children nodes of a_i by using features extracted from a_i and its children nodes. In spite of its limited features, it achieves the best results compared with pointer-networks and span-based Parser, as described in Section 3.4.3. Similar to the BiLSTM-CRF model discussed earlier, the 2-stage Parser is a discriminative model that does not consider the global relationships among labels. This motivates this thesis

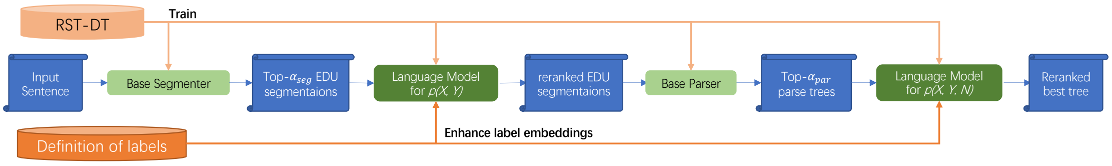

RST-DT
Train
Input Sentence
Base Segmenter
Top-α_{seg} EDU segmentaions
Language Model for $p(X, Y)$
reranked EDU segmentaions
Base Parser
Top-α_{par} parse trees
Language Model for $p(X, Y, N)$
Reranked best tree
Definition of labels
Enhance label embeddings

Figure 3.2: Overview of the reranking procedure by using the language model-based reranker (LMR).

to employ a generative model that takes into account the representations of labels to further improve the final results. Because 2-stage Parser utilizes SVMs, this chapter normalizes the action scores and inherits the top-α beam search results of 2-stage Parser for the LMR to perform sentence-level discourse parsing.

3.3 Reranking Rhetorical Structure with Masked Language Model

This section introduces the proposed generative model, LMR, that utilizes a masked and permuted language model to compute sequence probabilities for reranking in both discourse segmentation and sentence-level discourse parsing tasks. More specifically, as we mention in Section 3.4, we can utilize the LMR in three tasks, (a) discourse segmentation, (b) sentence-level discourse parsing with gold segmentation, and (c) sentence-level discourse parsing with automatic segmentation. Figure 3.2 shows the overview of using the LMR for reranking in the whole task (c). The prediction process in LMR can be summarized as follows, assuming that, in task (c), discourse segmentation and sentence-level discourse parsing are performed in a pipeline manner with models trained for tasks (a) and (b).

1. Predict the top-α_{seg} EDU segmentations $\mathcal{Y}_{\alpha_{seg}} = \{Y_1, \ldots, Y_{\alpha_{seg}}\}$ from a given sentence X using the base discourse segmenter described in Section 3.2.1.

2. Compute the joint probability $p(X,Y)$ and select the best segmentation Y^* from $\mathcal{Y}_{\alpha_{seg}}$ with a language model, as we describe below.

3. Parse and rank the top-α_{par} trees $\mathcal{A}_{\alpha_{par}} = \{A_1, \ldots, A_{\alpha_{par}}\}$ from X and the best segmentation Y^* using the base discourse parser described in Section 3.2.2.

4. Compute the joint probability $p(X, Y^*, A)$ to select the best tree A^* from $\mathcal{A}_{\alpha_{par}}$ with a language model, as we describe below.

In task (a), we apply Step 2 to predict the best segmentation after Step 1. In task (b), we skip Steps 1 and 2, and apply just Steps 3 and 4 for gold segmentation to yield the best parse tree.

(a) Sentence with EDU boundary labels

X^1 _ [EDU] _ X^2 _ [EDU] _ X^3 _ [EDU]

(b) Sentence with span labels

(Span _ (Span _ X^1 _)$_{\text{Span}}$ _ (Span _ X^2 _)$_{\text{Span}}$ _)$_{\text{Span}}$ _ (Span _ X^3 _)$_{\text{Span}}$

(c) Sentence with nuclearity labels

(N _ (N _ X^1 _)$_{\text{N}}$ _ (S _ X^2 _)$_{\text{S}}$ _)$_{\text{N}}$ _ (S _ X^3 _)$_{\text{S}}$

(d) Sentence with relation labels

(Span _ (Span _ X^1 _)$_{\text{Span}}$ _ (Elaboration _ X^2 _)$_{\text{Elaboration}}$ _)$_{\text{Span}}$ _ (Attribution _ X^3 _)$_{\text{Attribution}}$

(e) Sentence with all labels

(Span:N _ (Span:N _ X^1 _)$_{\text{Span:N}}$ _ (Elaboration:S _ X^2 _)$_{\text{Elaboration:S}}$ _)$_{\text{Span:N}}$ _ (Attribution:S _ X^3 _)$_{\text{Attribution:S}}$

Figure 3.3: Example joint representations of an input text and labels for the sentence *We've got a lot to do, he acknowledged.*, which was segmented and parsed as illustrated in Figure 3.1. X^i represents the corresponding EDU, and "_" is whitespace.

3.3.1 Tree Representations

To calculate joint probabilities for a discourse tree with a language model, we need to represent a tree as a linear form, like Figure 3.1(b). Because there are several predictable label sets in discourse segmentation and parsing tasks, as shown in Figure 3.3, we prepare linearized forms for each label set. Note that using just a raw s-expression-style tree of Figure 3.1(b) in our language model cannot work because of its much more tokens. Therefore, we transform the tree into the format depicted in Figure 3.3(e) where the nuclearity and relation labels are connected together by the colons.

In discourse segmentation, we can consider joint probability $p(X,Y)$ for a sequence with inserting a symbol, [EDU], at an EDU boundary (Figure 3.3(a)). In discourse parsing, a discourse tree is represented as a sequence with several kinds of label sets: span labels D, nuclearity labels U including span labels, and relation labels R including span and nuclearity labels (Figures 3.3(b)-(d)). To investigate the effectiveness of each label set in the reranking step, we consider $p(X,Y,D)$, $p(X,Y,U)$, and $p(X,Y,R)$ for each label set to represent $p(X,Y,A)$ in this paper. To build a sequence, we combine each label in a tree with brackets to imply the boundary for the label. For example, "(N" and ")$_{\text{N}}$" stand for the start and end of a nucleus EDU. For a node a_i of the tree, r_i describes the relation between its children nodes, leading to r_i of leaf nodes being "Null". When the child nodes of a_i are *nucleus* and *satellite*, we assign label "Span" to the *nucleus* child node of a_i and label r_i to the *satellite* child node of a_i, respectively. When the child nodes of a_i are both *nucleus*, we assign label r_i to both child nodes of a_i.

For simpler illustration, in Figure 3.1(b), we show the linearized discourse tree only with nuclearity and relation labels, because the nuclearity labels can also show span and EDU boundary labels. “Null” labels for leaf nodes are also omitted in the figure.

3.3.2 Joint Probabilities

To calculate joint probabilities in the last subsection with a language model, we consider probability $p(Z)$ for a sequence $Z = (z_1, \ldots, z_{|Z|})$, which could correspond to the probabilities for any sequential representations $p(X,Y)$, $p(X,Y,D)$, $p(X,Y,U)$, $p(X,Y,R)$, or $p(X,Y,A)$.

According to Song et al. [2020], masked and permuted language modeling (MPNet) takes the advantages of both masked language modeling in BERT and permuted language modeling in XLNet [Yang et al., 2019] while overcoming their issues. Compared with BERT and XLNet, MPNet considered more information about tokens and positions, and achieved better results for several downsteam tasks (GLUE, SQuAD, etc). Taking into account its better performance, we choose pre-trained MPNet as our language model. Because considering all possible inter-dependence between z_t is intractable, we follow the decomposition of PLL scores in the model, which was introduced in Section 2.6.2. Thus, we decompose and calculate logarithmic $p(Z)$ as follows:

$$\log p(Z;\theta) \approx \mathrm{PLL}(Z;\theta) := \sum_{t=1,\kappa=\{t\}}^{|Z|} \log p(z_t|Z_{\backslash\kappa}, t;\theta), \tag{3.3.1}$$

where $p(z_t|Z_{\backslash\kappa}, t;\theta)$ is computed by two-stream self-attention [Yang et al., 2019]. This model converts Z into continuous vectors $E = (e_1, \ldots, e_{|Z|})$ through the embedding layer. Multi-head attention layers further transform the vectors to predict each z_t in the softmax layer, as described in Section 2.7.2.

Because pre-trained MPNet does not consider EDU, span, nuclearity, and relation labels in the pre-training step, we need to construct vectors E for these labels from the pre-trained parameters to enhance the prediction performance. We describe the details of this method in the next subsection.

3.3.3 Label Embeddings

In LMR, we embed input text tokens and labels in the same vector space [Wang et al., 2018b] of the embedding layer. Under the setting, to deal with unseen labels in the pre-trained model, we compute the label embeddings by utilizing token embeddings in the pre-trained model.

We try to combine the input text with four kinds of labels, EDU, span, nuclearity, and relation labels, which were defined and clearly described in the annotation document [Carlson and Marcu, 2001]. We list our extracted label descriptions from Carlson

Label	**Definition**
[EOS]	elementary discourse units are the minimal building blocks of a discourse tree
Span	span
Nucleus	a more salient or essential piece of information
Satellite	a supporting or background piece of information
Attribution	attribution, attribution represents both direct and indirect instances of reported speech
Background	background or circumstance
Cause	cause or result
Comparison	comparison, preference, analogy or proportion
Condition	condition, hypothetical, contingency or otherwise
Contrast	contrast relation, spans contrast with each other along some dimension. Typically, it includes a contrastive discourse cue, such as but, however, while.
Elaboration	elaboration, elaboration provides specific information or details to help define a very general concept
Enablement	enablement, enablement presentes action to increase the chances of the unrealized situation being realized.
Evaluation	evaluation, interpretation, conclusion or comment
Explanation	evidence, explanation or reason
Joint	list, list contains some sort of parallel structure or similar fashion between the units
Manner-Means	explaining or specifying a method , mechanism , instrument , channel or conduit for accomplishing some goal
Topic-Comment	problem solution, question answer, statement response, topic comment or rhetorical question
Summary	summary or restatement
Temporal	situations with temporal order, before, after or at the same time
Topic change	topic change
Textual-organization	links that are marked by schemata labels
Same-unit	links between two non-adjacent parts when separated by an intervening relative clause or parenthetical

Table 3.1: Extracted label definitions.

and Marcu [2001] in Table 3.1. For parsing symbols with brackets "(" and ")" like "(N" and ")$_N$", we inserted the position phrase, *the start of* and *the end of*, to the beginning of their label definitions. So the description of ")$_N$" is *the end of a more salient or essential piece of information*. To construct the label embedding for $p(X, Y, A)$, we combined the descriptions of the nuclearity and relation, and assigned the combination to the corresponding node. For example, the description of "(Attribution:S" is *the start of a supporting or background piece of information attribution, attribution represents both direct and indirect instances of reported speech.* In taking into account the descriptions for the labels as additional information, we adopt two different methods, Average and Concatenate, for representing the label embeddings.

Average: We average the embeddings of tokens that appear in the definition of a label and assign the averaged embedding to the label.

Concatenate: We concatenate a label name with its definition and insert the concatenated text to the end of sequence Z,[1] so that the label embedding can be captured by self-attention mechanisms [Vaswani et al., 2017]. Note that we do not try it in the parsing task, because the length of a sequence increases in proportion to the increase of the number of labels, that causes a shortage of memory space.

3.3.4 Objective Function

As a reranker, the LMR should compare all corresponding segmentations $\mathcal{Y}$ and parsing trees $\mathcal{A}$ for a given text X to select the most coherent combination. However, considering all possible segmentations and parsing trees for X is computationally infeasible [Stahlberg and Byrne, 2019]. Therefore, we consider subsets $\mathcal{Y}_{\alpha_{seg}}$ and $\mathcal{A}_{\alpha_{par}}$ that consist of the top-α_{seg} segmentations from the base segmenter and the top-α_{par} parsing trees from the base parser, respectively. After building $\mathcal{Y}_{\alpha_{seg}}$ or $\mathcal{A}_{\alpha_{par}}$, we could select a label sequence from $\{Y, D, U, R, A\}$ and construct a corresponding subset $\mathcal{Z}_\alpha$ for the joint sequence Z from all possible sequences $\mathcal{Z}$. We denote $Z_{gold} \in \mathcal{Z}$ as the correct label sequence of X. To keep pre-trained information in MPNet, for $Z \in \mathcal{Z}_\alpha \cup \{Z_{gold}\}$, we follow the setting of MPNet to randomly mask and permute Z and denote κ as the set of masked positions. To address issues like text degeneration and exposure bias, we followed previous research [Welleck et al., 2020, Song et al., 2021] to employ contrastive learning to optimize the parameters θ of the LMR. Given the masked sentence $Z_{\setminus\kappa}$, this is achieved by maximizing the likelihood of positive samples and minimizing the likelihood of negative samples as follows:

$$\begin{aligned}\theta^* &:= \arg\max_{\theta} \log p(Z_\kappa | Z_{\setminus\kappa}; \theta) \qquad (3.3.2)\\ &\approx \arg\max_{\theta} \sum_{t\in\kappa} [\underbrace{\mathbb{1}_Z \log p(z_t | Z_{\setminus\kappa}, t; \theta)}_{\text{likelihood}} + \underbrace{(1 - \mathbb{1}_Z) \log(1 - p(z_k | Z_{\setminus\kappa}, t; \theta))}_{\text{unlikelihood}}],\end{aligned}$$

where $\mathbb{1}_Z$ is the indicator function, defined as:

$$\mathbb{1}_Z := \begin{cases} 1 & \text{if } Z = Z_{gold} \\ 0 & \text{if } Z \neq Z_{gold} \end{cases}. \qquad (3.3.3)$$

[1] Note that the concatenated text of the label name and its definition is not masked during training.

Task	**Train**	**Valid**	**Test**
(a) Segmentation	6,768	905	991
(b) Parsing w/ gold segmentation	4,524	636	602
(c) Parsing w/ auto segmentation	-	861	951

Table 3.2: The number of sentences for each task.

3.3.5 Inference

In inference, for $Z \in \mathcal{Z}_\alpha$, the LMR scores Z by using softmax function as follows:

$$f(Z) = \frac{\exp(\mathrm{PLL}(Z;\theta)/|Z|)}{\sum_{Z' \in \mathcal{Z}_\alpha} \exp(\mathrm{PLL}(Z';\theta)/|Z'|)}. \tag{3.3.4}$$

We select Z based on $f(Z)$.

3.4 Experimental Setup

This section presents the experimental settings in three tasks, (a) discourse segmentation, (b) sentence-level discourse parsing with gold segmentation, and (c) sentence-level discourse parsing with automatic segmentation.

3.4.1 Datasets

Following previous studies [Wang et al., 2017, 2018c, Lin et al., 2019], this chapter used the RST Discourse Treebank (RST-DT) corpus [Carlson et al., 2002] as our dataset. This corpus contains 347 and 38 documents for training and test datasets, respectively. We divided the training dataset into two parts, following the module RSTFinder[2] [Heilman and Sagae, 2015], where 307 documents were used to train models and the remaining 40 documents were used as the validation dataset.

We split the documents into sentences while ignoring footnote sentences, as in Joty et al. [2012]. There happens two possible problematic cases for the splitted sentences: (1) The sentence consists of exactly an EDU, and so it has no tree structure. (2) The tree structure of the sentence goes across to other sentences. Following the setting of Lin et al. [2019], we did not filter any sentences in task (a). In task (b), we filtered sentences of both cases. In task (c), we filtered sentences of case (2). Table 3.2 shows the number of available sentences for the three different tasks.

[2] https://github.com/EducationalTestingService/rstfinder

Model	**Precision**	**Recall**	F_1
Reported*	92.04	94.41	93.21
Shared	92.22	95.35	93.76
Reproduced (ELMo)	**93.16**	**96.26**	**94.68**
Reproduced (MPNet)	92.84	95.63	94.21

Table 3.3: Performances of BiLSTM-CRF [Wang et al., 2018c] in the discourse segmentation task. The highest score in each metric among the models is indicated in **bold**. * indicates the reported score by Lin et al. [2019]. Shared is the publicly shared model by Wang et al. [2018c]. Reproduced (ELMo) and Reproduced (MPNet) are our reproduced models with different word embeddings.

Model	**Span**	**Nuclearity**	**Relation**
2-Stage Parser*	95.60	87.80	77.60
Pointer-networks*	97.44	91.34	81.70
Span-based Parser	96.67	90.23	74.76
2-Stage Parser	**97.92**	**92.07**	**82.06**

Table 3.4: Performance of retrained parsers in the sentence-level discourse parsing task with gold segmentation. The highest score in each metric among the models is indicated in **bold**. * indicates the reported score by Lin et al. [2019].

3.4.2 Evaluation Metrics

In task (a), we evaluated the segmentation in micro-averaged precision, recall, and F_1 score with respect to the start position of each EDU. The position at the beginning of a sentence was ignored. In task (b), we evaluated the parsing in micro-averaged F_1 score with respect to span, nuclearity, and relation. In task (c) for parsing with automatic segmentation, we evaluated both the segmentation and parsing in micro-averaged F_1 score. We used the paired bootstrap resampling [Koehn, 2004] for the significance test in all tasks when comparing two systems.

3.4.3 Compared Methods

As our proposed methods, we used LMR$_y$, LMR$_d$, LMR$_u$, LMR$_r$, and LMR$_a$, which respectively model probability $p(X,Y)$, $p(X,Y,D)$, $p(X,Y,U)$, $p(X,Y,R)$, and $p(X,Y,A)$ with initialized label embeddings. We represent LMR with Average and Concatenate label embeddings as Enhance and Extend, respectively.

We used the base discourse segmenter and parser described in Section 3.2 as our baseline. We reproduced the base discourse segmenter BiLSTM-CRF[3] [Wang et al.,

[3] https://github.com/PKU-TANGENT/NeuralEDUSeg

2018c]. Because BiLSTM-CRF adopted the hidden states of ELMo [Peters et al., 2018] as word embeddings, we also tried the last hidden state of MPNet as the word embeddings for BiLSTM-CRF for fairness. We retrained the segmenter in five runs, and the experimental results are showed in Table 3.3. The publicly shared BiLSTM-CRF by Wang et al. [2018c] is our base segmenter in the following experiments.

As for the base parser, we retrained two models, 2-stage Parser[4] [Wang et al., 2017] and span-based Parser[5] [Kobayashi et al., 2020]. Different from the setting of Lin et al. [2019], we retrained 2-stage Parser in the sentence-level rather than in the document-level. Because the experimental results in Table 3.4 show our retrained 2-stage Parser achieved the highest F_1 scores among several parsers , we selected it as our base parser in the following experiments.

Furthermore, for comparing LMR with an unidirectional generative model [Mabona et al., 2019], we constructed another baseline method which utilizes a GPT-2 as a reranker. This method follows an unidirectional language model-based generative parser [Choe and Charniak, 2016], and considers top-α results from the base model by an add-1 version of infinilog loss [Ding et al., 2020] during training. We denote this baseline as GPT2LM hereafter. Following the steps in Choe and Charniak [2016], GPT2LM with parameter θ utilized Equation (2.6.1) to compute $p(Z)$ as follows:

$$p(Z;\theta) = p(z_1, \ldots, z_{|Z|};\theta) = \prod_{t=1}^{|Z|} p(z_t|Z_{<t};\theta), \tag{3.4.1}$$

where $p(z_t|Z_{<t});\theta$ was computed by GPT-2 as described in Section 2.7.3. And in inference, it selected Z based on $\frac{1}{|Z|}\log p(Z)$. An add-1 version of infinilog loss [Ding et al., 2020] was utilized for training GPT2LM as follows:

$$\theta^* := \arg\max_{\theta} \log \frac{f(Z)}{1 + \sum_{Z' \in \mathcal{Z}_\alpha, Z' \neq Z} f(Z')}, \tag{3.4.2}$$

$$\text{where } f(Z) = \frac{\exp\left(\log p(Z;\theta)/|Z|\right)}{\sum_{Z' \in \mathcal{Z}_\alpha} \exp\left(\log p(Z';\theta)/|Z'|\right)}, \tag{3.4.3}$$

$\log p(Z;\theta)$ was computed by using cross-entropy loss as in Equation (2.6.3). GPT2LM models $p(X, Y)$ for task (a) and $p(X, Y, R)$ for tasks (b) and (c), respectively. Both LMR and GPT2LM are the ensemble of 5 models with different random seeds.

3.4.4 Hyperparmeters

For LMR, we used the source code shared in the public github[6] of Song et al. [2020]. We used the uploaded pre-trained MPNet and same setup as illustrated in Table 3.5. 15% tokens as the predicted tokens were masked by replacement strategy 8:1:1. Relative

[4] `https://github.com/yizhongw/StageDP`

[5] `https://github.com/nttcslab-nlp/Top-Down-RST-Parser`

[6] `https://github.com/microsoft/MPNet`

Hyperparameter	LMR	GPT2LM
Optimizer	adam	adam
Adam β_1	0.9	0.9
Adam β_2	0.98	0.98
Adam ϵ	$1e$ - 6	$1e$ - 6
weight decay	0.01	0.01
Learning rate	0.00009	0.0001
Batch size	8192 tokens	512 gold tokens + candidate tokens
Warm up steps	2.4 epoch	2.4 epoch
Epoch	30	30
Attention layer	12	12
Attention head	12	12
dropout	0.1	0.1
attention dropout	0.1	0.1
Hidden size	768	768
Vocab size	30527	50257+ added tokens
Tokenizer	Byte pair encoder	Byte pair encoder
Max sentence length	512	512

Table 3.5: List of used hyperparameters for LMR and GPT2LM.

positional embedding mechanism [Shaw et al., 2018] was utilized. Because the vocab we used is same as the one of BERT, we used the symbol [SEP] to represent [EDU] and symbol [unused#] starting from 0 to represent parsing labels such as "(N" and "(Attribution".

For GPT2LM, we used the source code shared in the public github[7] [Ott et al., 2019]. We used the uploaded pre-trained "gpt2" model [Wolf et al., 2020] and same setup as illustrated in Table 3.5. We used symbol "=====" in vocab to represent the symbol [EDU]. Because the vocab of GPT-2 has no available symbol for representing an unseen symbol, we added <pad> and our relation symbols to the vocab of GPT-2 and resized the pre-trained word embeddings.

3.5 Results

3.5.1 Candidates Tuning

As described in Section 3.3, LMR requires parameters α_{seg} and α_{par} for the number of candidates in the steps for different tasks. We tuned α_{seg} and α_{par} based on the performance on the validation dataset. Table 3.6 shows the setting of candidates for

[7] https://github.com/pytorch/fairseq/tree/master/fairseq/models/huggingface

Task	**Data**	**Segmentation**	**Parsing**			**# of data**
		α_{seg}	**1st stage**	**2rd stage**	α_{par}	
(a)	Training	20	-	-	-	140924
	Prediction	5	-	-	-	-
(b)	Training$_{\text{w/ span or nuclearity}}$	-	20	1	20	60742
	Training$_{\text{w/ relation or all}}$	-	3	7	20	95004
	Prediction	-	5	5	5	-
(c)	Prediction	5	5	5	5	-

Table 3.6: Setting of top candidates for different tasks. The Prediction data denotes the validation and test dataset.

Model	α_{seg} **for training**	**Precision**	**Recall**	F_1
LMR$_y$	0	87.76	95.72	91.57
	10	97.67	97.73	97.70
	20	**97.99**	**97.86**	**97.92**
GPT2LM$_y$	0	81.72	**96.18**	88.36
	10	96.67	96.05	96.36
	20	**96.93**	96.05	**96.48**

Table 3.7: Results of tuning α_{seg} for training in task (a). The highest score in each metric among different α_{seg} for training is indicated in **bold**.

different tasks. As described in Section 3.3.4, we do data augmentation by using additional top-α results generated by a base method, a larger α during training in Equation (3.3.2) is expected to bring more promotion for LMR. However, a larger α during prediction step in Equation (3.3.1) introduces more candidates and may make the prediction more difficult. Taking this into consideration, we tuned α_{seg} and α_{par} for training and prediction separately based on the performance on the validation dataset. The set of parameters was similarly tuned for GPT2LM on the validation dataset.

In task (a), we used the Viterbi-topk algorithm for the base segmenter to select top-α_{seg} segmentations. We tuned $\alpha_{seg} \in \{0, 10, 20\}$ for training while α_{seg} for prediction was fixed as 5.[8] Because the LMR$_y$ and GP2TLM$_y$ with $\alpha_{seg} = 20$ achieved the highest F_1 scores as shown in Table 3.7, we tuned $\alpha_{seg} \in \{5, 10, 20\}$ for prediction by using the LMR$_y$ and GP2TLM$_y$ trained with top-20 candidates.[9] The results in Table 4.10 shows that when α_{seg} for prediction was set to 5, the LMR$_y$ and GP2TLM$_y$ attained their highest F_1 scores on the validation datasets. Building upon these results, α_{seg} was set to 20 and 5 for training and prediction, respectively.

[8] Note that we used only gold segmentations for training when α_{seg} was set to 0.

[9] Oracle indicates the upper bound score that can be achieved with candidates generated by the base model. To compute the Oracle score, if the candidades by the base model include the correct answer, we assume the prediction is correct.

Model	**α_{seg} for prediction**	**Precision**	**Recall**	F_1
Oracle	5	99.94	99.68	99.81
	10	99.94	99.68	99.81
	20	99.94	99.68	99.81
LMR$_y$	5	**97.99**	**97.86**	**97.92**
	10	97.47	97.54	97.51
	20	97.41	97.60	97.51
GPT2LM$_y$	5	**96.93**	**96.05**	**96.48**
	10	96.47	95.59	96.03
	20	95.76	95.14	95.45

Table 3.8: Results of tuning α_{seg} for prediction in task (a). The highest score in each metric among different α_{seg} for prediction is indicated in **bold**.

Model	**α_{par} for training**	**Span**	**Nuclearity**	**Relation**
LMR$_r$	0	97.25	92.21	83.37
	10	97.46	92.71	83.23
	20	**97.50**	**93.02**	**83.44**
GPT2LM$_r$	0	**97.36**	**92.07**	79.11
	10	96.93	90.80	80.76
	20	96.79	90.66	**80.94**

Table 3.9: Results of tuning α_{par} for training in task (b). The highest score in each metric among different α_{par} for training is indicated in **bold**.

Model	**α_{par} for prediction**	**Span**	**Nuclearity**	**Relation**
Oracle	5	98.66	96.41	92.11
	10	99.30	98.03	94.43
	20	**99.47**	**98.48**	**95.42**
LMR$_r$	5	**97.50**	**93.02**	**83.44**
	10	97.50	92.46	83.30
	20	97.29	92.25	83.30
GPT2LM$_r$	5	**96.79**	**90.66**	**80.94**
	10	94.26	81.08	70.82
	20	93.27	77.20	66.67

Table 3.10: Results of tuning α_{par} for prediction in task (b). The highest score in each metric among different α_{par} for prediction is indicated in **bold**.

In task (b), we utilized beam search in each stage of the base parser and after two stages we computed the perplexity to keep top-α_{par} parsings. We tuned $\alpha_{par} \in \{0, 10, 20\}$ for training while α_{par} for prediction was fixed as 5. Table 3.9 shows that when α_{par} for training was set to 20, both the LMR$_r$ and GP2TLM$_r$ achieved the highest F_1 scores for the relation label. Then we tuned $\alpha_{par} \in \{5, 10, 20\}$ for prediction by using the LMR$_r$ and GP2TLM$_r$ trained with top-20 candidates. Table 3.10 shows the LMR$_r$ and

Model	α_{seg} for prediction	Precision	Recall	F_1
Oracle	5	99.93	99.65	99.79
	10	99.93	99.65	99.79
	20	99.93	99.65	99.79
LMR$_y$	5	**97.96**	**97.74**	**97.85**
	10	97.32	97.39	97.36
	20	97.33	97.53	97.43
GPT2LM$_y$	5	**96.94**	**95.91**	**96.42**
	10	96.45	95.63	96.04
	20	95.75	95.35	95.55

Table 3.11: Results of tuning α_{seg} for prediction in task (c). The highest score in each metric among different α_{seg} for prediciton is indicated in **bold**.

Model	α_{par} for prediction	Span	Nuclearity	Relation
Oracle	5	95.05	92.95	89.02
	10	95.93	94.73	91.25
	20	**96.21**	**95.36**	**92.45**
LMR$_r$	5	**94.39**	**90.12**	**80.88**
	10	94.39	89.45	80.74
	20	94.18	89.24	80.63
GPT2LM$_r$	5	**93.65**	**87.80**	**78.59**
	10	91.18	78.55	68.99
	20	90.30	74.96	65.19

Table 3.12: Results of tuning α_{par} for prediction in task (c). The highest score in each metric among different α_{par} for prediciton is indicated in **bold**.

GP2TLM$_r$ achieved the highest F_1 scores for all labels when $\alpha_{par} = 5$. Thus, α_{par} was set to 20 and 5 for training and prediction, respectively.

In task (c), we skipped tuning α_{seg} and α_{par} for training and instead directly used the tuned models from task (a) and (b). To tune α_{seg} and α_{par} for prediction, same as in task (a), we tuned $\alpha_{seg} \in \{5, 10, 20\}$ for predicting discourse segmentation by using the LMR$_y$ and GP2TLM$_y$ trained with top-20 candidates for task (a), Table 3.11 shows the result. Because LMR$_y$ and GP2TLM$_y$ achieved highest scores when $\alpha_{seg} = 5$, while LMR$_y$ achieved higher scores than GP2TLM$_y$, we utilized LMR$_y$ to select the best segmentation from top-5 segmentations for following discourse parsing. Then same as in task (b), we tuned $\alpha_{par} \in \{5, 10, 20\}$ for predicting discourse parsing by using the LMR$_r$ and GP2TLM$_r$ trained with top-20 candidates for task (b). Table 3.12 shows the result that LMR$_r$ and GP2TLM$_r$ yield highest scores when α_{par} for prediction was set to 5. Buidling upon these results, α_{seg} and α_{par} were both set to 5 for prediction.

In tasks (b) and (c), LMR$_d$ and Enhance$_d$ cannot distinguish the candidates with the same span labels but different nulearity or relation labels, LMR$_u$ and Enhance$_u$ cannot distinguish the candidates with the same nulearity labels but different relation labels.

Model	**Precision**	**Recall**	F_1
Oracle	97.73	98.67	98.20
Pointer-networks*	93.34	97.88	95.55
Base segmenter	92.22	95.35	93.76
GPT2LM$_y$	94.05	95.72	94.88
LMR$_y$	95.31	97.56	96.43†
Enhance$_y$	**95.54**	**97.93**	**96.72**†
Extend$_y$	95.05	97.86	96.44†

Table 3.13: Results for the discourse segmentation task. * indicates the reported score by Lin et al. [2019]. The highest score in each metric among the models is indicated in **bold**. † indicates that the score is significantly superior to GPT2LM with a p-value < 0.01.

Under this condition, in the following experiments, in task (b), for training data with span or nuclearity labels, we used the beam sizes 20 and 1 in the first and second stages of the base parser, respectively, as presented in Table 3.6. And the indistinguishable parsings would be ranked by the base parser.

3.5.2 Effect of Reranking

Table 3.13 shows the experimental results for the discourse segmentation task, where LMR$_y$ significantly outperformed GPT2LM$_y$.[10] We think the reason is similar to what Zhu et al. [2020] reported: BERT-based bidirectional encoder-only Transformer encodes more rhetorical features than GPT2-based unidirectional decoder-only Transformer. Using Average label embeddings is more helpful than using Concatenate label embeddings for LMR$_y$. Enhance$_y$ achieved the state-of-the-art F_1 score of 96.72, which outperformed both the base segmenter and the pointer-networks.

Table 3.14 shows the experimental results for the sentence-level discourse parsing task with gold segmentation. In Table 3.14, LMR$_u$ achieved the highest span and nuclearity F_1 scores of 98.31 and 94.00, respectively. Enhance$_r$ achieved the state-of-the-art relation F_1 score of 84.69, which is significantly superior to the base parser. Although using Average label embeddings improved LMR$_r$, it can provide no or only limited improvement for LMR$_u$ and LMR$_d$. We guess that this difference is caused by the number of different kinds of labels in span, nuclearity, and relation. The performance of GPT2LM$_r$ is even worse than the base parser. We think this is because we added the relation labels to the vocabulary of GPT-2 and resized the pre-trained word embeddings. Although Enhance$_a$ implements more label representations than Enhance$_r$, it actually achieved lower F_1 scores in terms of span, nuclearity, and relation. This discrepancy

[10] We chose GPT2LM$_y$ for the significance test because we had only reported scores for the pointer-networks.

Model	**Span**	**Nuclearity**	**Relation**
Oracle	98.67	95.88	90.07
Pointer-networks*	97.44	91.34	81.70
Base parser	97.92	92.07	82.06
GPT2LM$_r$	96.35	88.11	77.86
LMR$_d$	98.23‡	92.31	82.22
Enhance$_d$	98.27‡	92.39	82.42
LMR$_u$	**98.31**‡	**94.00**†	83.63†
Enhance$_u$	**98.31**†	93.88†	83.56†
LMR$_r$	98.00	93.09†	83.99†
Enhance$_r$	98.12	93.13†	**84.69**†
LMR$_a$	97.84	92.90	84.11
Enhance$_a$	98.04	92.74	84.18

Table 3.14: Results for the sentence-level discourse parsing task with gold segmentation. * indicates the reported score by Lin et al. [2019]. The highest score in each metric among the models is indicated in **bold**. † and ‡ indicate that the score is significantly superior to the base parser with a p-value < 0.01 and < 0.05, respectively.

may be attributed to the fact that Enhance$_a$ has a larger number of label types, leading to difficulties in training and causing the training loss not decreasing in our 2 out of 5 runs. As a consequence, the parsing results with automatic segmentation did not include the results using this type of tree.

Table 3.15 shows the experimental results for the sentence-level discourse parsing task with automatic segmentation. The second and third blocks in the table show the results for the first and second stages, discourse segmentation and sentence-level discourse parsing, respectively.[11] Enhance$_r$ achieved the highest relation F_1 score of 81.18, which is a significant improvement of 2.43 points compared to the base parser. Enhance$_d$ and LMR$_u$ achieved the highest span and nuclearity F_1 scores of 94.00 and 89.90, respectively. Because LMR$_*$ and Enhance$_*$ were the models trained in task (b), and Enhance$_y$ achieved the F_1 score of 96.79 in discourse segmentation, it is not surprising to find that the tendency of those results is similar to that in sentence-level discourse parsing with gold segmentation.

3.5.3 Evaluation with Respect to Relation Labels

Figure 3.4 shows the comparison between the base parser and Enhance$_r$ with respect to each ralation label. Overall, Enhance$_r$ outperformed 2-stage Parser in most relation labels, with a few exceptions such as *Explanation*, *Evaluation*, and *Topic-Comment*. While 2-stage Parser achieved an F_1 score of 17.14 for label *Temporal*, Enhance$_r$ sig-

[11] Note that F_1 scores for discourse segmentation in the second block are not the same as in Table 3.13 owing to the different test dataset.

Model	**Seg**	**Parse**		
		Span	**Nuclearity**	**Relation**
Pointer-networks*	-	91.75	86.38	77.52
Oracle$_{seg}$	98.24	-	-	-
Base segmenter	93.92	-	-	-
LMR$_y$	96.51	-	-	-
Enhance$_y$	**96.79**	-	-	-
Extend$_y$	96.48	-	-	-
Oracle	-	93.95	91.25	85.93
Base parser	-	93.53	88.08	78.75
GPT2LM$_r$	-	92.02	84.20	74.49
LMR$_d$	-	93.96‡	88.46	79.25
Enhance$_d$	-	**94.00**†	88.50	79.33
LMR$_u$	-	93.96†	**89.90**†	80.33†
Enhance$_u$	-	93.92‡	89.74†	80.22†
LMR$_r$	-	93.65	89.08†	80.57†
Enhance$_r$	-	93.73	89.16†	**81.18**†

Table 3.15: Results for the sentence-level discourse parsing task with automatic segmentation. * indicates the reported score by Lin et al. [2019]. The highest score in each metric among the models for each block is indicated in **bold**. We used the discourse segmentation results of Enhance$_y$ as the input of the discourse parsing stage for all models, for fair comparison of sentence-level discourse parsing. † and ‡ indicate that the score is significantly superior to the base parser with a p-value < 0.01 and < 0.05, respectively.

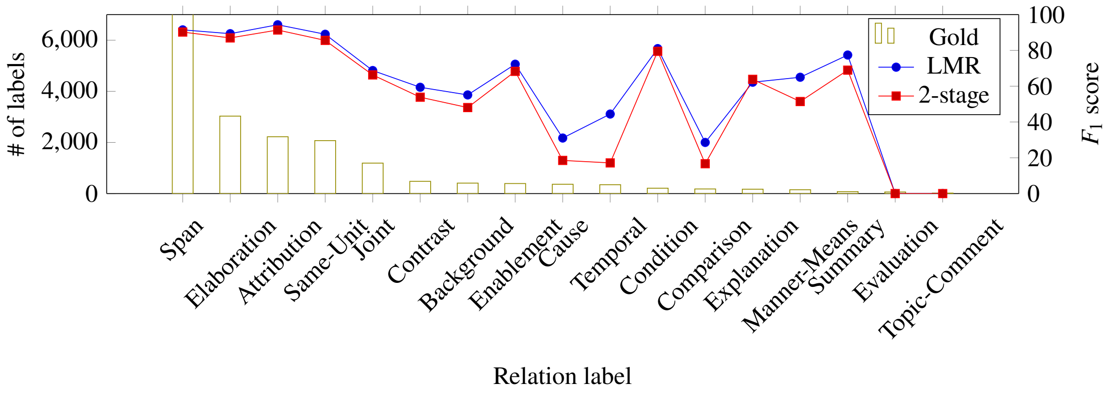
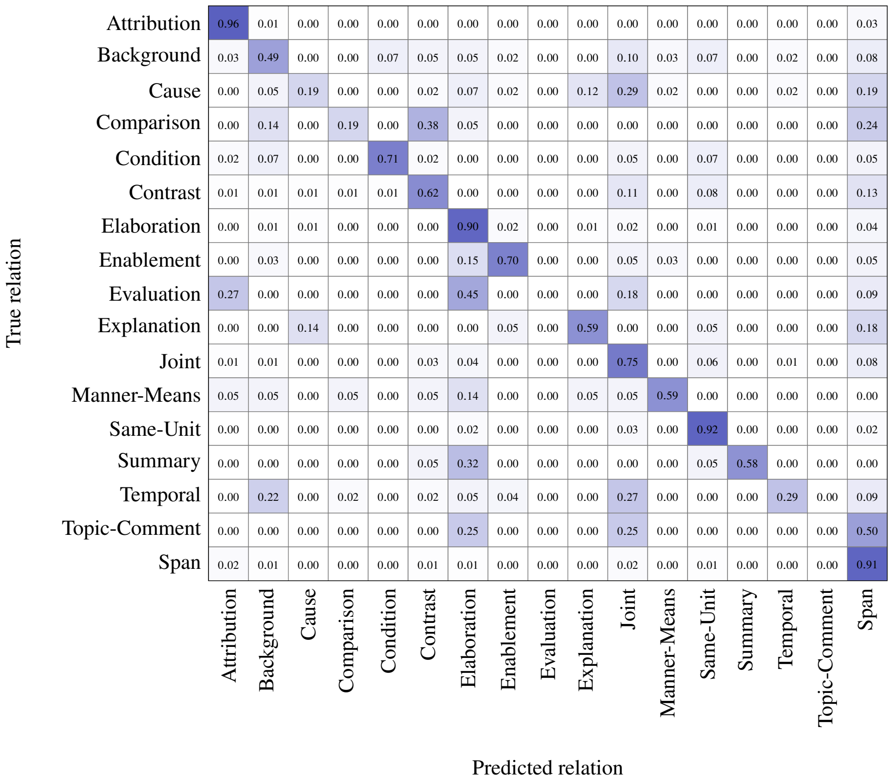


Figure 3.4: Performance of 2-stage parser and Enhance$_r$ in the sentence-level discourse parsing task with gold segmentation. The hollow bar denotes the number of different gold labels in the training dataset. Blue and red lines indicate the F_1 scores of Enhance$_r$ and 2-stage parser, respectively, for each relation label.

nificantly improved this score to 44.44 by reranking the parsing results from the 2-stage Parser. Similar substantial improvements can also be observed for labels like *Contrast*, *Background*, and *Cause*. This observation suggests that Enhance$_r$ tends to enhance the



	Attribution	Background	Cause	Comparison	Condition	Contrast	Elaboration	Enablement	Evaluation	Explanation	Joint	Manner-Means	Same-Unit	Summary	Temporal	Topic-Comment	Span
Attribution	0.96	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
Background	0.03	0.49	0.00	0.00	0.07	0.05	0.05	0.02	0.00	0.00	0.10	0.03	0.07	0.00	0.02	0.00	0.08
Cause	0.00	0.05	0.19	0.00	0.00	0.02	0.07	0.02	0.00	0.12	0.29	0.02	0.00	0.00	0.02	0.00	0.19
Comparison	0.00	0.14	0.00	0.19	0.00	0.38	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.24
Condition	0.02	0.07	0.00	0.00	0.71	0.02	0.00	0.00	0.00	0.00	0.05	0.00	0.07	0.00	0.00	0.00	0.05
Contrast	0.01	0.01	0.01	0.01	0.01	0.62	0.00	0.00	0.00	0.00	0.11	0.00	0.08	0.00	0.00	0.00	0.13
Elaboration	0.00	0.01	0.01	0.00	0.00	0.00	0.90	0.02	0.00	0.01	0.02	0.00	0.01	0.00	0.00	0.00	0.04
Enablement	0.00	0.03	0.00	0.00	0.00	0.00	0.15	0.70	0.00	0.00	0.05	0.03	0.00	0.00	0.00	0.00	0.05
Evaluation	0.27	0.00	0.00	0.00	0.00	0.00	0.45	0.00	0.00	0.00	0.18	0.00	0.00	0.00	0.00	0.00	0.09
Explanation	0.00	0.00	0.14	0.00	0.00	0.00	0.00	0.05	0.00	0.59	0.00	0.00	0.05	0.00	0.00	0.00	0.18
Joint	0.01	0.01	0.00	0.00	0.00	0.03	0.04	0.00	0.00	0.00	0.75	0.00	0.06	0.00	0.01	0.00	0.08
Manner-Means	0.05	0.05	0.00	0.05	0.00	0.05	0.14	0.00	0.00	0.05	0.05	0.59	0.00	0.00	0.00	0.00	0.00
Same-Unit	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.03	0.00	0.92	0.00	0.00	0.00	0.02
Summary	0.00	0.00	0.00	0.00	0.00	0.05	0.32	0.00	0.00	0.00	0.00	0.00	0.05	0.58	0.00	0.00	0.00
Temporal	0.00	0.22	0.00	0.02	0.00	0.02	0.05	0.04	0.00	0.00	0.27	0.00	0.00	0.00	0.29	0.00	0.09
Topic-Comment	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.50
Span	0.02	0.01	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.91

Figure 3.5: Confusion matrix for Enhance$_r$ in the sentence-level discourse parsing task with gold segmentation. We show the ratio of the number of instances with predicted labels (for a column) to the number of instances with gold labels (for a row) in the corresponding cell.

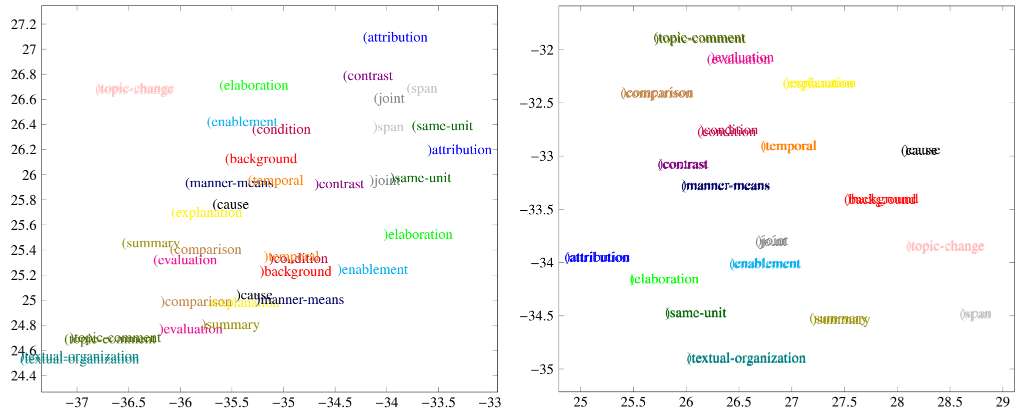
performance for labels with limited training data.

Figure 3.5 shows a confusion matrix of Enhance$_r$ for each relation label. The matrix reveals that Enhance$_r$ sometimes predicts the relation labels *Comparison*, *Cause*, and *Temporal* incorrectly as *Contrast*, *Joint*, and *Joint* or *Background*, respectively, despite the presence of at least 100 training instances for each label. We guess this might be owing to some similarities between those labels.

By using the t-SNE plot [Van der Maaten and Hinton, 2008], we visualize the trained relation label embeddings of LMR$_r$ and Enhance$_r$. The results are presented in Figure 3.6(a) and Figure 3.6(b). In Figure 3.6(a), a clear diagonal division can be observed, separating labels with parenthesis "(" from labels with ")", while Figure 3.6(b) shows more distinct divisions between labels.

3.6 Conclusion

This section proposed a *language model-based reranker*, LMR. Given the top-α discourse segmentations or parsings from the base model, as a reranker, the LMR achieved



(a) LMR$_r$

(b) Enhance$_r$

Figure 3.6: t-SNE plot of relation label embeddings trained in LMR$_r$ and Enhance$_r$.

the state-of-the-art performances in both discourse segmentation and sentence-level discourse parsing. The experimental results also showed the potential of constructing label embeddings from token embeddings by using label descriptions in the manual. Furthermore, the experiments demonstrated the effectiveness of the combined approach involving two strategies: implementing contrastive learning for optimization and utilizing bidirectional representations of target labels as input. This combination successfully encouraged the language model to assign higher probability to natural and coherent text, addressing concerns such as text degeneration. This promising result inspired the application of this strategy to the seq2seq architecture, which will be discussed in the subsequent chapter.

Chapter 4

Bidirectional Transformer Reranker

Grammatical error correction (GEC) is a sequence-to-sequence task which requires a model to aim to correct an ungrammatical sentence. An example is presented in Table 4.1. Various neural models for GEC have emerged [Junczys-Dowmunt et al., 2018, Kiyono et al., 2019, Kaneko et al., 2020, Rothe et al., 2021] owing to the importance of this task for language-learners who tend to produce ungrammatical sentences.

Previous studies have shown that GEC can be approached as machine translation by using a seq2seq model [Luong et al., 2015] with a Transformer architecture [Junczys-Dowmunt et al., 2018, Zhao et al., 2019, Kiyono et al., 2019, Kaneko et al., 2020, Rothe et al., 2021]. As a neural model consists of an encoder and a decoder, the seq2seq architecture typically requires a large amount of training data. Because GEC suffers from limited training data, applying a seq2seq model for GEC results in a low-resource setting, that can be handled by introducing synthetic data for training [Kiyono et al., 2019, Omelianchuk et al., 2020, Stahlberg and Kumar, 2021]. However, as pointed out by Rothe et al. [2021], using synthetic data in GEC may result in a distributional shift and require language-specific tuning, which can be time-consuming and resource-intensive.

Considering the limitations of the synthetic data, the current trend is to utilize the learned and general representations from a pre-trained model, such as BERT, XLNet, BART, and T5, which have been trained on large corpora and shown to be effective for various downstream tasks. According to Kaneko et al. [2020], incorporating a pre-trained masked language model into a seq2seq model could facilitate correction. In addition, as reported by Rothe et al. [2021], the pre-trained T5 model achieved state-of-the-art results on GEC benchmarks for four languages after successive fine-tuning with the cleaned LANG-8 corpus (cLang-8) [Rothe et al., 2021].

Although the seq2seq model with pre-trained representations has shown to be effective for GEC, its performance was still constrained by its unidirectional decoding. As suggested by Liu et al. [2021], for an ungrammatical sentence, a fully pre-trained seq2seq GEC model [Kiyono et al., 2019] could generate several high-quality grammatical sentences using beam search. However, even among these candidates, there may be still a gap between the selected hypothesis and the most grammatical one. Our experimental results, listed in Table 4.14, also demonstrate their investigation. To solve this decoding problem, given the hypotheses of a seq2seq GEC model, Kaneko et al. [2019] used BERT to classify between ungrammatical and grammatical hypotheses, and reranked

Source	However , it is a good practice not to intesively use social media all the time .
Gold 1	However , it is a good practice not to intensely use social media all the time .
Gold 2	However , it is good practice not to intensively use social media all the time .
Candidate 1 (R2L, RoBERTa, T5GEC)	However , it is good practice not to intesively use social media all the time .
Candidate 2	However , it is good practice not to intensely use social media all the time .
Candidate 3 (BTR)	However , it is good practice not to insensitively use social media all the time .
Source	It is true that social media makes people be able to connect one another more conveniently .
Gold 1	It is true that social media allows people to connect to one another more conveniently .
Gold 2	It is true that social media make people able to connect with one another more conveniently
Candidate 1 (RoBERTa, T5GEC)	It is true that social media makes people be able to connect with one another more conveniently .
Candidate 2 (BTR, R2L)	It is true that social media makes people able to connect with one another more conveniently .
Candidate 3	It is true that social media makes people able to connect to one another more conveniently .
Source	Speed camera can be placed in many locations along a highway .
Gold 1	Speed cameras can be placed in many locations along a highway .
Candidate 1 (RoBERTa, T5GEC)	A speed camera can be placed in many locations along a highway .
Candidate 2 (BTR, R2L)	Speed cameras can be placed in many locations along a highway .
Candidate 3	A Speed camera can be placed in many locations along a highway .
Source	Disadvantage is parking their car is very difficult .
Gold 1	A disadvantage is that parking their cars is very difficult .
Gold 2	A disadvantage is that parking their car is very difficult .
Gold 3	The disadvantage is that parking their car is very difficult .
Candidate 1 (R2L, RoBERTa, T5GEC)	Disadvantage is parking their car is very difficult .
Candidate 2 (BTR)	The disadvantage is parking their car is very difficult .
Candidate 3	The disadvantage is that parking their car is very difficult .

Table 4.1: Examples of reranked outputs. The 3 candidate sentences were generated using T5GEC (§4.4.1). Blue indicates the range of corrections. “Candidate 1 (T5GEC)” denotes that T5GEC regards “Candidate 1” as the most grammatical correction. Examples in the first two and last two block were extracted from the CoNLL-14 [Ng et al., 2014] and JFLEG test corpus [Napoles et al., 2017], respectively.

them on the basis of the classification results. The previous studies [Kiyono et al., 2019, Kaneko et al., 2020] also showed that the seq2seq GEC model decoding in an opposite direction, i.e., right-to-left, is effective as a reranker for a left-to-right GEC model.

Therefore, to further improve the performance of the pre-trained seq2seq model for GEC, it is essential to find ways to leverage the bidirectional representations of the target context. In this study, on the basis of the seq2seq-style Transformer model, we propose a *bidirectional Transformer reranker* (BTR) to handle the interaction between the source sentence and the bidirectional target context. The BTR utilizes a BERT-style self-attention mechanism in the decoder to predict each target token using MLM. Given several candidate target sentences from a base model, the BTR can re-estimate the sentence probability for each candidate from the bidirectional representation of the candidate, which is different from the conventional seq2seq model. During training, for guiding the reranking, we adopt negative sampling for the objective function to minimize the unlikelihood while maximizing the likelihood. In inference, considering the robustness of pre-trained models, we compare the reranked top-1 results with the original ones using an acceptance threshold β to decide whether to accept the suggestion from the BTR.

We regard the state-of-the-art model for GEC [Rothe et al., 2021], a pre-trained Transformer model, T5 (either T5-base or T5-large), as our base model and utilize its generated candidates for reranking. Because the BTR can inherit learned representations from a pre-trained Transformer model, we construct the BTR on top of T5-base. Our experimental results showed that, by reranking candidates from a fully pre-trained and fine-tuned T5-base model, the BTR on top of T5-base can achieve an $F_{0.5}$ score of 65.47 on the CoNLL-14 benchmark. The BTR on top of T5-base also outperformed T5-base on the BEA test set [Bryant et al., 2019] by 0.76 points, achieving an $F_{0.5}$ score of 71.27. Adopting negative sampling for the BTR also generated a peaked probability distribution for ranking, and so grammatical suggestions could be selected by using β. Furthermore, the BTR on top of T5-base was robust even when reranking candidates from T5-large and improved the performance by 0.26 points on the BEA test set.

4.1 Related Work on Grammatical Error Correction

For directly predicting the target corrections from given input tokens, Omelianchuk et al. [2020] and Malmi et al. [2022] regarded the encoder-only Transformer as a non-autoregressive GEC sequence tagger. The experimental results of Omelianchuk et al. [2020] showed that, compared with the randomly initialized LSTM, the pre-trained models, such as RoBERTa [Liu et al., 2019], GPT-2, and ALBERT [Lan et al., 2020], can achieve higher $F_{0.5}$ scores as a tagger. Sun et al. [2021] considered GEC as a seq2seq task and introduced the Shallow Aggressive Decoding (SAD) for the decoder of the Transformer. With the SAD, the performance of a pre-trained seq2seq model, BART, surpassed the sequence taggers of Omelianchuk et al. [2020]. The T5 xxl model is a pre-trained seq2seq model with 11B parameters [Raffel et al., 2020]. After fine-tuning with the cLang-8 corpus, T5 xxl and mT5 xxl [Xue et al., 2021], a multilingual version of T5, achieved state-of-the-art results on GEC benchmarks in four languages: English, Czech, German, and Russian [Rothe et al., 2021]. This demonstrated that performing a single fine-tuning step for a fully pre-trained seq2seq model is a simple and effective method for GEC without incorporating a copy mechanism [Zhao et al., 2019], the SAD or the output from a pre-trained masked language model [Kaneko et al., 2020]. Despite the improvements brought about by the pre-trained representations, the conventional seq2seq structure suffers from a prediction bias due to its unidirectional decoding. According to Liu et al. [2021], by using beam search, a fully pre-trained seq2seq GEC model [Kiyono et al., 2019] can generate several high-quality grammatical hypotheses, which include one that is more grammatical than the selected one.

To address the shortcoming of the unidirectional decoding, previous studies [Kiyono et al., 2019, Kaneko et al., 2019, 2020] introduced reversed representations to rerank the hypotheses. Kiyono et al. [2019] and Kaneko et al. [2020] utilized a seq2seq GEC model that decodes in the opposite direction (right-to-left) to rerank candidates, which was effective to select a more grammatical sentence than the original one. This finding motivated us to use a bidirectional decoding method for our model. Instead of using a seq2seq model, Kaneko et al. [2019] fine-tuned BERT as a reranker to evaluate the grammatical quality of a sentence. By using masked language modeling, BERT learned deep bidirectional representations to distinguish between grammatical and ungrammatical sentences. However, BERT did not account for the positions of corrections, as it discarded the source sentence and considered only the target sentence. This made it difficult for BERT, as a reranker, to recognize the most suitable corrected sentence for an ungrammatical sentence. Salazar et al. [2020] proposed the use of pseudo-log-likelihood scores (PLL) for reranking. They demonstrated that RoBERTa, with the PLL for reranking, outperformed the conventional language model GPT-2 when reranking candidates in speech recognition and machine translation tasks. Chapter 3 also proved that the pre-trained model, MPNet, was more effective than GPT-2 when using PLL for reranking in discourse segmentation and parsing.

Zhang and van Genabith [2021] proposed a bidirectional Transformer-based alignment (BTBA) model, which aims to assess the alignment between the source and target tokens in machine translation. To achieve this, BTBA masked and predicted the current token with attention to both left and right sides of the target context to produce alignments for the current token. Specifically, to assess alignments from the attention scores in all cross-attention layers, the decoder in BTBA discarded the last feed-forward layer of the Transformer model and directly predicted masked tokens from the output of the last cross-attention layer. Even though the target context on both sides was taken into consideration, one limitation of BTBA was that the computed alignments ignored the representation of the current token. To produce more accurate alignments, Zhang and van Genabith [2021] introduced full context based optimization (FCBO) for fine-tuning, in which BTBA no longer masks the target sentence to use the full target context.

This chapter, to determine the most appropriate correction for a given erroneous sentence, models the BTR as a seq2seq reranker, which encodes the erroneous sentence using an encoder and decodes a corrected sentence using a decoder. In contrast to the conventional seq2seq model, we use masked language modeling to mask and predict each target token in the decoder and estimate the sentence probability for each candidate using PLL. Unlike BTBA, the BTR preserves the last feed-forward layer in the decoder to predict masked tokens more accurately. Because the original data of the masked tokens should be invisible in the prediction, the FCBO fine-tuning step is not used in the BTR. Compared with BTBA, the BTR keeps the structure of the Transformer model and can easily inherit parameters from pre-trained models.

4.2 Generating Corrections with Discriminative Models

Given an ungrammatical sentence $X = (x_1, \ldots, x_I)$, a GEC model corrects X into its grammatical sentence $Y = (y_1, \ldots, y_J)$. Because the pre-trained T5 model with Transformer architecture achieved state-of-the-art results in GEC by using beam search for decoding [Rothe et al., 2021], this chapter regards the T5 as the base model for generating candidates, whose last hidden state is computed as described in Section 2.7.1. However, previous studies [Li and Jurafsky, 2016, Vijayakumar et al., 2018] have suggested that beam search tends to generate sequences with slight differences. This can constrain the upper bound score when reranking candidates [Ippolito et al., 2019]. To select the optimal decoding method for a Transformer-based GEC Model, T5GEC, we compared

Method	**Gold (%)**	**Unique (%)**	**Oracle ($F_{0.5}$)**
Nucleus sampling	28.70	43.61	49.27
Top-k sampling	29.62	48.57	48.40
Beam search	**37.97**	**98.93**	**55.11**
Diverse beam search	28.46	38.78	50.39

Table 4.2: Results for the T5GEC on the CoNLL-13 corpus with various decoding methods.

beam search with diverse beam search [Vijayakumar et al., 2018], top-k sampling [Fan et al., 2018], and nucleus sampling [Holtzman et al., 2020]. For each pair of data in CoNLL-13 corpus [Ng et al., 2013], we required all decoding methods to generate 5 candidate sequences with a maximum sequence length of 200. When using diverse beam search, we fixed the beam group and diverse penalty to 5 and 0.4, respectively. Meanwhile, we set the top-k as 50 and the top-p as 0.95 for top-k sampling and nucleus sampling, respectively.

Table 4.2 presents the compared results among different decoding methods. Unique (%) indicates the rate of unique sequences among all candidates. Gold (%) indicates the rate of pairs of data whose candidates include the correct answer. The results show that beam search generates more diverse sentences with the highest Oracle score compared to nucleus sampling, top-k sampling, and diverse beam search. This may be because, in the GEC task, most of the tokens in the target are the same as the source, which causes a peaked probabilities distribution to focus on one or a small number of tokens. And thus, a top-k filtering method like beam search generates more diverse sentences than sampling or using probability as a diverse penalty. Based on these results, we have chosen beam search as the decoding method for T5GEC during inference. For evaluating T5GEC, it generates the top-ranked hypothesis with a beam size of 5. To generate the top-α candidates $\mathcal{Y}_\alpha = \{Y_1, \dots, Y_\alpha\}$ for reranking, it generates hypotheses with a beam size of α and a maximum sequence length of 128 and 200 for the datasets in training and prediction, respectively.

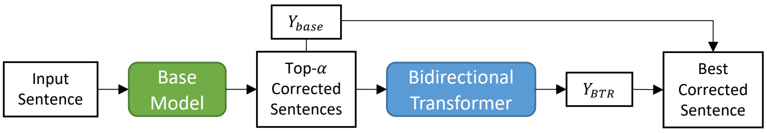


Figure 4.1: Overview of the reranking procedure by using the bidirectional Transformer reranker (BTR).

4.3 Reranking Corrections with Bidirectional Representations from Transformer

The BTR uses MLM in the decoder to estimate the probability of a corrected sentence. Given an ungrammatical sentence X, a base GEC model first generates the top-α corrected sentences $\mathcal{Y}_\alpha$, as described in Section 4.2. Assume $Y_{base} \in \mathcal{Y}_\alpha$ is the top-ranked hypothesis from the base GEC model. The BTR selects and accepts the most optimal corrected sentence Y_{BTR} from $\mathcal{Y}_\alpha$ on the basis of the estimated sentence probability, as described in the following. Figure 4.1 shows the overview of the BTR for the whole procedure.

4.3.1 Target Sentence Probability

As PLL has been effective in estimating the sequence probability for reranking, we decompose the conditional sentence probability of Y as:

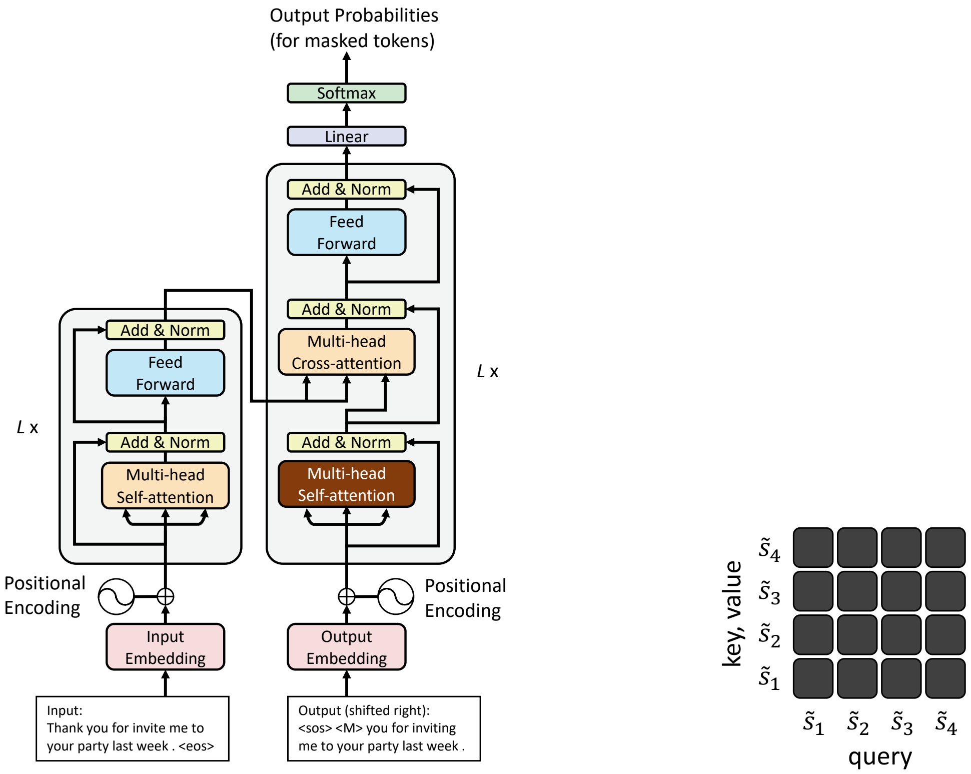
$$\log p(Y|X;\theta) \approx \text{PLL}(Y|X;\theta) = \sum_{j=1,\kappa=\{j\}}^{|Y|} \log p(y_j|Y_{\backslash\kappa}, j, X;\theta). \tag{4.3.1}$$

As in Equation (2.7.2), a linear transformation with the softmax function is utilized for the final hidden state $\tilde{s}_{j\backslash\kappa}$ to predict $p(y_j|Y_{\backslash\kappa}, j, X;\theta)$.

Same as the seq2seq-style Transformer architecture, $\tilde{s}_{j\backslash\kappa}$ is the result of $s_{j\backslash\kappa}$ after the cross-attention and feed-forward layers, as presented in Equation (2.7.7). We assume the decoder consists of L layers. To capture the bidirectional representation, for $\ell \in L$, we compute $s^\ell_{j\backslash\kappa}$ as:

$$s^\ell_{j\backslash\kappa} = \text{Attn}_s(\tilde{s}^{\ell-1}_{j\backslash\kappa}, \widetilde{S}^{\ell-1}_{\backslash\kappa}, \widetilde{S}^{\ell-1}_{\backslash\kappa}), \tag{4.3.2}$$

where $\tilde{s}^0_{j\backslash\kappa}$ is the embedding of the $(j-1)^{\text{th}}$ word in $Y_{\backslash\kappa}$ and $\tilde{s}_1$ is the state of the start token <sos>. $\widetilde{S}^{\ell-1}_{\backslash\kappa} = (\tilde{s}^{\ell-1}_{1\backslash\kappa}, \ldots, \tilde{s}^{\ell-1}_{J\backslash\kappa})$ denotes a set of hidden states for the shifted right $Y_{\backslash\kappa}$, i.e. the joint sequence of <sos> and $Y_{\backslash\kappa}$ without <eos>. The procedure of using the BTR to predict $p(y_1|Y_{\backslash\{1\}}, 1, X;\theta)$ is shown in Figure 4.2(a). Figure 4.2(b) shows our fully-visible attention mask for computing $S^\ell_{\backslash\kappa}$ in parallel, where "$\backslash\kappa$" is omitted for easier reading.


Output Probabilities
(for masked tokens)
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-head Cross-attention
Add & Norm
Multi-head Self-attention
L x
Add & Norm
Feed Forward
Add & Norm
Multi-head Self-attention
L x
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Input:
Thank you for invite me to your party last week . <eos>
Output (shifted right):
<sos> <M> you for inviting me to your party last week .
key, value
$\tilde{s}_4$ $\tilde{s}_3$ $\tilde{s}_2$ $\tilde{s}_1$
$\tilde{s}_1$ $\tilde{s}_2$ $\tilde{s}_3$ $\tilde{s}_4$
query

(a) Bidirectional Transformer

(b) Fully-visible

Figure 4.2: The bidirectional Transformer architecture (a) utilizes a fully-visible self-attention mechanism (b) in the decoder, distinguishing it from the conventional Transformer.

4.3.2 Objective Function

We followed LMR to consider a subset $\mathcal{Y}_\alpha$ of all corresponding corrected sentences $\mathcal{Y}$ based on the top-α results from the base GEC model instead. Let $Y_{gold} \in \mathcal{Y}$ denote the gold correction for X. For $Y \in \mathcal{Y}_\alpha \cup \{Y_{gold}\}$, we follow the setting of BERT to randomly mask 15% of Y and denote κ as the set of masked positions. As a result, the distribution of the masked tokens satisfies the 8:1:1 masking strategy. Following the LMR, given the masked sentence $Y_{\backslash\kappa}$, the model parameter θ of the BTR is optimized by maximizing the likelihood and minimizing the unlikelihood as:

$$\begin{aligned}\theta^* &:= \arg\max_\theta \log p(Y_\kappa | Y_{\backslash\kappa}, X; \theta) \\ &\approx \arg\max_\theta \sum_{j\in\kappa} [\mathbb{1}_Y \log p(y_j | Y_{\backslash\kappa}, j, X; \theta) + (1 - \mathbb{1}_Y) \log(1 - p(y_j | Y_{\backslash\kappa}, j, X; \theta))],\end{aligned} \tag{4.3.3}$$

where $p(y_j|Y_{\backslash\kappa}, j, X; \theta)$ is computed as in Section 4.3.1. $\mathbb{1}_Y$ is an indicator function defined as follows:

$$\mathbb{1}_Y := \begin{cases} 1 & \text{if } Y = Y_{gold} \\ 0 & \text{if } Y \neq Y_{gold} \end{cases}. \tag{4.3.4}$$

4.3.3 Inference

Following the LMR, in inference, for $Y \in \mathcal{Y}_\alpha$, the BTR scores Y by

$$f(Y|X) = \frac{\exp(\mathrm{PLL}(Y|X;\theta)/|Y|)}{\sum_{Y' \in \mathcal{Y}_\alpha} \exp(\mathrm{PLL}(Y'|X;\theta)/|Y'|)}. \tag{4.3.5}$$

Hereafter, we denote $Y_{BTR} \in \mathcal{Y}_\alpha$ as the candidate with the highest score $f(Y_{BTR}|X)$ for given X in the BTR. Here, $f(Y|X)$ is also considered to indicate the confidence of the BTR. Because the BTR is optimized with Equation (4.3.3), a high score for Y_{BTR} indicates a confident prediction while a low score indicates an unconfident prediction, as proved in Section 4.5.7.

Considering that we build the base GEC model from a fully pre-trained seq2seq model and the BTR from an insufficiently pre-trained model, we introduce an acceptance threshold β to decide whether to accept the suggestion from the BTR. We accept Y_{BTR} only when it satisfies the following equation; otherwise, Y_{base} is still the final result:

$$f(Y_{BTR}|X) - f(Y_{base}|X) > \beta, \tag{4.3.6}$$

where β is a hyperparameter tuned on the validation data.

4.4 Experimental Setup

4.4.1 Compared Methods

We evaluated the BTR as a reranker for two versions of candidates, normal and high-quality ones, generated by two seq2seq GEC models, T5GEC and T5GEC (large). We compared the BTR with three other rerankers, R2L, BERT, and RoBERTa.

T5GEC: We used the state-of-the-art model [Rothe et al., 2021] as our base model for GEC. This base model inherited the pre-trained T5 version 1.1 model (T5-base) [Raffel et al., 2020] and was fine-tuned as described in Section 2.7.1. We denote this base model as T5GEC hereafter. Although the T5 xxl model yielded the most grammatical sentences in Rothe et al. [2021], it contained 11B parameters and was not suitable for our current experimental environment. Thus, we modeled T5GEC on top of a 248M-parameter T5-base model. To reproduce the experimental results of Rothe et al. [2021], we followed their setting and fine-tuned T5GEC once with the cLang-8 dataset.

T5GEC (large): To investigate the potential of the BTR for reranking high-quality

Model	**Inputs**	**Targets**
Self-supervised learning for pre-training		
BERT / RoBERTa	Thank you so <M> me to your party <M> week .	Thank you for inviting me to your party last week .
T5 / R2L	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
BTR	Thank you <X> me to your party <Y> week .	<X> for <M> you last <Z>
Supervised learning for fine-tuning		
BERT	Thank you for inviting me to your party last week .	<1>
T5 / R2L	Thank you for invite me to your party last week .	Thank you for inviting me to your party last week .
BTR / RoBERTa	Thank you for invite me to your party last week .	Thank you so <M> me to your party <M> week .

Table 4.3: Examples of data pairs for self-supervised and supervised learning used by each model. The grammatical text is “ Thank you for inviting me to your party last week . ” <M> denotes a mask token. <X>, <Y>, and <Z> denote sentinel tokens that are assigned unique token IDs. <1> denotes the input sentence is classified as a grammatical sentence. Red indicates an error in the source sentence while Blue indicates a token randomly replaced by the BERT-style masking strategy.

candidates, we also fine-tuned one larger T5GEC model with a 738M-parameter T5-large structure. We denote this model as T5GEC (large).

R2L: The decoder of the conventional seq2seq model can generate a target sentence either in a left-to-right or right-to-left direction. Because T5GEC utilized the left-to-right direction, and previous research [Sennrich et al., 2016, Kiyono et al., 2019, Kaneko et al., 2020] showed the effectiveness of reranking using the right-to-left model, we followed Kaneko et al. [2020] to construct four right-to-left T5GEC models, which we denote as R2L. R2L reranks candidates based on the sum score of the base model (L2R) and ensembled R2L.

BERT: We followed Kaneko et al. [2019] to fine-tune four BERT with 334M parameters. During fine-tuning, both source and target sentences were annotated with either <0> (ungrammatical) or <1> (grammatical) label for BERT to classify. During inference, the ensembled BERT reranks candidates based on the predicted score for the <1> label.

RoBERTa: We fine-tuned four 125M parameters RoBERTa to compare our bidirectional Transformer structure with the encoder-only one. During fine-tuning, the source and target sentences were concatenated, and RoBERTa masked and predicted only the target sentence as the BTR. During prediction, the ensembled RoBERTa reranks candidates with the acceptance threshold β as the BTR.

4.4.2 Setup for the BTR

Because there was no pre-trained seq2seq model with a self-attention mechanism for masked language modeling in the decoder, we constructed the BTR using the 248M T5 model (T5-base) and pre-trained it with the Realnewslike corpus [Zellers et al., 2019]. To compare the BTR with R2L, we also constructed R2L using T5-base, and pre-trained both models as follows. To speed up pre-training, we initialized the BTR and R2L

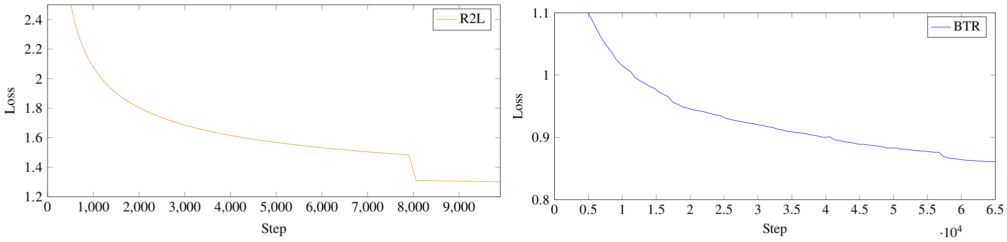


Figure 4.3: Pre-training loss for R2L (left) and the BTR (right).

Dataset	Usage	Lang	Level	# of data (pairs)
Realnewslike	pre-train	EN	-	148,566,392
cLang-8	train	EN	-	2,372,119
CoNLL-13 (cleaned)	valid	EN	-	1,381
CoNLL-14	test	EN	-	1,312
JFLEG	test	EN	-	747
BEA	test	EN	A	1,107
			B	1,330
			C	1,010
			N	1,030

Table 4.4: Dataset sizes.

model parameters with the fine-tuned parameters θ of T5GEC. During pre-training, we followed Raffel et al. [2020] for self-supervised learning with a span masking strategy. Specifically, 15% of the tokens in a given sentence were randomly sampled and removed. The input sequence was constructed by the rest tokens while the target sequence was the concatenation of dropped-out tokens. An example is provided in Table 4.3. We pre-trained the BTR and R2L with $65536 = 2^{16}$ and 10000 steps, respectively. Because the BTR masked and predicted only 15% of the tokens in Eq. (4.3.3), the true steps for the BTR were $2^{16} \times 0.15 \approx 10000$. We used a maximum sequence length of 512 and a batch size of $2^{20} = 1048576$ tokens. In total, we pre-trained $10000 \times 2^{20} \approx 10.5$B tokens, which were less than the pre-trained T5 with 34B tokens. The pre-training for R2L and the BTR took 2 and 13 days, respectively, with 2 NVIDIA A100 80GB GPUs. This indicates the BTR requires more training time and resources than R2L. We provide a plot of the pre-training loss in Figure 4.3, the training loss of R2L suddenly dropped from 1.48 to 1.3 after the first epoch (7957 steps).

After pre-training, we successively fine-tuned the BTR with the cLang-8 dataset. Like R2L, BERT, and RoBERTa, our fine-tuned BTR is the ensemble of four models with random seeds.

Hyperparameters	**T5GEC**	**BERT**	**RoBERTa**	**R2L (pretrain)**	**R2L (finetune)**	**BTR (pretrain)**	**BTR (finetune)**
# of updates	15 (epochs)	15 (epochs)	15 (epochs)	10000	15 (epochs)	65536	15 (epochs)
Max src / tgt length (train)	128	128	128	512	128	512	128
Max src / tgt length (eval)	512	1	512	512	512	512	512
α_{train}	-	-	{0, 5, 10, 20}	-	-	-	{0, 5, 10, 20}
α_{pred}	-	{5, 10, 15, 20}	{5, 10, 15, 20}	-	{5, 10, 15, 20}	-	{5, 10, 15, 20}
Threshold (β)	-	-	{0, 0.1, 0.2,. . . ,0.9}	-	-	-	{0, 0.1, 0.2,. . . ,0.9}

Table 4.5: Used hyperparameters.

4.4.3 Datasets

For fair comparison, we pre-trained R2L and the BTR with the Realnewslike corpus. This corpus contains 37 GB of text data and is a subset of the C4 corpus [Raffel et al., 2020]. To shorten the input and target sequences, we split each text into paragraphs. During fine-tuning, we followed the steps of Rothe et al. [2021] and regarded the cLang-8 corpus as the training dataset.

While the CoNLL-13 dataset was used for validation, the standard benchmarks from JFLEG, CoNLL-14, and the BEA test set were used for evaluation. While the CoNLL-14 corpus considers the minimal edit of corrections, JFLEG evaluates the fluency of a sentence. The BEA corpus contains much more diverse English language levels and domains than the CoNLL-14 corpus. Each sentence in the BEA test set is classified into either A (beginner), B (intermediate), C (advanced), or N (native) corresponding to the Common European Framework of Reference for Languages (CEFR) level [Council of Europe., 2001]. We used a cleaned version of CoNLL-13 with consistent punctuation tokenization styles. Section 4.5.5 lists our cleaning steps and the experimental results on the cleaned CoNLL-14 set. Table 4.4 summarizes the data statistics.

4.4.4 Evaluation Metrics

The evaluation on the BEA test set was automatically executed in the official BEA-19 competition in terms of span-based correction $F_{0.5}$ using the ERRANT [Bryant et al., 2017] scorer. For the CoNLL-13 and 14 benchmarks, we evaluated the correction $F_{0.5}$ using the official M^2 [Dahlmeier and Ng, 2012] scorer. For the JFLEG corpus, we evaluated the GLEU [Napoles et al., 2015].

We report only significant results on the CoNLL-14 set, because the gold data for the BEA test set is unavailable, and the evaluation metric GLEU for the JFLEG test set requires a sampling strategy for multiple references. We used the paired t-test to evaluate whether the difference between Y_{BTR} and Y_{base} on the CoNLL-14 set is significant, as only limited Y_{BTR} differed from Y_{base} among the suggestions from the BTR.

4.4.5 Hyperparameters

Table 4.5 lists the hyperparameter settings used for each model. And Table 4.6 lists the used artifacts. The setting for T5GEC (large) was the same as T5GEC. We followed the setting of Kaneko et al. [2019] to use a 0.0005 learning rate for the BERT reranker. We

Used artifacts	**Note**
T5-base	`https://huggingface.co/google/t5-v1_1-base`
T5-large	`https://huggingface.co/google/t5-v1_1-large`
T5GEC	`https://github.com/google-research-datasets/clang8/issues/3`
RoBERTa	`https://huggingface.co/roberta-base`
BERT	`https://huggingface.co/bert-large-cased`
cLang-8	`https://github.com/google-research-datasets/clang8`
CoNLL-13	File *revised/data/official-preprocessed.m2*
CoNLL-14	File *alt/official-2014.combined-withalt.m2*
JFLEG	File *test/test.src*
ERRANT	`https://github.com/chrisjbryant/errant`
Fairseq	`https://github.com/facebookresearch/fairseq/`
HuggingFace	`https://github.com/huggingface/transformers/`
BEA-19 competition	`https://competitions.codalab.org/competitions/20229`

Table 4.6: Used artifacts.

α_{train}	**# of training data (pairs)**
0	2,371,961
5	13,727,133
10	22,396,187
20	30,423,347

Table 4.7: Number of sentence pairs for cLang-8 dataset with candidates. All pairs of data that satisfy the length constraint of 128 are listed.

used a 0.0001 learning rate for the RoBERTa reranker. For both BERT and RoBERTa, we utilized the adam optimizer, "inverse square root" learning rate schedule, and 1.2 epochs warm-up steps. For other models based on a T5 structure, we used a 0.001 learning rate and adafactor optimizer. The batch size was 1048576 tokens for all models. We used the Fairseq [Ott et al., 2019] and HuggingFace [Wolf et al., 2020] to reproduce all models and run the BTR.

4.5 Results

4.5.1 Candidate and Threshold Tuning

We followed the setting of the LMR to separately tune α for training and prediction, based on the model performance on the validation dataset with candidates generated by T5GEC. Table 4.7 lists the size of training data with candidates generated by T5GEC. We denote α for training and prediction as α_{train} and α_{pred}, respectively. The threshold (β) for the BTR and RoBERTa was tuned together with α.

$F_{0.5}$ **Threshold(β)** α_{train}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0	**45.34**	49.36									
5	49.14	49.10	49.64	49.86	**49.92**	49.87	49.61	49.19	49.37	49.36	
10	48.84	49.50	49.62	50.09	**50.10**	50.07	49.96	49.91	49.91	49.57	49.36
20	49.13	49.42	49.74	50.08	**50.22**	49.89	50.00	49.92	49.62	49.46	49.36

Table 4.8: Results of tuning α_{train} for BTR. α_{pred} was fixed to 5. The highest $F_{0.5}$ score on the CoNLL-13 corpus for each α_{train} among different threshold is shown in bold. The scores that were the same as those of the base model ($\beta = 1$) were ignored and greyed out.

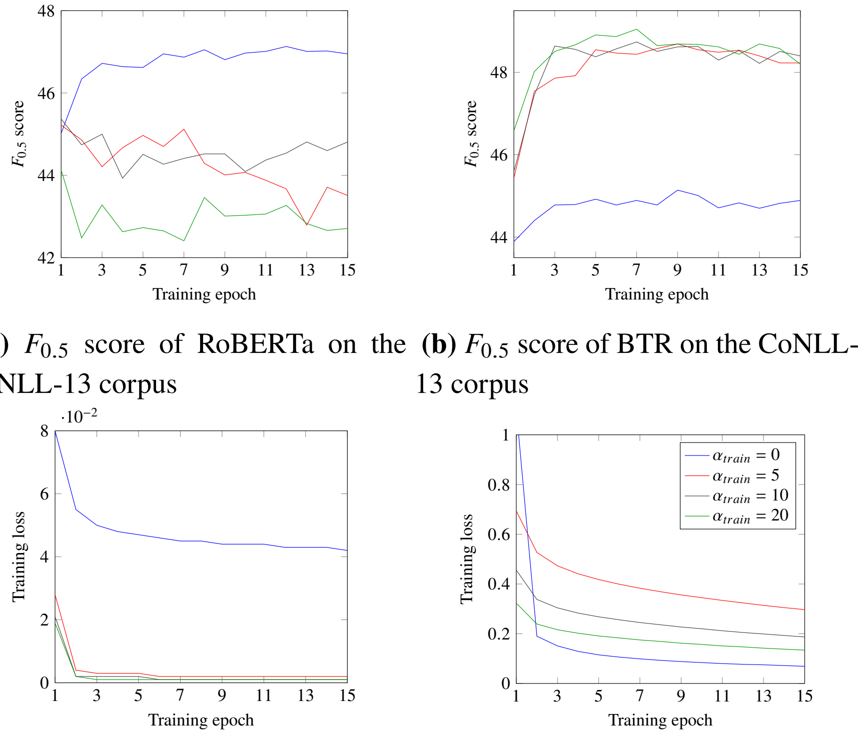
$F_{0.5}$ **Threshold(β)** α_{train}	0	0.1	0.2, . . . , 1
0	46.48	**49.35**	49.36
5	44.89	**49.38**	49.36
10	45.68	**49.38**	49.36
20	41.91	**49.38**	49.36

Table 4.9: Results of tuning α_{train} for RoBERTa. α_{pred} was fixed to 5. The highest $F_{0.5}$ score on the CoNLL-13 corpus for each α_{train} among different threshold is shown in bold. The scores that were the same as those of the base model ($\beta = 1$) were ignored and greyed out.

When tuning $\alpha_{train} \in \{0, 5, 10, 20\}$ for the BTR, α_{pred} was fixed to 5.[1] Because the BTR with $\beta = 0.4$ and $\alpha_{train} = 20$ achieved the highest score as shown in Table 4.8, α_{train} was fixed to 20, this BTR was also used to tune $\alpha_{pred} \in \{5, 10, 15, 20\}$. When tuning $\alpha_{train} \in \{0, 5, 10, 20\}$ for RoBERTa, α_{pred} was fixed to 5. The results in Tables 4.8 and 4.9 indicate the different distributions of $F_{0.5}$ score between RoBERTa and the BTR. To investigate the reason, we compared the training loss and $F_{0.5}$ score of RoBERTa with the BTR. Figure 4.4 shows the comparison. Different from the BTR, when using negative sampling ($\alpha_{train} > 0$) for training RoBERTa, the $F_{0.5}$ score on the CoNLL-13 corpus decreased with the epoch increasing. The training loss of RoBERTa also dropped suddenly after finishing the first epoch. This result suggests that negative sampling in the GEC task for an encoder-only structure leads in the wrong direction in learning representations from the concatenated source and target. And therefore, we fixed α_{train} to 0 for RoBERTa. This RoBERTa was also used to tune $\alpha_{pred} \in \{5, 10, 15, 20\}$. The results in Tables 4.10, 4.11, and 4.12 show that when α_{pred} was set to 5, the BTR, R2L, RoBERTa, and BERT attained their highest scores on the CoNLL-13 corpus. Building upon these results, in the following subsections, we set α_{train} to 20, 0 for the BTR and RoBERTa, respectively, and α_{pred} was set to 5 for all rerankers.

Tables 4.8 and 4.10 also show the performances of the BTR concerning β on the CoNLL-13 corpus with candidates generated by T5GEC. Without using any candidate for training, the BTR($\beta = 0$) could achieve the highest $F_{0.5}$ score. When using 20

[1] Setting α to 0 indicates training with only gold data.



(a) $F_{0.5}$ score of RoBERTa on the CoNLL-13 corpus

(b) $F_{0.5}$ score of BTR on the CoNLL-13 corpus

(c) Training loss of RoBERTa

(d) Training loss of BTR

Figure 4.4: Performances of BTR and RoBERTa with various α_{train} without β during fine-tuning. α_{pred} was fixed to 5 with candidates from T5GEC. Both $F_{0.5}$ score and training loss were averaged over the four trials.

$F_{0.5}$ **Threshold**(β) α_{pred}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
5	49.13	49.42	49.74	50.08	**50.22**	49.89	50.00	49.92	49.62	49.46	49.36
10	48.92	49.34	**50.01**	49.85	49.85	49.51	49.71	49.62	49.49	49.39	49.40
15	48.91	49.22	**49.65**	49.36	49.21	49.18	49.04	49.08	48.90	48.92	48.88
20	36.50	36.83	38.21	38.85	40.24	41.84	43.11	44.41	45.65	**46.87**	49.40

Table 4.10: Results of tuning α_{pred} for BTR. The highest $F_{0.5}$ score on the CoNLL-13 corpus for each α_{pred} among different threshold is shown in bold. The scores that were same as those of the base model ($\beta = 1$) were ignored and greyed out.

candidates for training, the BTR ($\beta = 0.4$) achieved the highest $F_{0.5}$ score of 50.22. Table 4.13 shows the BTR ($\alpha_{train} = 20, \beta = 0.8$) achieved the highest $F_{0.5}$ score on the CoNLL-13 dataset with the candidates generated by T5GEC(large). Thus, in the following subsections, our tuned β for the BTR was set to 0.2 when $\alpha_{train} = 0$. When $\alpha_{train} = 20$, β was set to 0.4 and 0.8 for the candidates generated by T5GEC and T5GEC(large), respectively. Similarly, when $\alpha_{train} = 0$, our tuned β for RoBERTa was set to 0.1 for the two versions of candidates.

$F_{0.5}$ **Threshold(β)** α_{pred}	0	0.1	0.2, . . . , 1
5	46.48	**49.35**	49.36
10	**46.08**	49.40	
15	**45.04**	48.88	
20	**44.28**	49.40	

Table 4.11: Results of tuning α_{pred} for RoBERTa. The highest $F_{0.5}$ score on the CoNLL-13 corpus for each α_{pred} among different threshold is shown in bold. The scores that were same as those of the base model ($\beta = 1$) were ignored and greyed out.

α_{pred}	**R2L**	**BERT**
5	**50.02**	**42.44**
10	49.94	40.53
15	49.85	39.98
20	39.81	39.37

Table 4.12: Results of tuning α_{pred} for R2L and BERT. The highest $F_{0.5}$ score on the CoNLL-13 corpus for each reranker among different α_{pred} is shown in bold.

Model	β										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
RoBERTa	47.90	**50.76**					50.79				
BTR	49.44	50.17	50.00	49.98	49.98	50.58	50.47	50.92	**51.00**	50.82	50.79

Table 4.13: Results of RoBERTa and BTR on the CoNLL-13 corpus with candidates generated by T5GEC (large). The scores that were the same as those of the base model ($\beta = 1$) were ignored and greyed out.

Model	CoNLL-13			CoNLL-14			BEA			JFLEG
	p	r	$F_{0.5}$	p	r	$F_{0.5}$	p	r	$F_{0.5}$	GLEU
Oracle	65.50	33.71	55.11	73.74	51.38	67.87	-	-	-	61.13
T5GEC*	-	-	-	-	-	65.13	-	-	69.38	-
T5GEC	59.19	29.65	49.36	71.27	48.37	65.11	73.96	59.45	70.51	59.04
R2L	**60.94**	29.14	50.02	**71.87**	46.81	64.92	**75.51**	58.69	**71.42**	58.93
w/o L2R	59.56	28.97	49.19	71.36	46.68	64.54	73.51	57.96	69.76	58.69
BERT	44.53	**35.74**	42.44	55.93	**53.18**	55.36	49.91	**64.37**	52.26	55.69
RoBERTa (β = 0.1) w/o α_{train}	59.20	29.63	49.35	71.14	48.42	65.04	74.04	59.37	70.55	59.17
w/o α_{train}, β	54.83	28.88	46.48	65.64	47.24	60.90	65.85	57.71	64.05	57.49
BTR (β = 0.4)	59.87	30.54	**50.22**	71.62	48.74	**65.47**	74.68	60.27	71.27	59.17
w/o β	58.10	30.37	49.13	69.52	48.07	63.82	72.69	60.71	69.93	**59.52**
w/o α_{train}, β	51.30	30.94	45.34	62.83	49.03	59.48	64.35	60.74	63.60	57.62

Table 4.14: Results for the models on each dataset with candidates from T5GEC. * indicates the score presented in Rothe et al. [2021]. Bold scores represent the highest (p)recision, (r)ecall, $F_{0.5}$, and GLEU for each dataset.

Model	CoNLL-13 ($F_{0.5}$)	CoNLL-14 ($F_{0.5}$)	BEA ($F_{0.5}$)	JFLEG (GLEU)
R2L	50.02 ± 0.15	64.96 ± 0.10	**71.42** ± 0.05	59.09 ± 0.19
RoBERTa (β = 0.1) w/o α_{train}	48.66 ± 1.20	63.96 ± 1.90	68.90 ± 2.88	58.68 ± 0.66
BTR (β = 0.4)	**50.05** ± 0.07	**65.29** ± 0.19	70.84 ± 0.05	**59.12** ± 0.07

Table 4.15: The mean ± std results on each dataset with candidates from T5GEC. Bold scores are the highest mean for each dataset.

4.5.2 Effect of Reranking

Table 4.14 presents our main results.[2] While reranking by R2L yielded the highest $F_{0.5}$ score of 71.42 on the BEA test set, it yielded only lower $F_{0.5}$ and GLEU scores than the BTR (β = 0.4) on CoNLL-14 and JFLEG test sets. Meanwhile, the improvements brought by R2L depended on the beam searching score from L2R, suggesting that the unidirectional representation offers fewer gains compared to the bidirectional representation from the BTR. Reranking candidates by BERT resulted in the lower $F_{0.5}$ and GLEU scores than T5GEC. This may be because BERT considers only the target sentence and ignores the relationship between the source and the target. The BTR (β = 0.4) achieved an $F_{0.5}$ score of 71.27 on the BEA test set. On the CoNLL-14 test set, the BTR (β = 0.4) attained the highest $F_{0.5}$ score of 65.47, with improvements of 0.36 points from T5GEC. The use of the threshold and negative candidates played an important role in the BTR. Without these two mechanisms, the BTR achieved only 59.48 and 63.60 $F_{0.5}$ scores, respectively, on the CoNLL-14 and BEA test sets, which were lower than those of the original selection. In the meantime, the BTR without the threshold could achieve the highest GLEU score of 59.52 on the JFLEG corpus, which indicates β = 0.4 is too

[2] The mean and standard deviation results of the BTR, R2L, and RoBERTa are listed in Table 4.15.

Model	Level	Missing	Replacement	Unnecessary	All
T5GEC	A	62.30	**69.92**	73.74	68.40
	B	73.99	67.94	78.26	70.93
	C	78.54	71.51	85.16	75.54
	N	80.66	**69.48**	53.78	71.36
	All	71.23	69.47	74.30	70.51
BTR (β = 0.4)	A	**63.65**	69.86	**74.40**	**68.76**
	B	**74.81**	**68.94**	**79.01**	**71.84**
	C	**81.85**	**72.61**	**86.00**	**77.36**
	N	**83.17**	68.23	**57.88**	**72.01**
	all	**72.91**	**69.69**	**75.69**	**71.27**

Table 4.16: Results for each operation type with classified CEFR levels on the BEA test set with candidates from T5GEC. Edit operations are divided into *Missing*, *Replacement*, and *Unnecessary* corresponding to inserting, substituting, and deleting tokens, respectively. Bold scores are the highest $F_{0.5}$ for each operation with the corresponding level.

Model	PUNCT	DET	PREP	ORTH	SPELL
T5GEC	74.62	77.57	73.33	70.32	78.38
BTR (β = 0.4)	**75.73**	**79.08**	**73.77**	**70.72**	**78.87**

Table 4.17: Results for the top five error types on the BEA test set. Bold scores are the highest $F_{0.5}$ for each error type. Error types *PUNCT*, *DET*, *PREP*, *ORTH*, and *SPELL* are corresponding to punctuation, determiner, preposition, orthography, and spelling errors, respectively.

high for the JFLEG corpus. This is because of the different distributions and evaluation metrics between the CoNLL-13 and JFLEG corpus, as proved in Section 4.5.7. Compared to RoBERTa (β = 0.1) w/o α_{train} of the encoder-only structure, the BTR (β = 0.4) can achieve higher $F_{0.5}$ scores on CoNLL-13, 14, and BEA test sets, and a competitive GLEU score on the JFLEG corpus. These results show the benefit of using the Transformer with the encoder-decoder architecture in the BTR.

4.5.3 Evaluation with Respect to CEFR Levels and Error Types

Tables 4.16 and 4.17 present more details for different CEFR levels and error types. Compared with A (beginner) level sentences, the BTR was more effective for B (intermediate), C (advanced), and N (native) level sentences. As shown in Table 4.17, the BTR (β = 0.4) improved T5GEC for all top-5 error types. Furthermore, as presented in Table 4.16, the BTR (β = 0.4) could effectively handle *Missing* and *Unnecessary* tokens but not *Replacement* for the native sentences. It was more difficult to correct the *Replacement* and *Unnecessary* operations in the native sentences for both models compared with the advanced sentences. This may be because the writing style of na-

Model	CoNLL-13			CoNLL-14			BEA			JFLEG
	p	r	$F_{0.5}$	p	r	$F_{0.5}$	p	r	$F_{0.5}$	GLEU
Oracle	66.34	35.34	56.44	76.04	53.36	70.08	-	-	-	63.87
T5GEC (large)*	-	-	-	-	-	66.10	-	-	72.06	-
T5GEC (large)	60.24	31.20	50.79	73.10	**49.76**	66.83	75.65	60.87	72.15	61.88
R2L	**61.55**	30.03	50.87	**73.60**	48.47	66.68	**77.06**	60.24	**72.98**	61.32
RoBERTa (β = 0.1) w/o α_{train}	60.22	31.17	50.76	73.12	**49.76**	**66.85**	75.74	60.83	72.20	61.85
BTR (β = 0.8)	60.54	**31.28**	**51.00**	72.71	**49.76**	66.57	75.91	**61.13**	72.41	**61.97**

Table 4.18: Results for the models on each dataset with candidates generated by T5GEC (large). * indicates the score presented in Rothe et al. [2021]. Bold scores represent the highest (p)recision, (r)ecall, $F_{0.5}$, and GLEU for each dataset.

tive speakers is more natural and difficult to correct with limited training data, whereas language learners may tend to use a formal language to make the correction easier.

4.5.4 Reranking High-quality Candidates

Table 4.18 lists the performances when reranking high-quality candidates. While R2L still achieved the highest $F_{0.5}$ score on the BEA test set, it was less effective than the BTR on the JFLEG corpus. Although the BTR (β = 0.8) used only 248M parameters and was trained with the candidates generated by T5GEC, it could rerank candidates from T5GEC (large) and achieve 61.97 GLEU and 72.41 $F_{0.5}$ scores on the JFLEG and BEA test sets, respectively. This finding indicates the sizes of the BTR and the base model do not need to be consistent, and a smaller BTR can also work as a reranker for a larger base model. RoBERTa (β = 0.1) w/o α_{train} achieved the highest $F_{0.5}$ score of 66.85 on the CoNLL-14 corpus with only 0.02-point improvement from T5GEC (large), which reflects the difficulty in correcting uncleaned sentences.

4.5.5 Evaluation on the Cleaned CoNLL Corpus

The original texts of CoNLL-13 and 14 contain several styles of punctuation tokenization, such as "DementiaToday,2012" and "known , a". While these punctuation styles with/without spaces are not considered grammatical errors by a human, they are often identified as errors by automatic GEC scorers. Moreover, while most of the sequences in CoNLL-14 are of sentence-level, several sequences are of paragraph-level owing to the punctuation without spaces. In this chapter, we cleaned the texts of CoNLL-13 and 14 using the "en_core_web_sm" tool in spaCy [Honnibal et al., 2020] so that all punctuation included spaces. The paragraph-level sequences were split into sentences with respect to the position of full stops. The cleaned CoNLL-14 corpus contains 1326 pairs of data.

Tables 4.19, 4.20, and 4.21 show the experimental results on the cleaned CoNLL-14 corpus. Compared to the results on the original CoNLL-14 corpus (Tables 4.14 and 4.18), the Oracle performance on the cleaned version could achieve higher precision

Model	**Precision**	**Recall**	$F_{0.5}$
Oracle	80.62	51.98	72.62
T5GEC	78.01	48.57	69.58
R2L	**78.81**	46.83	69.34
w/o L2R	77.69	46.55	68.52
BERT	58.84	**53.53**	57.70
RoBERTa (β = 0.1) w/o α_{train}	77.86	48.62	69.50
w/o α_{train}, β	71.07	47.36	64.60
BTR (β = 0.4)	78.52	48.82	**70.00**
w/o β	76.02	48.30	68.19
w/o α_{train}, β	67.44	49.45	62.87

Table 4.19: Results for the models on the cleaned CoNLL-14 corpus with candidates from T5GEC. Bold scores represent the highest precision, recall, and $F_{0.5}$.

Model	$F_{0.5}$
R2L	69.36 ± 0.13
RoBERTa (β = 0.1) w/o α_{train}	68.12 ± 2.39
BTR (β = 0.4)	**69.80** ± 0.18

Table 4.20: The mean ± std results on the cleaned CoNLL-14 corpus with candidates from T5GEC. Bold scores represents the highest mean.

Model	**Precision**	**Recall**	$F_{0.5}$
Oracle	82.01	54.19	74.38
T5GEC (large)	79.27	49.91	70.92
R2L	**79.72**	48.71	70.72
RoBERTa (β = 0.1) w/o α_{train}	79.30	49.91	70.94
BTR (β = 0.8)	79.65	**49.98**	**71.20**

Table 4.21: Results for the models on the cleaned CoNLL-14 corpus with candidates from T5GEC (large). Bold scores represent the highest precision, recall, and $F_{0.5}$.

$F_{0.5}$ Threshold(β) / α_{train}, α_{pred}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0, 5	47.44	**52.83**	52.77	52.52	52.51						
0, 10	44.93	**52.65**	52.37								
0, 15	43.74	**52.46**	**52.46**	**52.46**	**52.46**	**52.46**	**52.46**	**52.46**	**52.46**	52.45	
0, 20	31.01	51.86	52.21	52.33	52.38	52.41	52.42	52.44	**52.45**	**52.45**	52.47
5, 5	52.51	53.22	**53.49**	53.41	53.42	53.45	53.40	53.28	53.17	53.06	52.51
5, 10	51.21	53.19	53.52	53.41	53.40	**53.56**	53.34	53.21	53.15	53.04	52.37
5, 15	50.68	53.11	53.37	53.45	**53.54**	53.46	53.30	53.23	53.20	53.10	52.45
5, 20	30.44	32.42	34.86	36.39	38.28	41.33	42.73	43.90	45.16	**46.67**	52.47
10, 5	53.47	53.95	**54.04**	53.95	53.85	53.68	53.64	53.38	53.09	53.01	52.51
10,10	52.51	53.21	**53.99**	53.87	53.70	53.73	54.49	53.30	53.02	52.99	52.37
10,15	52.05	53.97	**54.01**	53.64	53.66	53.63	53.44	53.26	53.03	53.05	52.45
10,20	30.03	31.55	32.76	34.12	36.07	39.25	40.89	42.50	43.68	**45.45**	52.47
20, 5	53.26	**53.87**	53.85	53.75	53.79	53.77	53.70	53.50	53.31	52.98	52.51
20,10	52.37	53.15	53.75	53.77	**53.91**	53.83	53.54	53.44	53.24	53.15	52.37
20,15	52.29	53.69	53.82	**53.85**	53.84	53.64	53.46	53.33	53.21	53.21	52.45
20,20	29.68	31.03	32.11	33.47	35.15	38.52	39.99	41.64	43.25	**44.95**	52.47

Table 4.22: Results of tuning α_{train} and α_{pred} for BTR on the BEA dev set. The highest $F_{0.5}$ score for each pair of α_{train} and α_{pred} among different threshold is shown in bold. The scores that were the same as those of the base model ($\beta = 1$) were ignored and greyed out.

and competitive recall scores, leading to an improved $F_{0.5}$ score. The score distributions on the cleaned corpus among the rerankers are similar to those observed in the original corpus. In particular, the BTR ($\beta = 0.4$) maintains its position as the top-performing reranker, achieving the highest $F_{0.5}$ scores of 70.00 on the cleaned corpus using candidates from T5GEC. Meanwhile, different from the results on the original corpus, the BTR ($\beta = 0.8$) could achieve the highest $F_{0.5}$ score of 71.20 when employing candidates from T5GEC (large), emphasizing the significance of using cleaned text in the process.

4.5.6 Tuning Parameters on JFLEG and BEA dev Datasets

The BEA and JFLEG corpus also provide a dev set with 4384 and 754 sentences for validation, respectively. To determine the optimal α_{train}, α_{pred}, and β for the BTR listed in Table 4.8 on these two datasets, we re-evaluated the performances of the BTR on the corresponding dev sets. Tables 4.22 and 4.23 show the results on the BEA and JFLEG dev sets, respectively. On the BEA dev set, the highest $F_{0.5}$ score of 54.04 was achieved with $\alpha_{train} = 10$, $\alpha_{pred} = 5$, and $\beta = 0.2$. On the JFLEG dev set, the highest GLEU score of 54.46 was achieved with $\alpha_{train} = 5$, $\alpha_{pred} = 15$, and $\beta = 0$. These results demonstrate the differences in evaluating the minimal edit and fluency for grammar corrections. Given the previous $\alpha_{train}, \alpha_{pred}$ and β, we re-evaluated the BTR on the BEA and JFLEG test sets. Table 4.24 lists the results. Tuning hyperparameters on the JFLEG dev set led to a higher GLEU score of 60.14 on the JFLEG test set, compared to the tuned hyperparameters on the CoNLL-13 set. However, tuning hyperparameters on the BEA dev set resulted in a lower $F_{0.5}$ score of 71.12 on the BEA test set, compared

GLEU **Threshold**(β) α_{train}, α_{pred}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0, 5	**52.43**	53.25									
0, 10	**51.91**	53.25									
0, 15	**51.36**	53.25									
0, 20	44.59	**52.97**	53.25								
5, 5	54.35	**54.37**	54.12	54.05	53.81	53.67	53.42	53.32	53.20	53.22	53.25
5, 10	**54.41**	54.34	54.05	53.68	53.48	53.31	53.30	53.26	53.26	53.25	
5, 15	**54.46**	54.44	53.88	53.43	53.33	53.29	53.22	53.22	53.26	53.20	53.25
5, 20	44.42	44.99	45.82	46.32	46.80	47.43	47.73	48.13	48.98	**49.37**	53.25
10, 5	54.15	**54.23**	53.99	53.88	53.69	53.51	53.37	53.28	53.24	53.23	53.25
10,10	**54.23**	54.20	53.93	53.73	53.54	53.37	53.29	53.24	53.24	53.23	53.25
10,15	**54.29**	54.16	53.89	53.57	53.48	53.33	53.26	53.19	53.22	53.23	53.25
10,20	44.22	44.71	45.29	45.70	46.20	47.21	47.45	47.98	48.52	**49.07**	53.25
20, 5	**53.92**	53.87	53.85	53.68	53.60	53.49	53.50	53.38	53.25	53.26	53.25
20,10	**53.92**	53.88	53.65	53.54	53.53	53.42	53.32	53.22	53.23	53.26	53.25
20,15	**54.12**	53.89	53.61	53.42	53.42	53.38	53.28	53.23	53.19	53.22	53.25
20,20	44.37	44.79	45.22	45.56	45.98	46.93	47.36	47.83	48.48	**49.08**	53.25

Table 4.23: Results of tuning α_{train} and α_{pred} for BTR on the JFLEG dev set. The highest GLEU score for each pair of α_{train} and α_{pred} among different threshold is shown in bold. The scores that were the same as those of the base model (β = 1) were ignored and greyed out.

Tuned on corpus	α_{train}	α_{pred}	β	**BEA**	**JFLEG**
CoNLL-13	20	5	0.4	**71.27**	59.17
BEA dev	10	5	0.2	71.12	-
JFLEG dev	5	15	0	-	**60.14**

Table 4.24: Results for the BTR on the BEA and JFLEG test sets with tuned hyperparameters.

to the tuned hyperparameters on the CoNLL-13 set.

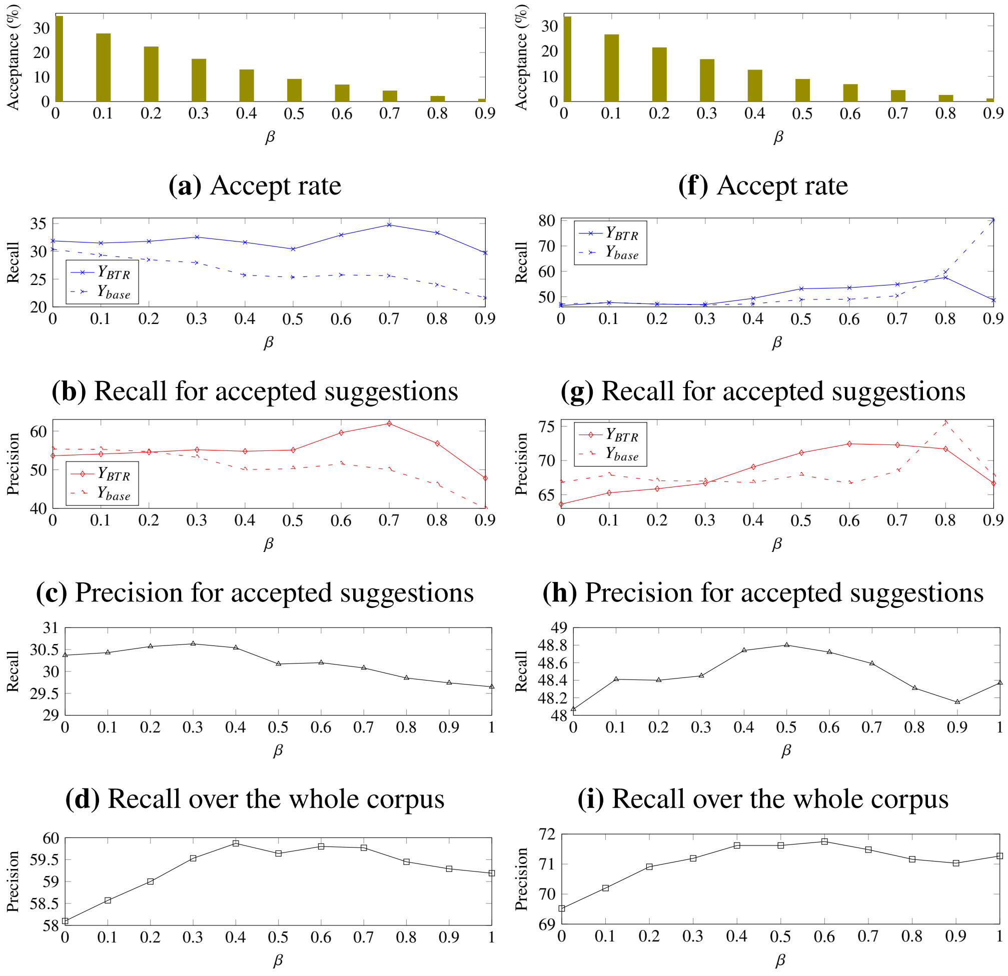
Candidate	**Accept**	**Reject**	**Equal**
Proportion(%)	12.50	21.11	66.39
Y_{base}	61.67	**61.66**†	**68.78**
Y_{BTR}	**63.97**†	57.28	**68.78**

Table 4.25: Results for the BTR (β = 0.4) on CoNLL-14 with candidates from T5GEC. Y_{base} and Y_{BTR} denote the selections by T5GEC and suggestions by the BTR, respectively. † indicates that the difference between Y_{BTR} and Y_{base} is significant with a p-value < 0.05. Bold scores represent the highest $F_{0.5}$ for each case.

4.5.7 Effect of β

Table 4.25 demonstrates the effect of using β. *Equal* denotes the suggestion Y_{BTR} is exactly Y_{base}. *Accept* denotes Y_{BTR} satisfies Section 4.3.3 and Y_{BTR} will be the final selection, while *Reject* denotes Y_{BTR} does not satisfy the equation and Y_{base} is still the final selection. Most of the final selections belonged to *Equal* and achieved the highest $F_{0.5}$ score of 68.78. This indicates the sentences in *Equal* can be corrected easily by both the BTR (β = 0.4) and T5GEC. Around 1/3 of the new suggestions proposed by the BTR (β = 0.4) were accepted and achieved an $F_{0.5}$ score of 63.97, which was a 2.3-point improvement from Y_{base}. However, around 2/3 of the new suggestions were not accepted, and the original selection by T5GEC resulted in a higher $F_{0.5}$ score than these rejected suggestions. These results show that, among the new suggestions, the BTR was confident only for some suggestions. The confident suggestions tended to be more grammatical, whereas the unconfident suggestions tended to be less grammatical than the original selections.

To investigate the role of β, we also analyzed the relationship between β and the corresponding precision, recall, and GLEU scores. Figure 4.5 shows the performance of the BTR ($\alpha_{train} = 20, \alpha_{pred} = 5$) on the CoNLL-13 and 14 corpus, respectively. With β increasing, the acceptance rate, i.e., the percentage of suggestions that the BTR accepts, decreases while the precision and recall for the *Accept* suggestions increases. This demonstrates our assumption in Section 4.3.3 that the value of $f(Y|X)$ indicates the confidence of the BTR. It also reaffirms our previous findings, indicating that suggestions with higher confidence tend to be more grammatical than the original selections. As for the whole corpus, when $\beta = 0.7$, this BTR achieved lower precision and recall score than $\beta = 0.4$ due to the limited amount of $Accept$ suggestions. Figure 4.6 shows the performance of BTR ($\alpha_{train} = 10, \alpha_{pred} = 5$) on the BEA dev and test corpus, respectively. In Figure 4.6, the BTR shows a similar performance to that on the CoNLL-13 and 14, where a larger β leads to higher precision and recall for $Accept$ suggestions. However, the performance over the whole corpus also depends on the acceptance rate. Differently, as shown in Figure 4.7, the experimental results of the BTR ($\alpha_{train} = 5, \alpha_{pred} = 15$) on the JFLEG corpus achieved the highest GLEU score for the whole corpus when $\beta \leq 0.1$. This may be because using $\alpha_{pred} = 15$ makes a flatter



(a) Accept rate

(f) Accept rate

(b) Recall for accepted suggestions

(g) Recall for accepted suggestions

(c) Precision for accepted suggestions

(h) Precision for accepted suggestions

(d) Recall over the whole corpus

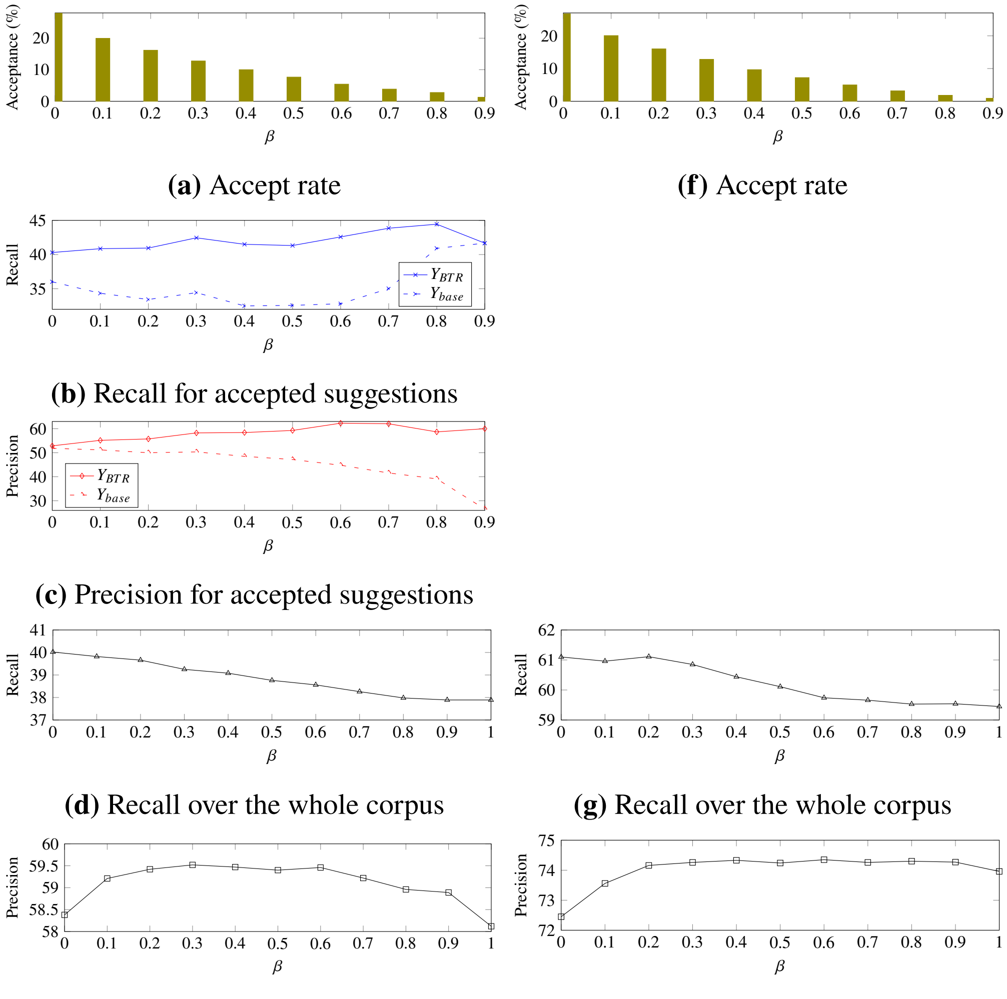
(i) Recall over the whole corpus

(e) Precision over the whole corpus

(j) Precision over the whole corpus

Figure 4.5: Precision and recall of BTR ($\alpha_{train} = 20, \alpha_{pred} = 5$) with respect to different β on the CoNLL-13 set (left, a - e) and CoNLL-14 set (right, f - j), respectively.

probability than $\alpha_{pred} = 5$ as shown in Figure 4.8. Besides, recognizing the fluency of a sentence by the BTR may be easier than recognizing the minimal edit of corrections.



(a) Accept rate

(f) Accept rate

(b) Recall for accepted suggestions

(c) Precision for accepted suggestions

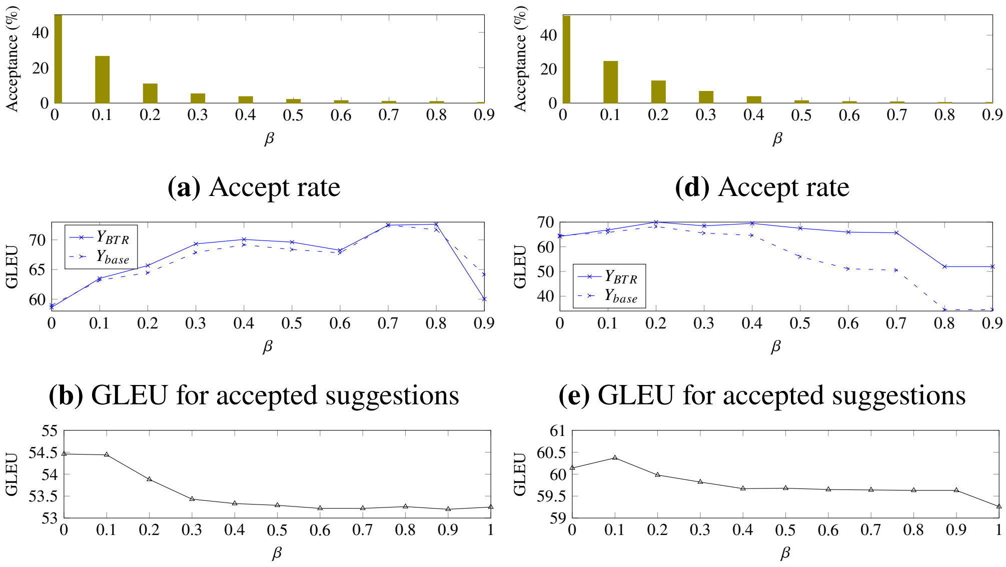
(d) Recall over the whole corpus

(g) Recall over the whole corpus

(e) Precision over the whole corpus

(h) Precision over the whole corpus

Figure 4.6: Precision and recall of BTR ($\alpha_{train} = 10, \alpha_{pred} = 5$) with respect to different β on the BEA dev (left, a - e) and test (right, f - h) set.



(a) Accept rate

(d) Accept rate

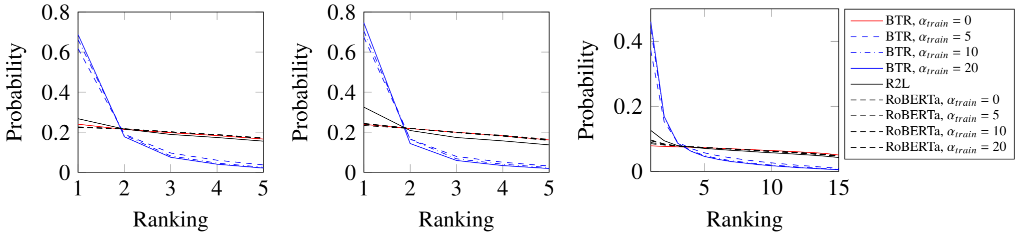
(b) GLEU for accepted suggestions

(e) GLEU for accepted suggestions

(c) GLEU over the whole corpus

(f) GLEU over the whole corpus

Figure 4.7: GLEU of BTR ($\alpha_{train} = 5, \alpha_{pred} = 15$) with respect to different β on the JFLEG dev (left, a - c) and test (right, d - f) set.



(a) CoNLL-14

(b) BEA

(c) JFLEG

Figure 4.8: Average probability for each rank on the CoNLL-14 (a), BEA (b) and JFLEG (c) test set. The top candidate sentences were generated by T5GEC.

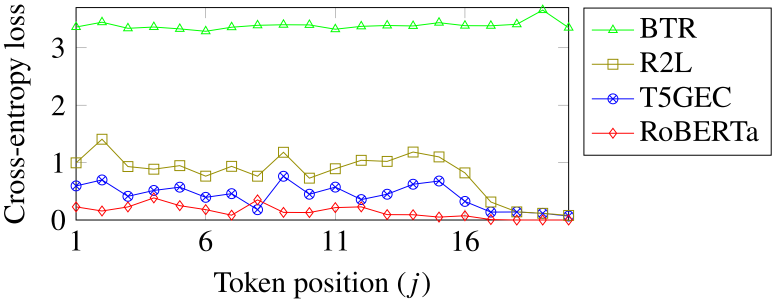


Figure 4.9: Cross-entropy loss of y_j versus j. The loss was averaged over CoNLL-14's 149 tokenized utterances with length in interval $[18, 20]$ (including <eos>).

4.5.8 Difference Among Rerankers

To investigate the difference among R2L, RoBERTa ($\beta = 0.1$) w/o α_{train}, and the BTR ($\beta = 0.4$), we compared the precision and recall of the three rerankers in Table 4.14. In most cases, R2L tended to improve the precision but lower the recall from T5GEC. The improvements brought by RoBERTa from T5GEC for both precision and recall are limited. Meanwhile, the BTR could improve both precision and recall from the original ranking. Because T5GEC already achieved a relatively high precision and low recall, there was more room to improve recall, which was demonstrated by the BTR.

Figure 4.8 shows the probability distribution of reranking. When $\alpha_{train} > 0$, the probability distribution of the BTR becomes peaked, which indicates that using Equation (4.3.3) to minimize the unlikelihood could increase the probability gap between the 1st-ranked candidate and the rest. Compared with the BTR, when $\alpha_{train} > 0$, the probability distribution of RoBERTa is as flat as $\alpha_{train} = 0$, which suggests the effectiveness of the encoder-decoder structure compared with the encoder-only one when minimizing unlikelihood.

Figure 4.9 shows both T5GEC and R2L have a relatively high cross-entropy loss for tokens at the beginning positions and a low loss for tokens at the ending positions, even though the loss of R2L was the sum of two opposite decoding directions. This may be because the learning by the auto-regressive models for the latest token was over-fitting and for the global context was under-fitting, as Qi et al. [2020] indicated. RoBERTa has a flatter loss with less sharp positions than T5GEC and R2L. Meanwhile, the BTR has a flat loss, which is ideal for reranking candidate sentences with length normalization, as suggested by Salazar et al. [2020]. Figure 4.4 also presents the difference between RoBERTa and the BTR in training loss.

Table 4.1 provides examples of ranked outputs by T5GEC, R2L, RoBERTa w/o α_{train} ($\beta = 0.1$), and BTR ($\beta = 0.4$). The first block of output results demonstrates the difficulty of correcting spelling errors. In this block, the BTR outputs the token "insensitively" with the correct spelling but a mismatched meaning, whereas other rerankers tend to keep the original token "intesively" with a spelling error. The examples in the second block show that both the BTR and R2L are capable of correctly addressing verb tense

Model	CoNLL-13	CoNLL-14	BEA		JFLEG	
			dev	test	dev	test
T5GEC	778	790	3,638	3,776	451	444
BERT	22	21	68	69	12	13
R2L	34	32	108	109	19	19
RoBERTa	82	88	333	386	46	69
BTR	194	199	740	738	113	122

Table 4.26: Time cost (seconds) in inference over the whole corpus with 5 candidates generated by T5GEC.

errors. The examples in the third block show the methods utilized by various rerankers for handling count-mass noun errors. The examples in the last block show that even though the BTR recognizes the missing determiner "the" for the word "Disadvantage", it still misses a that-clause sentence.

In inference, we required all rerankers to compute one target sequence at a time to estimate the time cost. For RoBERTa and the BTR, we rearranged the given target sequence by masking each token. These rearranged sequences were then put into a mini-batch for parallel computation. For T5GEC, given the source sentence, we used the mini-batch with a size of 5 to parallelly compute all beams. Table 4.26 displays the time cost for each model to estimate scores over the entire corpus with 5 candidates, using one NVIDIA A100 80GB GPU. We only calculated the time for estimating probability and ignored the time for loading the model and dataset. T5GEC costs the most time among all rerankers, as it predicts tokens of the target sequence one by one. RoBERTa and the BTR took longer than BERT and R2L due to the target sequence rearrangement procedure. The BTR took 2 to 3 times as much as RoBERTa due to the additional decoder structure.

4.6 Conclusion

This chapter proposed a *bidirectional Transformer reranker* (BTR) as an extension of the LMR. The BTR is designed to rerank the top candidates generated by a pre-trained seq2seq model for GEC. For a fully pre-trained model, T5-base, the BTR could achieve 65.47 and 71.27 $F_{0.5}$ scores on the CoNLL-14 and BEA test sets. Our experimental results showed that the BTR on top of T5-base with limited pre-training steps could improve both precision and recall for candidates from T5-base. Because using negative sampling for the BTR generates a peaked probability distribution for ranking, introducing a threshold β benefits the acceptance of the suggestion from the BTR. This approach showcases the effectiveness of contrastive learning in mitigating issues such as text degeneration. Moreover, the BTR takes into account each position equally during prediction, leading to enhanced overall performance and a comprehensive understanding of the entire context compared to T5GEC and R2L. This highlights the potential of addressing exposure bias by utilizing bidirectional representations of the target context. Furthermore, the BTR on top of T5-base could rerank candidates generated from T5-large and yielded better performance. This finding suggests the effectiveness of the BTR even in experiments with limited GPU resources. While the BTR in our experiments lacked sufficient pre-training, it should further improve the performance with full pre-training for reranking in future.

Chapter 5

Conclusion

5.1 Conclusion

This thesis focuses on investigating the effectiveness of using pseudo-log-likelihood scores (PLL) as rerankers in natural language processing tasks. The experimental results obtained from three specific tasks, namely discourse segmentation, discourse parsing, and grammatical error correction, provide compelling evidence of the effectiveness of PLL-based models for improving the quality of predictions generated by discriminative models. In fact, these models employed in the experiments outperformed existing approaches and achieved state-of-the-art results. These findings yield three significant insights that can guide future research in constructing PLL-based models and their applications in addressing exposure bias and text degeneration.

The first insight focuses on the availability and benefits of constructing PLL-based models from pre-trained language or seq2seq models. In the discourse segmentation and discourse parsing tasks, the proposed PLL-based model, LMR, combines each source and prediction pair into a new sequence, allowing a pre-trained language model to rerank these combined sequences as a generative model. Additionally, the LMR utilizes dictionary definitions to generate representations for unseen labels, promoting the application of pre-trained representations in these tasks, as evidenced by the conducted experiments. Similarly, in the grammatical error correction task, the proposed PLL-based model, BTR, treats the source as background information and focuses on reranking the predictions. This sequence-to-sequence setting aligns well with the architecture of pre-trained sequence-to-sequence models, allowing effective utilization of their capabilities. The benefits brought by pre-trained models to the natural language processing community are notable. Researchers and practitioners could employ the existing advancements in pre-trained models without extensive retraining or starting from scratch, because the pre-trained models have already learned from vast amounts of data. This is particularly advantageous in situations where labeled data is scarce or expensive to obtain, allowing the constructed PLL-based models to effectively tackle challenges related to domain shift and limited training data.

The second insight arising from the experimental results demonstrates that text degeneration can be mitigated by using a reranking strategy. In contrast to discriminative models that rely solely on the source text, using reranking procedure helps PLL-based models take advantage of additional information from predictions to more accurately rank candidates. The use of negative sampling during the training of the BTR helps create a more peaked probability distribution among candidates, ensuring that a probability threshold can be applied to accept only the most convincing selections from the BTR. This procedure effectively assists to prioritize natural text over repetitive or awkward ones, resulting in outputs that exhibit improved fluency, coherence, and overall quality.

The final insight derived from the experimental findings highlights the potential of addressing exposure bias by using bidirectional representations of the target context. Auto-regressive-based discriminative models, which rely on past timestep representations to predict the next token, face challenges in accurately predicting tokens in early timesteps due to the lack of available context and the presence of a large number of unfactorized future timesteps. The proposed PLL-based models effectively overcome these limitations by factorizing both past and future steps, allowing them to equally consider each position for prediction and enhance overall performance with a comprehensive understanding of the entire context. To achieve this, this thesis employs two techniques: masked language modeling and pseudo-log-likelihood. These techniques enable the models to estimate the target sentence probability by predicting each corrupted token using surrounding bidirectional representations, which encompass both past and future timesteps. With such a comprehensive and balanced understanding of the context, PLL-based models can select accurate and contextually appropriate outputs.

5.2 Future Work

Chapter 3 and Chapter 4 of this thesis have proved the effectiveness of our various attempts to improve the applications of PLL-based models. Guided by the experimental results, this section concludes the thesis by exploring potential future directions that could benefit the natural language processing community.

5.2.1 Usage of Descriptions

In Chapter 3, we investigate the use of descriptions, specifically dictionary definitions, as additional information to construct label embeddings. Two methods, namely Average and Concatenate, are employed to represent these embeddings. The effectiveness of these methods is demonstrated through experimental results, which involve a comparison to a setting where no additional information is utilized.

As mentioned by Wang et al. [2018a], label embeddings have been extensively applied for classification tasks, including image classification and text recognition in images, to improve the prediction. Besides, label embeddings are particularly beneficial in zero-shot tasks, where certain classes have not been encountered previously.

The Average method in this thesis is a quite simple component-wise mean function and the Concatenate method may suffer from computational costs when dealing with long text. To benefit classification tasks, a promising direction for future research is to explore more efficient techniques to incorporate these descriptions for unseen tokens, particularly in scenarios involving limited computational resources.

5.2.2 Pre-training and Masking Strategies

In Chapter 4, we adopt the span masking strategy based on the evidence from Raffel et al. [2020] as the optimal strategy for pre-training the Transformer model (referred to as T5) with 34B tokens. Following the approach of Raffel et al. [2020], we apply this strategy to pre-train the bidirectional Transformer (referred to as BTR) with 10.5B tokens on a smaller-sized corpus. During the fine-tuning process, we adhere to the BERT setting of randomly masking 15% of the target elements, with the masked tokens distributed according to an 8:1:1 masking strategy. After fine-tuning, we observe that using BTR to rerank predictions from T5 resulted in improved final results.

It is worth noting that, until now, there has not been a fully pre-trained seq2seq model with a BERT-style self-attention mechanism in the decoder. Our findings emphasize the potential advantages of a fully pre-trained bidirectional Transformer for natural language processing. As a result, it would be worthwhile to explore other suitable strategies for pre-training the bidirectional Transformer, in addition to the span masking strategy. Additionally, future research should consider the scale of pre-training, the optimal mask ratio, and the distribution of masked tokens as important factors in order to further enhance the model's performance.

5.2.3 Learning with Negative Samples

The experimental results presented in Chapter 3 and Chapter 4 demonstrate that implementing the contrastive learning to distinguish positive and negative samples, resulting in improved performance when using our PLL-based models as rerankers. Particularly, the BTR model exhibits a notable characteristic of generating a peaked probability distribution among candidate outputs.

Our thesis and the work of Welleck et al. [2020] both employ unlikelihood training to minimize the likelihood of improbable generations while maximizing the likelihood of true target outputs. A notable contribution from Liu et al. [2022] introduces an interesting perspective that sentence probabilities should be correlated with their quality. According to their proposal, the model should assign higher probabilities to higher-quality candidate outputs. By incorporating such an insight, future research can focus on designing PLL-based models that not only prioritize generating plausible outputs but also emphasize the quality and relevance of the generated content.

Acknowledgement

My sincere appreciation goes out to everyone who has contributed to my academic and personal development.

First of all, I would like to express my deepest gratitude to Professor Okumura Manabu for providing me with the opportunity to join our lab. Over the past six years, Professor Okumura Manabu has been incredibly patient and supportive, sharing his vast knowledge and experience as a researcher and advisor. His guidance has played a pivotal role in my growth as a researcher, teaching me valuable skills such as writing high-quality papers and designing rigorous experiments.

I am also grateful to Professor Hidetaka Kamigaito for his enthusiasm in discussing research topics with me, patiently explaining complex mathematical problems, and providing valuable career advice. His encouragement motivates me to continue my research career, especially during times when my papers faced multiple rejections.

I would like to extend my thanks to Professor Takahiro Shinozaki, Professor Itsuo Kumazawa, Professor Kotaro Funakoshi, and Professor Minoru Nakayama for serving on my doctoral committee, careful reviewing this thesis, and providing valuble advice during my presentation. I am particularly grateful to Professor Kotaro Funakoshi for his suggestions on my research, which greatly influenced my experimental results.

I am grateful to Dr. Tomoya Iwakura for welcoming me as an intern at Fujitsu laboratory. I would also like to thank Dr. Makino Takuya, my mentor at Fujitsu laboratory, for engaging in discussions with me about language models . My internship experience at Fujitsu laboratory was quite valuable, and I extend my gratitude to all the members for their warm support and guidance during that time.

I would also like to express my appreciation to my fellow lab members. In particular, thank you to Dr. Naoki Kobayashi for teaching me how to use PyTorch and generously sharing your code. Thank you to Professor Hiroya Takamura and Tatsuya Aoki for providing valuable advice on my papers and experiments. Thank you to Thodsaporn Chayintr for inviting me to go to the gym together, promoting my physical well-being. Thank you to Dongyuan Li for sharing your writing skills. And thank you to Dr. Chenlong HU, Guolin Cao, Haitao Li, Jia Li, Wenting Sun, and Dr. Yuken Feng for your suggestions and encouragement regarding both my personal and professional life.

Lastly, I would like to express my heartfelt gratitude to my mom for her unwavering bravery and resilience in battling illness. I am thankful to my entire family for their constant support, understanding, and encouragement throughout my journey, despite being far away from home for over ten years.

Bibliography

Peter Atkins, Peter William Atkins, and Julio de Paula. *Atkins' physical chemistry*. Oxford university press, 2014.

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning, 2023.

Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952. doi: 10.1073/pnas.38.8.716. URL `https://www.pnas.org/doi/abs/10.1073/pnas.38.8.716`.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/e995f98d56967d946471af29d7bf99f1-Paper.pdf`.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, mar 2003. ISSN 1532-4435.

Julian Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 24(3):179–195, 1975. ISSN 00390526, 14679884. URL `http://www.jstor.org/stable/2987782`.

Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

Ludwig Boltzmann. Studien uber das gleichgewicht der lebenden kraft. *Wissenschafiliche Abhandlungen*, 1:49–96, 1868.

John S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Françoise Fogelman Soulié and Jeanny Hérault, editors, *Neurocomputing*, pages 227–236, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg. ISBN 978-3-642-76153-9.

GLENN W. BRIER. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1 – 3, 1950. doi: 10.1175/1520-0493(1950)078⟨0001:VOFEIT⟩2.0.CO;2. URL https://journals.ametsoc.org/view/journals/mwre/78/1/1520-0493_1950_078_0001_vofeit_2_0_co_2.xml.

Christopher Bryant, Mariano Felice, and Ted Briscoe. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1074. URL https://aclanthology.org/P17-1074.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4406. URL https://aclanthology.org/W19-4406.

Lynn Carlson and Daniel Marcu. Discourse tagging reference manual. *ISI Technical Report ISI-TR-545*, 2001. URL http://www.isi.edu/~marcu/discourse/.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurovsky. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, 2001. URL https://aclanthology.org/W01-1605.

Lynn Carlson, Daniel Marcu, and Ellen Okurowski Mary. Rst discourse treebank ldc2002t07. *Philadelphia:Linguistic Data Consortium*, 2002. URL https://catalog.ldc.upenn.edu/LDC2002T07.

Do Kook Choe and Eugene Charniak. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1257. URL https://aclanthology.org/D16-1257.

Council of Europe. *Common European framework of reference for languages: Learning, teaching, assessment*. Cambridge University Press, 2001.

Daniel Dahlmeier and Hwee Tou Ng. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada, June 2012. Association for Computational Linguistics. URL https://aclanthology.org/N12-1067.

Steven Derby, Paul Miller, and Barry Devereux. Analysing word representation from the input and output embeddings in neural network language models. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 442–454, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.conll-1.36. URL https://aclanthology.org/2020.conll-1.36.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Xiaoan Ding, Tianyu Liu, Baobao Chang, Zhifang Sui, and Kevin Gimpel. Discriminatively-Tuned Generative Classifiers for Robust Natural Language Inference. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8189–8202, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.657. URL https://aclanthology.org/2020.emnlp-main.657.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1024. URL https://www.aclweb.org/anthology/N16-1024.

Bryan Eikema and Wilker Aziz. Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4506–4520, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.398. URL https://aclanthology.org/2020.coling-main.398.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082.

Vanessa Wei Feng and Graeme Hirst. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521, Baltimore, Maryland, June 2014a. Association for Computational Linguistics. doi: 10.3115/v1/P14-1048. URL `https://aclanthology.org/P14-1048`.

Vanessa Wei Feng and Graeme Hirst. Two-pass discourse segmentation with pairing and global features, 2014b.

Seeger Fisher and Brian Roark. The utility of parse-derived features for automatic discourse segmentation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 488–495, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://aclanthology.org/P07-1062`.

G.D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973. doi: 10.1109/PROC.1973.9030.

Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

Jiatao Gu, Kyunghyun Cho, and Victor O.K. Li. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1210. URL `https://aclanthology.org/D17-1210`.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. Using discourse structure improves machine translation evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 687–698, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1065. URL `https://aclanthology.org/P14-1065`.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006. doi: 10.1109/CVPR.2006.100.

David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *J. Mach. Learn. Res.*, 1:49–75, sep 2001. ISSN 1532-4435. doi: 10.1162/153244301753344614. URL `https://doi.org/10.1162/153244301753344614`.

Michael Heilman and Kenji Sagae. Fast rhetorical structure theory discourse parsing. *arXiv preprint arXiv:1505.02425*, 2015.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/afdec7005cc9f14302cd0474fd0f3c96-Paper.pdf.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spacy: Industrial-strength natural language processing in python. 2020.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015. URL http://arxiv.org/abs/1508.01991.

Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

Daphne Ippolito, Reno Kriz, João Sedoc, Maria Kustikova, and Chris Callison-Burch. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3752–3762, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1365. URL https://aclanthology.org/P19-1365.

Shafiq Joty, Giuseppe Carenini, and Raymond Ng. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 904–915, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D12-1083.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 486–496, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/P13-1048.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. CODRA: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435, September 2015. doi: 10.1162/COLI_a_00226. URL `https://aclanthology.org/J15-3002`.

Shafiq Joty, Francisco Guzmán, Lluís Màrquez, and Preslav Nakov. Discourse structure in machine translation evaluation. *Computational Linguistics*, 43(4):683–722, December 2017. doi: 10.1162/COLI_a_00298. URL `https://aclanthology.org/J17-4001`.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1055. URL `https://aclanthology.org/N18-1055`.

Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

Masahiro Kaneko, Kengo Hotate, Satoru Katsumata, and Mamoru Komachi. TMU transformer system using BERT for re-ranking at BEA 2019 grammatical error correction on restricted track. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 207–212, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4422. URL `https://aclanthology.org/W19-4422`.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.391. URL `https://aclanthology.org/2020.acl-main.391`.

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*, Hong Kong, 1–3 December 2018. Association for Computational Linguistics. URL `https://aclanthology.org/Y18-1034`.

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 the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242, Hong Kong, China, November 2019.

Association for Computational Linguistics. doi: 10.18653/v1/D19-1119. URL https://aclanthology.org/D19-1119.

Naoki Kobayashi, Tsutomu Hirao, Hidetaka Kamigaito, Manabu Okumura, and Masaaki Nagata. Top-down rst parsing utilizing granularity levels in documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8099–8106, Apr. 2020. doi: 10.1609/aaai.v34i05.6321. URL https://ojs.aaai.org/index.php/AAAI/article/view/6321.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-3250.

Ilia Kulikov, Alexander Miller, Kyunghyun Cho, and Jason Weston. Importance of search and evaluation strategies in neural dialogue modeling. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 76–87, Tokyo, Japan, October–November 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8609. URL https://aclanthology.org/W19-8609.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Yann LeCun and Fu Jie Huang. Loss functions for discriminative training of energy-based models. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pages 206–213. PMLR, 06–08 Jan 2005. URL https://proceedings.mlr.press/r5/lecun05a.html. Reissued by PMLR on 30 March 2021.

Yann LeCun, Sumit Chopra, and Raia Hadsell. A tutorial on energy-based learning. 2006.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

Jiwei Li and Dan Jurafsky. Mutual information and diverse decoding improve neural machine translation. *CoRR*, abs/1601.00372, 2016. URL http://arxiv.org/abs/1601.00372.

Xiang Lin, Shafiq Joty, Prathyusha Jwalapuram, and M Saiful Bari. A unified linear-time framework for sentence-level discourse parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4200, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1410. URL https://www.aclweb.org/anthology/P19-1410.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020. doi: 10.1162/tacl_a_00343. URL https://aclanthology.org/2020.tacl-1.47.

Yixin Liu, Pengfei Liu, Dragomir Radev, and Graham Neubig. BRIO: Bringing order to abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2890–2903, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.207. URL https://aclanthology.org/2022.acl-long.207.

Zhenghao Liu, Xiaoyuan Yi, Maosong Sun, Liner Yang, and Tat-Seng Chua. Neural quality estimation with multiple hypotheses for grammatical error correction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5441–5452, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.429. URL https://aclanthology.org/2021.naacl-main.429.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://aclanthology.org/D15-1166.

Amandla Mabona, Laura Rimell, Stephen Clark, and Andreas Vlachos. Neural generative rhetorical structure parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2284–2295, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1233. URL `https://www.aclweb.org/anthology/D19-1233`.

Eric Malmi, Yue Dong, Jonathan Mallinson, Aleksandr Chuklin, Jakub Adamek, Daniil Mirylenka, Felix Stahlberg, Sebastian Krause, Shankar Kumar, and Aliaksei Severyn. Text generation with text-editing models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*, pages 1–7, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-tutorials.1. URL `https://aclanthology.org/2022.naacl-tutorials.1`.

William C Mann and Sandra A Thompson. Rhetorical structure theory: Toward a functional theory of text organization. 1988.

Clara Meister, Ryan Cotterell, and Tim Vieira. If beam search is the answer, what was the question? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2173–2185, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.170. URL `https://aclanthology.org/2020.emnlp-main.170`.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1096. URL `https://www.aclweb.org/anthology/D16-1096`.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013b. URL `https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf`.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint*

Conference on Natural Language Processing (Volume 2: Short Papers), pages 588–593, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2097. URL https://aclanthology.org/P15-2097.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2037.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1206. URL https://aclanthology.org/D18-1206.

Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/7b7a53e239400a13bd6be6c91c4f6c4e-Paper.pdf.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/W13-3601.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-1701. URL https://aclanthology.org/W14-1701.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.bea-1.16. URL https://aclanthology.org/2020.bea-1.16.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of* *the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL `https://aclanthology.org/N19-4009`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://www.aclweb.org/anthology/P02-1040`.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://www.aclweb.org/anthology/N18-1202`.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.217. URL `https://aclanthology.org/2020.findings-emnlp.217`.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21 (140):1–67, 2020. URL `http://jmlr.org/papers/v21/20-074.html`.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1511.06732`.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.89. URL `https://aclanthology.org/2021.acl-short.89`.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.240. URL https://aclanthology.org/2020.acl-main.240.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/432aca3a1e345e339f35a30c8f65edce-Paper.pdf.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2323. URL https://aclanthology.org/W16-2323.

Kim Sharp and Franz Matschinsky. Translation of ludwig boltzmann ’ s paper “ on the relationship between the second fundamental theorem of the mechanical theory of heat and probability calculations regarding the conditions for thermal equilibrium ” sitzungberichte der kaiserlichen akademie der wissenschaften. mathematisch-naturwissen classe. abt. ii, lxxvi 1877, pp 373-435 (wien. ber. 1877, 76:373-435). reprinted in wiss. abhandlungen, vol. ii, reprint 42, p. 164-223, barth, leipzig, 1909. *Entropy*, 17(4):1971–2009, 2015. ISSN 1099-4300. doi: 10.3390/e17041971. URL https://www.mdpi.com/1099-4300/17/4/1971.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 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)*, pages 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074. URL https://aclanthology.org/N18-2074.

Noah A. Smith. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, May 2011.

Haoyu Song, Yan Wang, Kaiyan Zhang, Wei-Nan Zhang, and Ting Liu. BoB: BERT over BERT for training persona-based dialogue models from limited personalized data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language*

Processing (Volume 1: Long Papers), pages 167–177, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.14. URL `https://aclanthology.org/2021.acl-long.14`.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/c3a690be93aa602ee2dc0ccab5b7b67e-Paper.pdf`.

Pavel Sountsov and Sunita Sarawagi. Length bias in encoder decoder models and a case for global conditioning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1525, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1158. URL `https://aclanthology.org/D16-1158`.

Caroline Sporleder and Mirella Lapata. Discourse chunking and its application to sentence compression. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 257–264, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. URL `https://aclanthology.org/H05-1033`.

Felix Stahlberg and Bill Byrne. On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3356–3362, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1331. URL `https://aclanthology.org/D19-1331`.

Felix Stahlberg and Shankar Kumar. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online, April 2021. Association for Computational Linguistics. URL `https://aclanthology.org/2021.bea-1.4`.

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. Instantaneous grammatical error correction with shallow aggressive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5937–5947, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.462. URL `https://aclanthology.org/2021.acl-long.462`.

Jun Suzuki and Masaaki Nagata. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European*

Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 291–297, Valencia, Spain, April 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/E17-2047`.

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 270–279, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01424-7.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1008. URL `https://www.aclweb.org/anthology/P16-1008`.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `https://aclanthology.org/P10-1040`.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, L ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search for improved description of complex scenes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32 (1), Apr. 2018. doi: 10.1609/aaai.v32i1.12340. URL `https://ojs.aaai.org/index.php/AAAI/article/view/12340`.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, dec 2010. ISSN 1532-4435.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331, Melbourne, Australia, July 2018a. Association for Computational Linguistics. doi: 10.18653/v1/P18-1216. URL https://aclanthology.org/P18-1216.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331, Melbourne, Australia, July 2018b. Association for Computational Linguistics. doi: 10.18653/v1/P18-1216. URL https://www.aclweb.org/anthology/P18-1216.

Yizhong Wang, Sujian Li, and Houfeng Wang. A two-stage parsing method for text-level discourse analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 184–188, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2029. URL https://www.aclweb.org/anthology/P17-2029.

Yizhong Wang, Sujian Li, and Jingfeng Yang. Toward fast and accurate neural discourse segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 962–967, Brussels, Belgium, October-November 2018c. Association for Computational Linguistics. doi: 10.18653/v1/D18-1116. URL https://www.aclweb.org/anthology/D18-1116.

Geoffrey I. Webb. *Naïve Bayes*, pages 713–714. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_576. URL https://doi.org/10.1007/978-0-387-30164-8_576.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. A reranking model for discourse segmentation using subtree features. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 160–168, Seoul, South Korea, July 2012. Association for Computational Linguistics. URL https://aclanthology.org/W12-1623.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.41. URL https://aclanthology.org/2021.naacl-main.41.

Takateru Yamakoshi, Thomas Griffiths, and Robert Hawkins. Probing BERT's priors with serial reproduction chains. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3977–3992, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.314. URL https://aclanthology.org/2022.findings-acl.314.

Yilin Yang, Liang Huang, and Mingbo Ma. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1342. URL https://aclanthology.org/D18-1342.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. Generative and discriminative text classification with recurrent neural networks. *ArXiv*, abs/1703.01898, 2017.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/3e9f0fc9b2f89e043bc6233994dfcf76-Paper.pdf.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Jingyi Zhang and Josef van Genabith. A bidirectional transformer based alignment model for unsupervised word alignment. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 283–292, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.24. URL `https://aclanthology.org/2021.acl-long.24`.

Tianyi Zhang and Tatsunori B. Hashimoto. On the inductive bias of masked language modeling: From statistical to syntactic dependencies. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5131–5146, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.404. URL `https://aclanthology.org/2021.naacl-main.404`.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1014. URL `https://aclanthology.org/N19-1014`.

Zining Zhu, Chuer Pan, Mohamed Abdalla, and Frank Rudzicz. Examining the rhetorical capacities of neural language models. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 16–32, 2020.

Publication

Thesis-related

Journal

- Ying Zhang, Hidetaka Kamigaito, and Manabu Okumura. Bidirectional transformer reranker for grammatical error correction. *Journal of Natural Language Processing*, 2024, Volume 31, Issue 1 (To be appeared).

Conference

- Ying Zhang, Hidetaka Kamigaito, and Manabu Okumura. Bidirectional transformer reranker for grammatical error correction. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3801–3825, Toronto, Canada. Association for Computational Linguistics. URL `https://aclanthology.org/2023.findings-acl.234/`
- Ying Zhang, Hidetaka Kamigaito, and Manabu Okumura. A language model-based generative classifier for sentence-level discourse parsing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2432–2446, Online and Punta Cana, Dominican Republic, November 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.188. URL `https://aclanthology.org/2021.emnlp-main.188.22`

Others

Journal

- Ying Zhang, Hidetaka Kamigaito, Tatsuya Aoki, Hiroya Takamura, and Manabu Okumura. Generic mechanism for reducing repetitions in encoder-decoder models. *Journal of Natural Language Processing*, 2023, Volume 30, Issue 2, Pages 401-431, Online ISSN 2185-8314, Print ISSN 1340-7619. doi: 10.5715/jnlp.30.401. URL `https://www.jstage.jst.go.jp/article/jnlp/30/2/30_401/_article/-char/en`

Conference

- Ying Zhang, Hidetaka Kamigaito, Tatsuya Aoki, Hiroya Takamura, and Manabu Okumura. Generic mechanism for reducing repetitions in encoder-decoder models. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1606–1615, Held Online, September 2021a. INCOMA Ltd. URL `https://aclanthology.org/2021.ranlp-1.180.21`