Towards Efficient, Customizable, and Communal Natural Language Processing

Jungo Kasai

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2023

Reading Committee:
Noah A. Smith, Chair
Yejin Choi
Ludwig Schmidt

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Towards Efficient, Customizable, and Communal
Natural Language Processing

Jungo Kasai

Chair of the Supervisory Committee:
Noah A. Smith
School of Computer Science and Engineering

I advocate for efficient, customizable, and communal approaches to natural language processing (NLP) and artificial intelligence (AI), where people with diverse skill levels and research backgrounds can: build, use, analyze, and evaluate models; collaborate to solve research problems; and accelerate advances in NLP and AI. AI and NLP have made remarkable progress from recent, large-scale training on massive datasets. These technologies are being developed and used by many cross-disciplinary researchers and practitioners. People with scant computer science training—including physicians, translators, and historians—now rely on AI models for work problems that can be solved by using massive amounts of data. This thesis discusses my key contributions to ways to make AI and NLP more accessible to researchers, practitioners, and users. How can we encourage model builders and practitioners to work as a community to broaden the appeal and utility of NLP and AI models across disciplines? How can we make it easier for them to formulate and answer complex real-world questions using these technologies and ensure these models are robustly evaluated?

I first introduce and empirically demonstrate efficient architectures and learning paradigms for state-of-the-art NLP models. More efficient methods will lower the cost of developing and using these models, making them deployable to less-well-funded fields or institutions. I then present an algorithm for flexible and customizable language generation in the areas of collabora-

tive inference between diverse models. This inference method avoids the computationally (and thus financially and environmentally) expensive training process of large models. Lastly, I propose methodologies and interfaces to make model evaluations more transparent, consistent, and reliable. I present a collaborative platform that bridges the modeling and evaluation research communities to enable robust evaluation of AI models.

Acknowledgements

First and foremost, I would like to thank my advisor, Noah A. Smith. Five years ago, Noah found me from among hundreds of Ph.D. applications that the UW Ph.D. program received. In retrospect, the offer from Noah was one of the luckiest things that have ever happened to me. I fully enjoyed the freedom that he gave me and the environment that I experienced in UW CSE. I would also like to express my gratitude to my committee members, Yejin Choi, Ludwig Schmidt, and Shane Steinert-Threlkeld. Despite their busy schedules, they generously offered me their time to improve my job talk and thesis. While I was not officially a student of Yejin's, I received a lot of help from her career advice and research collaborations throughout my Ph.D.

Research internships in industrial labs were an integral part of my five years as a Ph.D. student. These experiences helped me explore new areas, get fresh ideas, and meet amazing researchers whom I continued to collaborate with afterward. I had amazing and diverse experiences from all of my internships, and I am extremely grateful to my internship managers and hosts. Kun Qian and Yunyao Li at IBM Research introduced me to the real-world problem of entity resolution in a world full of databases. They continued to support my Ph.D. journey through an IBM Ph.D. Fellowship. James Cross in Facebook introduced me to the fascinating area of machine translation. Looking back, I had been focusing on more basic research like syntactic parsing, and our machine translation project was a turning point in my research career. James was always there to help me (he was literally sitting next to me in the office!), and he always gave me as much freedom as I wanted. Yizhe Zhang and Yi Mao fully supported me during my internship at Microsoft, which unfortunately happened in the middle of the pandemic. My fully remote internship would not have been such a great experience without them. In my final internship, I worked with Keisuke Sakaguchi and Ronan Le Bras at the Allen Institute for Artificial

Intelligence. I believe that this internship overlapped with the time when I was most productive in my career. It was a time when all my past experiences in various areas came together nicely, and Keisuke and Ronan were extremely helpful resources.

I was privileged to have the chance to learn from my amazing collaborators across many institutions: Akari Asai, Jonathan Bragg, Weizhu Chen, Eunsol Choi, Jonathan H. Clark, Lavinia Dunagan, Alexander R. Fabbri, Marjan Ghazvininejad, Jiatao Gu, Sairam Gurajada, Hannaneh Hajishirzi, Michael Hassid, Junjie Hu, Gabriel Ilharco, Kentaro Inui, Liwei Jiang, Daniel Khashabi, Lingpeng Kong, Chia-Hsuan Lee, Kenton Lee, Leo Z. Liu, Shayne Longpre, Nicholas Lourie, Ximing Lu, Ivan Montero Phoebe Mulcaire, Mari Ostendorf, Nikolaos Pappas, Hao Peng, Lucian Popa, Lianhui Qin, Daniel Rotem, Roy Schwartz, Weijia Shi, Gabriel Stanovsky, Yoichi Takahashi, Tianlu Wang, Yizhong Wang, Daniel S. Weld, Sean Welleck, Peter West, Dani Yogatama, Tao Yu, Youngjae Yu, Rowan Zellers, Luke Zettlemoyer, and Rui Zhang.

The work presented in this thesis received generous support from the Funai Overseas Scholarship, the Masason Foundation Fellowship, and an IBM Ph.D. Fellowship.

In Memory of Professor Dragomir Radev, My Undergraduate Advisor at Yale University On May 29th 2023, I was eagerly awaiting your email and feedback on my latest work as usual. To my great shock, I received the unexpected and sad news of your sudden passing. I went to your office at Yale to leave white lilies, which I believe symbolize the purity of your life-long commitment to mentorship, education, and research.

LILY (Language, Information, and Learning at Yale) is also the name of your NLP lab at Yale University. I am extremely fortunate to be part of the LILY lab since the beginning in Spring 2017. I still vividly remember the day I visited your office for the first time. At the time, I was in my senior year with almost no prior research experience. Despite this, you kindly offered to mentor me on my research project. After graduation, I sought your advice as to what I should do next. Soon after, you and Professor Robert Frank from Yale Linguistics very kindly secured funding and offered me a research assistant position. This experience became the foundation of my NLP research career. During my Ph.D., we continued to meet regularly and collaborate on many exciting projects.

Among many other things, your attitude towards research has always struck me as passion-

ate and open-minded. The field of NLP has experienced many changes since I started. We used to talk a lot about building core NLP models using LSTMs. Now, we are seeing tremendous progress from large language models. You always showed great enthusiasm for the latest advancements and how they are transforming the way we approach NLP problems. Your passion for this field was contagious. You consistently encouraged me to be open to new ideas, even when I was skeptical or anxious about new directions. As you led by example, no matter how the field changes, researchers have a responsibility to demonstrate limitations and potentials of new technologies for society.

As one of the researchers who were extremely lucky to have you as a mentor, I feel obliged to pay it forward by continuing to support younger generations of scholars. Thank you so much for the amazing seven years. May you find eternal rest and peace. ご冥福をお祈りいたします。

Jungo Kasai

笠井淳吾

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The growing computational (and thus financial and environmental) cost of large-scale AI/NLP models makes it difficult for many researchers to develop or even just use them (Schwartz et al., 2019). Only researchers working at a handful of extremely well-funded industry labs are able to perform necessary operations on the current state-of-the-art models to make progress. Efficient methods are needed to promote the accessibility of our technology and to ensure that people from diverse backgrounds can contribute to its progress from their various perspectives. I strongly believe that the inclusiveness of the AI community will be key to its success. In Part I, I introduce methods to improve the efficiency of widely-adopted AI models, using insights from empirical practice and classical machine learning.

In addition to efficiency, customization is important to support diverse applications of AI technologies. For example, while ChatGPT generates very fluent language, it has critical failure cases in the legal domain.[1] Can we customize these large-scale generation models for various applications without expensive model training? In Part II, I develop a language generation algorithm that customizes a large-scale, general-purpose model with guidance from a lightweight domain-expert model. This customization method thus avoids additional training of large-scale models, making it feasible in less-well-funded fields or institutions.

Lastly, I discuss evaluation methodologies, a key component for the success of diverse NLP applications. Evaluation lets us measure progress, understand model behaviors or failures, and

[1] https://www.forbes.com/sites/mattnovak/2023/05/27/lawyer-uses-chatgpt-in-federal-court-and-it-goes-horribly-wrong/?sh=77ed5cf13494.

guide research directions. Evaluating NLP systems presents a serious challenge, particularly for language generation tasks such as machine translation and summarization. Language generation is inherently open-ended, and generation quality cannot be measured using a simple metric like classification accuracy. In Part III, I establish a transparent human evaluation protocol for image captioning models and extend the effort further to develop a platform that facilitates progress in natural language generation tasks and their evaluation.

1.1 Thesis Outline

First, I focus on machine translation, one of the most well-studied areas in NLP that has been a driving force for new statistical machine learning approaches to language processing. Chapter 2 present a simple yet effective method for improving the efficiency of modern machine translation models: deep encoder, shallow decoder. I theoretically and empirically demonstrate that this strategy can improve the speed-quality tradeoff over standard transformer-based models and recent, fast translation methods: non-autoregressive machine translation that generates text using parallelism available in modern hardware like graphics processing units (GPUs) and tensor processing units (TPUs).

The transformer architecture (Vaswani et al., 2017) is a backbone of recent advances in NLP, computer vision, speech, computational biology, and beyond (e.g., Brown et al., 2020; Parmar et al., 2018; Jumper et al., 2021). Transformers outperform recurrent neural networks (RNNs) at the expense of their increased computational cost: their time and memory complexity scales quadratically with sequence length, in contrast to the linear complexity of RNNs. This computational requirement limits the usability of many strong transformers in the AI community. Chapter 3 introduces transformer-to-RNN (T2R), a method that converts any off-the-shelf transformer into an efficient RNN counterpart by performing a small amount of finetuning. Drawing inspiration from the classical kernel methods in machine learning, T2R learns to approximate transformers' quadratic computation during lightweight finetuning, resulting in a recurrent model with linear complexity. I empirically demonstrate that T2R generates long text with quality similar to the original transformer while achieving substantial speedup and memory savings (e.g., 10x speedup when producing 2048 consecutive words).

In Chapter 4, I develop an inference algorithm for language generation that combines strengths of diverse models (e.g., general-purpose and domain-expert models). Conventional ensembling methods make strong assumptions about vocabulary, tokenization, and probability factorization that often do not hold in practice. For example, these assumptions make it challenging to combine models with different specializations (e.g., medical/legal domains). I introduce the Twist algorithm, a simple yet effective method that relaxes these assumptions while avoiding any additional training, which could be expensive. Twist performs standard beam search alternately between two or more diverse generators with mutual guidance. Our extensive evaluations demonstrate that Twist decoding substantially outperforms each model used in isolation over various scenarios, encouraging researchers to seek out models with complementary strengths to the currently available models.

Chapter 5 proposes THumB, a rubric-based human evaluation protocol for image captioning models. The THumB rubric is carefully developed based on machine- and human-generated captions on a standard image captioning dataset. Each caption is evaluated along two main dimensions in a tradeoff (precision and recall) as well as other aspects that measure the text quality (fluency, conciseness, and inclusive language). The evaluations demonstrate several critical problems of the current evaluation practice. Human-generated captions show substantially higher quality than machine-generated ones, especially in coverage of salient information, while most automatic metrics say the opposite. Our rubric-based results reveal that a recent metric that uses image features from a large-scale model better correlates with human judgments than conventional text-only metrics because it is more sensitive to recall.

In Chapter 6, I argue that new advances on models and evaluation methods should each more directly benefit and inform the other. I formulate a generalization of leaderboards, bidimensional leaderboards (Billboards), that simultaneously tracks progress in language generation models and metrics for their evaluation. A Billboard facilitates two separate competitions, one for generators and another for evaluation metrics. Unlike conventional leaderboards that sort submitted systems by predetermined metrics, in this setup, both generators and evaluation metrics are accepted as competing entries in their respective categories. I show that a linear ensemble of a few diverse metrics sometimes substantially outperforms existing metrics in isolation. Billboards' built-in analysis shows that most automatic metrics overrate machine over human generation,

demonstrating the importance of updating metrics as generation models become strong.

Part I

Efficient Methods for Natural Language Processing

Chapter 2

Deep Encoder, Shallow Decoder: Reevaluating Non-autoregressive Machine Translation

This chapter focuses on efficient methods for machine translation, one of the most well-studied areas in natural language processing (NLP). Specifically, we present a simple yet effective approach to efficient machine translation: **deep encoder, shallow decoder**. Many machine translation models are now built upon a transformer-based (Vaswani et al., 2017) encoder-decoder model: the encoder transformer computes distributed representations for the source language (e.g., Japanese), and the decoder transformer generates text in the target language (e.g., English). Typically, researchers and practitioners assume that the encoder and the decoder have the same number of transformer layers. We empirically demonstrate that this assumption is suboptimal in the spectrum of the speed-quality tradeoff. We show that a simple strategy of a shallow decoder paired with a deep encoder can improve the tradeoff even over recent efficient methods using non-autoregressive generation.

The material in this chapter is adapted from Kasai et al. (2021a).

2.1 Introduction

Fast, accurate machine translation is a fundamental goal with a wide range of applications both in research and production. State-of-the-art neural machine translation systems generate translations *autoregressively* where words are predicted one-by-one conditioned on all previous words (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015; Wu et al., 2016; Vaswani et al., 2017). This sequential property limits parallelization, since multiple tokens in each sentence cannot be generated in parallel. A flurry of recent work developed ways to (partially) parallelize the decoder with *non-autoregressive* machine translation (NAR; Gu et al., 2018), thereby speeding up decoding during inference. NAR tends to suffer in translation quality because parallel decoding assumes conditional independence between the output tokens and prevents the model from properly capturing the highly multimodal distribution of target translations (Gu et al., 2018).

Recent work proposed methods to mitigate this multimodality issue, including iterative refinement (e.g., Lee et al., 2018; Ghazvininejad et al., 2019), and modeling with latent variables (e.g., Ma et al., 2019b; Shu et al., 2020). These approaches modify the decoder transformer to find a balance between decoding parallelism and translation quality. In this chapter, however, we adopt a different speed-quality tradeoff. Recent work by Kim et al. (2019) in autoregressive machine translation (AR) suggests that better speed-quality tradeoffs can be achieved by having different depths in the encoder and the decoder. Here, we make a formal argument in favor of *deep encoder, shallow decoder* configurations and empirically demonstrate better speed-quality tradeoffs for the AR baselines.

We provide extensive speed-quality comparisons between iterative NAR models and AR models with varying numbers of encoder and decoder layers. In particular, we use two types of speed measures for translation and discuss their relation to computational complexity. The two measures reflect two different application scenarios: feeding one sentence at a time, and feeding as many words as possible into the GPU memory. The first scenario is designed to simulate, for example, instantaneous machine translation that translates text (or even speech) input from users. This is where current NAR models shine—we can make full use of parallelism across decoding positions in a GPU. For this reason, much prior work in NAR only measures speed using

this metric (e.g., Gu et al., 2018, 2019b; Kasai et al., 2020; Li et al., 2020a). The second scenario aims at a situation where we want to translate a large amount of text as quickly as possible. In this case, we see that AR models run faster than NAR models by a large margin. Computation at each time step is large enough to exploit parallelism in a GPU, which cancels out the benefit from parallel NAR decoding. Further, AR models can cache all previous hidden states (Ott et al., 2019) and compute each step in linear time complexity with respect to the sequence length. In contrast, NAR models necessitate a fresh run of quadratic self and cross attention in every decoding iteration.

Interestingly, using a deep encoder and a shallow decoder in NAR models fails to retain the original translation accuracy by using 6 layers each (§2.4.1). This suggests that departure from AR decoding necessitates more capacity in the decoder; the strategy is effective specifically for AR models. In particular, our analysis demonstrates that an NAR decoder requires more layers to learn target word ordering (§2.5). In summary, our contributions are the following:

- We challenge three conventional assumptions in NAR evaluation: suboptimal layer allocation, lack of distillation for AR baselines, and insufficiently general speed measures.
- We provide a complexity analysis and identify an optimal layer allocation strategy that leads to better speed-quality tradeoffs, namely a *deep-shallow* configuration.
- We perform extensive analyses and head-to-head comparisons of AR and strong NAR models on seven standard translation directions. We demonstrate that the accuracy gap between the two model families is much wider than previously thought and that NAR models are unable to capture target word order well without sufficiently deep decoders.

2.2 Reevaluating Non-Autoregressive Machine Translation

We critically examine in this section the evaluation practices and assumptions that are widely held in the non-autoregressive neural machine translation (NAR) literature (e.g., Gu et al., 2018; Ghazvininejad et al., 2019; Kasai et al., 2020). In particular, we focus on three aspects: speed measurement (§2.2.1), layer allocation (§2.2.2), and knowledge distillation (§2.2.3).

2.2.1 Speed Measures

One major benefit of NAR models over AR ones is their ability to generate text in parallel. Current research on measuring speed has focused solely on the setting of translating one sentence at a time where full parallelization is trivial with a single GPU. However, we argue that this speed measure is not realistic in some scenarios because the GPU memory is finite and the GPU unit in such a setting is underused. To address this issue, we use two translation speed metrics to measure inference speed:

- $\mathbf{S_1}$ measures speed when translating one sentence at a time. This metric is used in standard practice and aligns with applications like instantaneous machine translation that translates text input from users immediately.
- $\mathbf{S_{max}}$ measures speed when translating in mini-batches as large as the hardware allows. This corresponds to scenarios where one wants to translate a large amount of text given in advance. For instance, such large-batch machine translation is implemented in the Google cloud service.[1]

For all models, both metrics measure wall-clock time from when the weights are loaded until the last sentence is translated. We report speedups relative to an AR baseline with a 6-layer encoder and a 6-layer decoder following prior work (Gu et al., 2018; Li et al., 2020a; Kasai et al., 2020).

2.2.2 Layer Allocation

Current evaluation practice in the NAR literature uses an equal number of layers for the encoder and decoder both in AR baselines and NAR models. However, previous studies in AR machine translation suggest that this allocation strategy leads to a suboptimal speed-quality tradeoff (Barone et al., 2017; Kim et al., 2019). These findings have several implications for evaluating NAR methods. We first discuss the strategy of *deep encoder, shallow decoder*, and provide a theoretical analysis of the speed-quality tradeoff in the context of NAR evaluation. Our analyses are verified empirically in the next section (§2.3).

[1] https://cloud.google.com/translate/docs/advanced/batch-translation.

Deep Encoder, Shallow Decoder

In line with prior work on deep encoders or shallow decoders (Barone et al., 2017; Wang et al., 2019a; Kim et al., 2019), we depart from the convention to allocate an equal number of layers on both sides and explore pairing a deep encoder with a shallow decoder for both AR and NAR methods. Here, we study the impact of such architectures and systematically compare AR and NAR methods.[2] As we will show in §2.3, an AR model with a *deep-shallow* configuration retains translation accuracy, but can substantially reduce decoding time. This is because at inference time, the encoder accounts for a smaller part of the overhead since its computation can be easily parallelized over source positions; on the other hand, the speedup gains from a lightweight decoder are substantial.

	By Layer			**Full Model**		
	Enc.	AR Dec.	NAR Dec.	AR *E*-*D*	AR *E*-1	NAR *E*-*D*
Total Operations	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(TN^2)$	$\mathcal{O}(EN^2 + DN^2)$	$\mathcal{O}(EN^2 + 1 \cdot N^2)$	$\mathcal{O}(EN^2 + DTN^2)$
Time Complex.	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$	$\mathcal{O}(TN)$	$\mathcal{O}(EN + DN^2)$	$\mathcal{O}(EN + N^2)$	$\mathcal{O}(EN + DTN)$

Table 2.1: Analysis of transformers. Time complex. indicates time complexity when full parallelization is assumed. N: source/target length; E: encoder depth; D: decoder depth; T: # NAR iterations.

Complexity Analysis

This section analyzes the complexities of transformer-based encoders, autoregressive and non-autoregressive decoders. We focus on two key properties: (1) the total amount of operations and (2) time complexity when full parallelization is assumed (Harris, 2007).

Notation For simplicity let us assume the source and target text have the same length N. T is the number of iterations in an iterative NAR method (typically $T < N$). Let E and D denote the numbers of encoder and decoder layers.

Table 2.1 breaks down the comparison. AR and NAR models use the same encoder architecture. There are several interesting distinctions between AR and NAR decoders. First, although their total amounts of operations are both quadratic in sequence length, an NAR decoder with T decoding iterations needs T times more computation. Second, an AR decoder has time com-

[2]Note that Kim et al. (2019) proposed other methods to optimize CPU decoding of AR models, but we do not apply them, to ensure fair comparisons between AR and NAR models.

plexity quadratic in sequence length. This contrasts with the linear time complexity of an NAR decoder, which is the powerhouse of its S_1 speedup (§2.4.1). This is because the attention computation can be readily parallelized across target positions in NAR decoders.

By such comparisons we make the following key observations:

(a) For both AR and NAR models, the time complexity is dominated by decoders. When $T < N$, an NAR model has an advantage over its AR counterpart with the same layers.

(b) Innocuous as it may seem, the constant T contributes major computational cost in terms of the total operations of NAR models. Empirically, T needs to be at least 4 to perform competitively to AR models (Ghazvininejad et al., 2019; Kasai et al., 2020).

(a) suggests that one can significantly speed up S_1 decoding by using shallower decoders, while increasing the encoder depth only results in a mild slowdown. As we will show later in the experiments, AR decoders are much more robust to using fewer layers than NAR decoders. For example, AR *E*-1 can decode much faster than AR *E*-*D* and comparably to NAR *E*-*D*, while retaining the accuracy of AR *E*-*D*. From (b), one may expect a different trend in S_{max} from S_1: in large mini-batch decoding, an AR model can make use of the GPU's compute units, since now parallelization happens over the instances in a mini-batch. In other words, under the S_{max} evaluation where the GPU is running close to its maximum flop/s, NAR can actually be slower since it needs more operations due to its iterative decoding. This is confirmed by our experiments (§2.4.1).

2.2.3 Knowledge Distillation

Most NAR models rely on sequence-level knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016) to achieve a reasonable speed-quality tradeoff, where NAR models are trained on output translations from a (larger) AR model. Nevertheless, standard practice in this area assumes that knowledge distillation is not required for the AR baseline. Here, we aim for fair evaluation by applying distillation to both model families; we depart from previous practice where NAR models trained *with* distillation are compared with AR models trained *without* (Ran et al., 2019; Sun et al., 2019; Shu et al., 2020; Zhou et al., 2020a; Saharia et al., 2020) with a few exceptions (Ghazvininejad et al., 2019; Kasai et al., 2020). Our analysis (§2.5) demonstrates that

AR models also benefit from knowledge distillation and that the accuracy gap between AR and NAR models is wider than previously established.

2.3 Experiments

We compare NAR and AR models with different layer allocation strategies on standard machine translation datasets of varying languages and sizes. Our results show that deep-shallow AR models provide a better speed-quality tradeoff than NAR models.

2.3.1 Baselines and Comparison

Prior work has proposed various approaches to non-autoregressive machine translation (NAR). These methods must seek a balance between speed and quality: the more decoding parallelization is introduced into a model, the more the output quality deteriorates due to a conditional independence assumption. Some of the existing NAR models rescore the candidates with external autoregressive models (Sun et al., 2019; Li et al., 2020a), or apply reordering modules (Ran et al., 2019). We mainly compare with two iterative NAR models (Ghazvininejad et al., 2019; Kasai et al., 2020) because of their strong performance *without* relying on any external system:

- **CMLM** (Ghazvininejad et al., 2019) predicts randomly masked target tokens given observed ones as well as the source. At inference time, it first predicts all target words non-autoregressively, and then iteratively masks and predicts the words that the model is least confident about. Following previous practice (Ghazvininejad et al., 2019, 2020b), we decode 5 candidate lengths in parallel (*length beam*) with $T = 4$ or $T = 10$ iterations.
- **DisCo** (Kasai et al., 2020) predicts every target token given an arbitrary subset of the rest of the target tokens. Following Kasai et al. (2020), we use their parallel easy-first inference, and set the maximum number of iterations to 10 and the length beam size to 5.

Knowledge Distillation We apply sequence-level knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016) when training both NAR and AR models (§2.2.3). For the teacher models, we use left-to-right AR transformer models: transformer-large for EN-DE, EN-ZH, and EN-FR, and transformer-base for EN-RO (Ghazvininejad et al., 2019; Kasai et al., 2020).

2.3.2 Experimental Setup

We experiment with 7 translation directions from four datasets of various training data sizes: WMT14 EN-DE (4.5M pairs, Bojar et al., 2014), WMT16 EN-RO (610K, Bojar et al., 2016), WMT17 EN-ZH (20M, Bojar et al., 2017), and WMT14 EN-FR (36M, EN→FR only). These datasets are all encoded into BPE subwords (Sennrich et al., 2016b). We follow the preprocessing and data splits of previous work (EN-DE: Vaswani et al., 2017; EN-RO: Lee et al., 2018; EN-ZH: Hassan et al., 2018; Wu et al., 2019; EN-FR: Gehring et al., 2017). Following previous practice, we use SacreBLEU (Post, 2018) to evaluate EN→ZH performance, and BLEU (Papineni et al., 2002) for others.[3] For all autoregressive models, we apply beam search with size 5 and length penalty 1.0. All models are implemented using `fairseq` (Ott et al., 2019). S_1 and S_{max} wall-clock time speedups (§2.2) are evaluated on the same single Nvidia V100 GPU with 16GB memory. We apply half-precision training and inference (Micikevicius et al., 2018; Ott et al., 2019). It speeds up NAR models' S_{max} by 30+%, but not S_1, in line with previous observations (Kim et al., 2019).

Hyperparameters We follow the hyperparameters of the base sized transformer (Vaswani et al., 2017): 8 attention heads, 512 model dimensions, and 2,048 hidden dimensions for both the encoder and decoder. For each model and dataset, the dropout rate is tuned from $[0.1, 0.2, 0.3]$ based on development BLEU performance. The EN→FR models are trained for 500K updates, while others for 300K (Kasai et al., 2020). Development BLEU is measured after each epoch, and we average the 5 best checkpoints to obtain the final model (Vaswani et al., 2017). See Appendix §A.1.1 for further details.

2.4 Results and Discussion

We provide in-depth results comparing performance and speedup across AR and NAR models.

2.4.1 Deep Encoder, Shallow Decoder

Fig. 2.1 shows translation speed-quality tradeoff curves of CMLM, DisCo, and AR models on WMT14 EN-DE and WMT16 EN-RO test data. For each model we plot the results of configurations with varying encoder and decoder depths. For brevity, we denote by *E-D* a model with

[3] SacreBLEU hash: BLEU+case.mixed+lang.en-zh+numrefs.1+smooth.exp+test.wmt17+tok.zh+version.1.3.7.

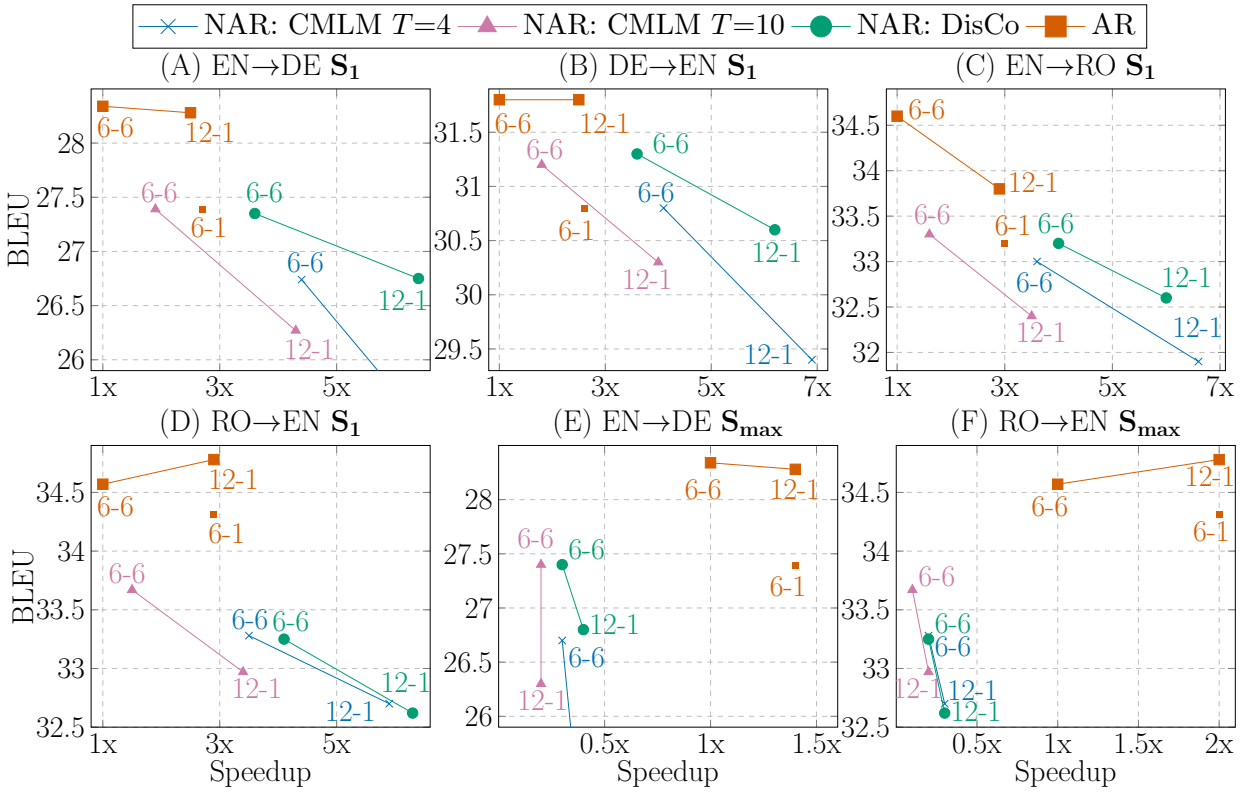


Figure 2.1: BLEU and speed comparisons with varying numbers of encoder and decoder layers on the test data. 12-1 denotes 12 encoder layers and 1 decoder layer. AR deep-shallow (12-1) finds a balanced middle ground in the tradeoff. Knowledge distillation is applied to all models (§2.3.1). See Table A.1 in Appendix for more results.

an E-layer encoder and a D-layer decoder. All speedups are measured relative to the AR 6-6 baseline (§2.2).

Firstly, under the 6-6 configuration, the AR model outperforms both CMLM and DisCo by a considerable margin in BLEU, but it achieves the slowest S_1 (see Fig. 2.1A–D). Using a single-layer decoder, AR 6-1 gains a substantial S_1 speedup (2.6x for EN→DE and 2.9x for RO→EN), but this comes at a cost of BLEU: 28.3 vs. 27.4 for EN→DE, and 34.6 vs. 34.3 for RO→EN. AR 12-1 lands on a balanced middle ground: it yields similar BLEU to AR 6-6, but its S_1 is more than 2.5 times faster. Notably, AR 12-1 achieves even faster S_1 than that of the CMLM 6-6 model with 10 iterations. In contrast, NAR 12-1 models generally suffer in BLEU compared to the 6-6 configuration; e.g., 26.8 (DisCo 12-1) vs. 27.4 (DisCo 6-6) in EN→DE.

Interestingly, all NAR models achieve slower S_{max} than the AR 6-6 baseline (DisCo 6-6: 0.3x; CMLM 6-6 T=10: 0.1x in RO→EN). This is consistent with our complexity analysis in §2.2.2, where we found that with the same layer allocation, iterative NAR models need more total

computation than the AR counterpart. AR 12-1 still gains a considerable speedup over AR 6-6 (2.0x in RO→EN). These results suggest that current NAR models have little advantage when translating a large amount of text given in advance, and one should clarify this distinction when discussing translation speed. See Table A.1 in the appendix for full results from all four directions.

Model			**WMT17 EN→ZH**			**WMT17 ZH→EN**			**WMT14 EN→FR**		
	T	***E-D***	**BLEU**	**S_1**	**S_{max}**	**BLEU**	**S_1**	**S_{max}**	**BLEU**	**S_1**	**S_{max}**
CMLM	4	6-6	33.6	**3.5×**	0.2×	22.6	**3.8×**	0.2×	40.2	**3.8×**	0.2×
CMLM	10	6-6	34.2	1.5×	0.1×	23.8	1.7×	0.1×	40.6	1.7×	0.1×
DisCo		6-6	34.6	2.5×	0.2×	23.8	2.6×	0.2×	40.6	3.6×	0.2×
AR Deep-Shallow		12-1	34.7	2.7×	**1.7×**	**24.2**	2.9×	**1.8×**	**42.0**	2.8×	**1.9×**
AR		6-6	**35.1**	1.0×	1.0×	**24.2**	1.0×	1.0×	**42.0**	1.0×	1.0×
Dist. Teacher		6-6	35.0	–	–	24.7	–	–	42.0	–	–

Table 2.2: Test BLEU and speed comparisons with varying numbers of encoder (E) and decoder (D) layers on large bitext. Best performance (excluding the distillation teachers) is bolded.

Table 2.2 presents results from large bitext experiments, EN↔ZH and EN→FR. We observe similar trends: AR deep-shallow achieves similar BLEU to AR 6-6 while boosting both S_1 and S_{max} speed substantially. For EN↔ZH, AR deep-shallow has a more S_1 speedup than DisCo (2.7x vs. 2.5x in EN→ZH, 2.9 vs. 2.6 in ZH→EN). Particularly noteworthy is its performance in EN→FR: 42.0 BLEU, a 1.4 point improvement over the best NAR model. These results illustrate that the strategy of having a deep encoder and shallow decoder remains effective in large bitext settings, when the model has to learn potentially more complex distributions from more samples.

Lastly, Table 2.3 compares AR deep-shallow to recent iteration-based NAR results. All NAR models use the 6-6 configuration with the base size except that Imputer (Saharia et al., 2020) uses 12 self-attention layers over the concatenated source and target. Overall, our AR deep-shallow models outperform most NAR models, with the only exception being EN→RO where it underperforms Imputer by 0.6 BLEU points. However, each iteration takes strictly more time in the Imputer model than in CMLM or DisCo, since it requires a fresh run of 12-layer self attention over a concatenation of input and output sequences. As we saw in Fig. 2.1, AR deep-shallow yields comparable S_1 to CMLM 6-6 with 4 iterations, which would be about twice as fast as Imputer with 8 iterations.

	WMT14 EN−DE				WMT16 EN−RO				WMT17 EN−ZH			
Model	**→DE**	***T***	**→EN**	***T***	**→RO**	***T***	**→EN**	***T***	**→ZH**	***T***	**→EN**	***T***
CMLM	25.9	4	29.9	4	32.5	4	33.2	4	32.6	4	21.9	4
	27.0	10	31.0	10	33.1	10	33.3	10	33.2	10	23.2	10
LevT	27.3	>7	–	–	–	–	33.3	>7	–	–	–	–
DisCo	27.3	4.8	31.3	4.2	33.2	3.3	33.2	3.1	34.6	5.4	23.8	5.9
SMART	27.0	4	30.9	4	–	–	–	–	33.4	4	22.6	4
	27.6	10	31.3	10	–	–	–	–	34.1	10	23.8	10
Imputer	28.0	4	31.0	4	34.3	4	34.1	4	–	–	–	–
	28.2	8	31.3	8	34.4	8	34.1	8	–	–	–	–
AR 6-6	**28.3**	*N*	**31.8**	*N*	**34.6**	*N*	34.6	*N*	**35.1**	*N*	**24.2**	*N*
AR 12-1	**28.3**	*N*	**31.8**	*N*	33.8	*N*	**34.8**	*N*	34.7	*N*	**24.2**	*N*
Teacher	28.6	*N*	31.7	*N*	34.6	*N*	34.6	*N*	35.0	*N*	24.7	*N*

Table 2.3: Test BLEU comparisons with iterative NAR methods. T indicates the average # iterations. CMLM: Ghazvininejad et al. (2019); LevT: Gu et al. (2019b); DisCo: Kasai et al. (2020); SMART: Ghazvininejad et al. (2020b); Imputer: Saharia et al. (2020). Best performance (excluding the distillation teachers) is bolded.

2.4.2 Constrained Views

In this section, we present two controlled experiments to compare NAR and AR models thoroughly.

S_1 Speed Constraint From §2.4, we see that compared to NAR models, AR deep-shallow yields a better translation speed-quality balance—despite being slightly slower in S_1 on some of the datasets, it achieves better BLEU across the board. To confirm this result, we further compare an AR deep-shallow model against two NAR models, controlling for S_1 speed. More specifically, we experiment with NAR models of varying encoder depths, and pair each with as many decoder layers as possible until it reaches AR 12-1's S_1 speed. Fig. 2.2 (left) shows the results. For CMLM T=4, CMLM T=10, and DisCo, the best configurations of 12-layer encoders were paired up with 12, 4, and 9 decoder layers, respectively. All NAR models improve performance as the encoder becomes deeper and surpass the scores of the 6-6 baselines (shown as squares along $x = 6$). Nonetheless, there is still a large BLEU gap from AR 12-1. This illustrates that the two NAR models are not able to match AR deep-shallow's accuracy under the same S_1 speed budget.

Layer Constraint We can speed up autoregressive translation (AR) by developing a model with a deep encoder and a one-layer decoder. Here we thoroughly compare layer allocation

strategies. Shown in Fig. 2.2 (middle) are results of NAR and AR methods under the constraint of 12 transformer layers in total. NAR models perform well when the decoder and encoder are balanced with slight tendency to deep encoders. On the other hand, the AR models perform consistently well with 4 or more encoder layers. This confirms that using deep encoders and shallow decoders is more effective in AR models than in NAR ones. Note that the number of parameters in each layer allocation differs since a decoder layer contains 30% more parameters than an encoder layer, due to cross attention.

2.5 Further Analysis

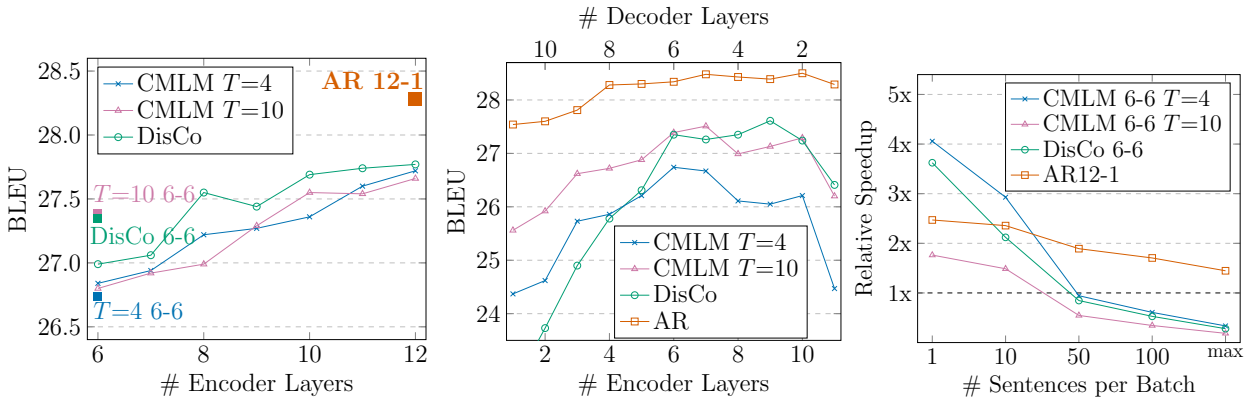


Figure 2.2: WMT14 EN→DE test results under various conditions. **Left**: varying depths of the encoder under the S_1 speed constraint of AR 12-1 ■. **Middle**: varying allocation of a total of 12 transformer layers over the encoder and decoder. **Right**: varying inference batch sizes.

Speedup and Batch Size When decoding with large batches, NAR models can be slower than their AR counterpart (§2.4). Here we further study this effect. Fig. 2.2 (right) plots the relative speedups of different models' decoding with varying numbers of sentences per batch up to the hardware limit ("max," §2.2.1). The speedup by NAR models diminishes as the batch size grows: they have similar decoding speed to AR 6-6 with batch size 50, and become slower with larger batch sizes. In contrast, the speedup from AR deep-shallow decreases much more gradually.

Decoder Depth and Reordering Words From earlier results we see that NAR models need deeper decoders than AR models to perform well. We hypothesize that one reason is that NAR decoders need to learn to adjust to diverging word order between the source and the target: an AR decoder takes as input all preceding tokens and explicitly learns a conditional distribution, while an NAR decoder needs to learn target word ordering from scratch.

Model	*E-D*	Orig.	Reorder	Δ
CMLM, $T = 10$	6-6	27.4	31.7	4.3
CMLM, $T = 10$	12-1	26.3	31.0	4.7
DisCo	6-6	27.4	31.0	3.6
DisCo	12-1	26.8	31.6	**4.8**
AR	6-6	**28.3**	**32.6**	4.3
AR Deep-Shallow	12-1	**28.3**	**32.6**	4.3

Model	*E-D*	Raw	Dist.	Δ
CMLM, $T = 4$	6-6	22.3	25.9	**3.6**
CMLM, $T = 10$	6-6	24.6	27.0	2.4
Imputer, $T = 4$	12	24.7	27.9	3.2
Imputer, $T = 8$	12	25.0	27.9	2.9
DisCo	6-6	24.8	27.4	2.6
AR Deep-Shallow	12-1	26.9	**28.3**	1.4
AR	6-6	**27.4**	**28.3**	0.9

Table 2.4: Left: WMT14 EN→DE test results in BLEU using reordered English input. **Right**: WMT14 EN→DE test results in BLEU that analyze the effects of distillation in fast translation methods. All distillation data are obtained from a transformer large. *E*: encoder depth; *D*: decoder depth; *T*: # iterations. Imputer (Saharia et al., 2020) uses 12 self-attention layers over the concatenated source and target, instead of the encoder-decoder architecture.

To test this hypothesis, we conduct the following controlled experiment in EN→DE translation. We choose German because of its divergence in word order from English. We first run the `fast_align` tool (Dyer et al., 2013)[4] on all bitext data (including the test set), and disable the NULL word feature to ensure that every English word is aligned to exactly one German word. We then shuffle the English words according to the order of their aligned German words. When multiple English words are aligned to the same German word, we keep the original English order. Finally, we apply the same BPE operations as the original data, and train and evaluate various models on the new reordered data. Table 2.4 (left) shows the results. AR gains the same improvement regardless of the layer configuration; in contrast, NAR 12-1 benefits more than NAR 6-6. This result supports our hypothesis that word reordering is one reason why NAR models need a deeper decoder.

Effects of Distillation We applied sequence-level knowledge distillation (Kim and Rush, 2016) to all models. Here we analyze its effects over the WMT14 EN→DE test data (Table 2.4 right). An AR transformer large model is used as the teacher model. All models benefit from distillation as indicated by positive Δ, including the AR models.[5] Many recent works only compare NAR models trained with distillation to AR models trained *without*. Our finding shows that that AR models with distillation can be an additional baseline for future NAR research. AR

[4] https://github.com/clab/fast_align.

[5] While the same distillation data are used in Ghazvininejad et al. (2019), they report a smaller improvement from distillation in BLEU. There are several potential reasons: we tuned the dropout rate for each model on the validation data and averaged the checkpoints that achieved the top 5 BLEU scores (§2.3.2). We note that our results are in line with the previous observations (Kim et al., 2019; Zhou et al., 2020a; Kasai et al., 2020) where a similar improvement is gained by distilling an AR transformer large model to a base one.

deep-shallow deteriorates much less on the raw data compared to the iterative NAR methods, suggesting that the strategy of speeding up AR models is better suited to modeling raw, complex data than the NAR methods.

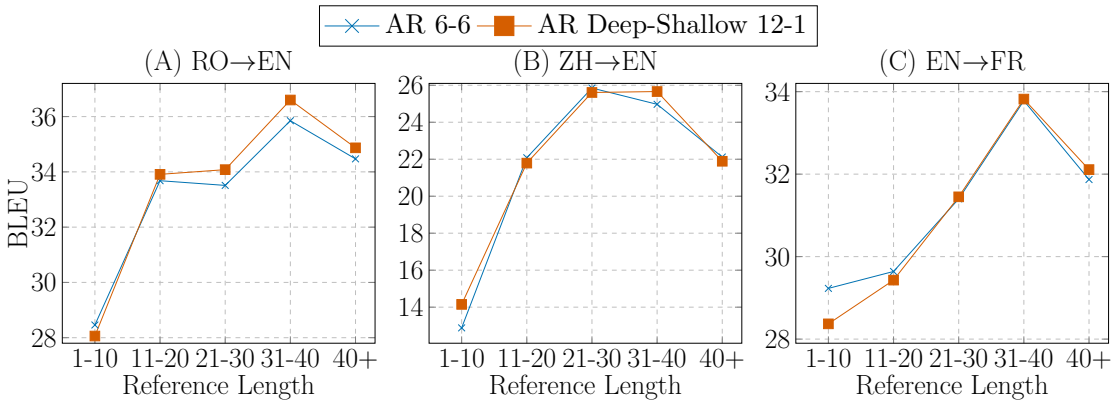


Figure 2.3: Test BLEU and target length comparisons for the AR 6-6 and deep-shallow 12-1 models.

Breakdown by Sentence Length Fig. 2.3 illustrates the relation between BLEU scores and reference translation lengths. We observe almost identical patterns between AR 6-6 and deep-shallow models, suggesting that they perform similarly regardless of the translation length.

Can we reduce the decoder further? We saw that an autoregressive model with a single-layer decoder and a sufficiently deep encoder can retain the accuracy of the baseline with 6 layers each. One may ask whether we can make the decoder even more compact. Our preliminary experiments showed that we can remove the feed-forward module from the decoder without hurting performance. This increases the S_1 speed by 10%. We leave further exploration to future work.

Length Candidates and S_{max} for NAR Following the original works (Ghazvininejad et al., 2019; Kasai et al., 2020), we fixed the number of length candidates (i.e., the length beam size) to 5 for all NAR models, but a smaller beam size can speed up S_{max} by allowing more sentences to be fed in a batch. Indeed, we found that NAR models can improve their S_{max} by reducing the beam size at the expense of some accuracy drop. For example, we observed a loss of 0.5 BLEU points in EN→DE when decreasing the length beam size from 5 to 1. Nonetheless, NAR 6-6 models with beam size 1 still resulted in 0.6–0.9x S_{max} compared to the AR 6-6 baseline.

2.6 Related Work

In addition to the work we already discussed, we highlight related work on efficient methods for generation.

2.6.1 Non-autoregressive Machine Translation

In addition to the work already discussed, several other works proposed to iteratively refine (or insert) output predictions (Mansimov et al., 2019; Stern et al., 2019; Gu et al., 2019a; Chan et al., 2019a,b; Li et al., 2020a; Guo et al., 2020). Other approaches include adding a light autoregressive module to parallel decoding (Kaiser et al., 2018; Sun et al., 2019; Ran et al., 2019), partially decoding autoregressively (Stern et al., 2018, 2019), rescoring output candidates autoregressively (e.g., Gu et al., 2018), mimicking hidden states of an autoregressive teacher (Li et al., 2019), training with different objectives than vanilla cross-entropy (Libovický and Helcl, 2018; Wang et al., 2019b; Shao et al., 2020; Tu et al., 2020; Saharia et al., 2020; Ghazvininejad et al., 2020a), reordering input sentences (Ran et al., 2019), training on additional data from an autoregressive model (Zhou and Keung, 2020), and modeling with latent variables (Ma et al., 2019b; Shu et al., 2020). The approach of adding a light autoregressive module is closest to our method, but note that we pack all *non-autoregressive* computation into the encoder.

2.6.2 Optimizing Autoregressive Transformer

Prior work has suggested various ways to optimize autoregressive transformers for fast inference. For example, Kim et al. (2019) considered shallow decoders and layer tying (Dabre and Fujita, 2019; Dehghani et al., 2019) on the transformer decoder and found that it sped up inference on CPUs, but not on a GPU, which was our focus. Kim et al. (2019) also explored concurrent streams where multiple batches are fed at once to make better use of a GPU. Shi and Knight (2017) proposed a vocabulary reduction method to speed up the last softmax computation. Senellart et al. (2018) also adopted vocabulary reduction and explored “fat decoder, thin encoder" on RNN-based models. Zhang et al. (2018a) used dynamic programming in an average attention network to accelerate inference. Wu et al. (2019) developed a model with dynamic convolutions and compared its speed and accuracy with non-autoregressive models. Other works proposed

methods to reduce attention computation in autoregressive transformers (Kitaev et al., 2020; Katharopoulos et al., 2020; Chelba et al., 2020; Peng et al., 2021). Some of these methods can be used orthogonally to further facilitate fast inference in a transformer, but our goal is to fairly reexamine the speed-quality tradeoff between autoregressive and non-autoregressive approaches under the same conditions.

2.7 Summary

This chapter presented theoretical and empirical studies to demonstrate that autoregressive neural machine translation can be dramatically sped up by a simple layer allocation strategy: deep encoder, shallow decoder. Compared to strong non-autoregressive models, deep-shallow autoregressive models achieve substantial improvement in translation quality with comparable inference speed. Our results suggest that layer allocation, knowledge distillation, and speed measurement are important aspects that future work on non-autoregressive machine translation should take into consideration. More generally, a model with a deep encoder and a shallow decoder can be used for any sequence-to-sequence task, including large-scale pretraining (Lewis et al., 2020; Liu et al., 2020).

Chapter 3

Finetuning Pretrained Transformers into RNNS

Most state-of-the-art systems in natural language processing (NLP) are now built upon *pretrained* transformers, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and GPT-2/3 (Radford et al., 2019; Brown et al., 2020). These pretrained models benefit from their large-scale model and data sizes and are computationally (and thus financially and environmentally) expensive to train and even use. One of the fundamental challenges of using these models is that transformers' time and memory complexity scales quadratically with sequence length. This contrasts with the previous state-of-the-art architecture, recurrent neural networks (RNNs), that scales linearly with sequence length. This chapter introduces T2R (Transformer-to-RNN), a method that converts any pretrained transformer to an efficient recurrent neural network. This conversion can be performed without losing the downstream performance or repeating the expensive pretraining process.

3.1 Introduction

Transformer models (Vaswani et al., 2017) have advanced the state of the art beyond recurrent neural network models (e.g., LSTMs, Hochreiter and Schmidhuber, 1997; GRUs, Cho et al., 2014) across a wide range of natural language processing tasks. In particular, the transformer architec-

The material in this chapter is adapted from Kasai et al. (2021b).

ture has been widely used in autoregressive modeling such as language modeling (Baevski and Auli, 2019) and machine translation (Vaswani et al., 2017). The transformer makes crucial use of interactions between feature vectors over the input sequence through the attention mechanism (Bahdanau et al., 2015). However, this comes with significant computation and memory footprint during generation (Kasai et al., 2021a). Since the output is incrementally predicted conditioned on the prefix, generation steps cannot be parallelized over time steps and require quadratic time complexity in sequence length. The memory consumption in every generation step also grows linearly as the sequence becomes longer. This bottleneck for long sequence generation limits the use of large-scale pretrained transformers, such as GPT-3 (Brown et al., 2020), Image Transformer (Parmar et al., 2018), and DALL-E (Ramesh et al., 2021).

Recent work aims at reducing the overhead of autoregressive transformers (Child et al., 2019; Kitaev et al., 2020; Beltagy et al., 2020, *inter alia*). Among them are recurrent alternatives that approximate the standard softmax attention (Katharopoulos et al., 2020; Peng et al., 2021; Choromanski et al., 2021; Schlag et al., 2021). Similar to recurrent neural networks (RNNs), those models represent the context by a recurrent state with a fixed size, thereby achieving linear time and constant memory complexity in generation sequence length. When the recurrent state size is smaller than the sequence length, these variants provide substantial speed and memory advantages over the transformer. A small state size, however, tends to deteriorate the generation quality (Peng et al., 2021), leading to a tradeoff between efficiency and accuracy.

We improve the balance between efficiency and accuracy by a *conversion* approach: instead of training a recurrent alternative from scratch, we develop a method to *convert* a pretrained transformer into an efficient RNN that speeds up generation and reduces memory footprints. Our conversion proceeds with a *swap-then-finetune* process. Specifically, we change the exponential similarity function in the attention mechanism to the dot product after a single-layer MLP feature mapping. We then finetune the MLP parameters and the other network parameters. Our experiments in language modeling and machine translation show that the conversion can compress the context into a much smaller recurrent state than the sequence length (e.g., 1/16 of the sequence length in WikiText-103 language modeling) while retaining high accuracy. In addition, this conversion requires much less GPU time than training randomly initialized models from scratch.

State-of-the-art models in many natural language tasks are increasingly dependent on large-scale pretrained transformer models (e.g., GPT-2, Radford et al., 2019; BERT, Devlin et al., 2019; RoBERTa, Liu et al., 2019; T5, Raffel et al., 2020; BART, Lewis et al., 2020; DeBERTa, He et al., 2021). Converting a large off-the-shelf transformer to a lightweight inference model without repeating the whole training procedure is particularly useful in many downstream applications. This chapter focuses on text generation and presents a viable approach towards efficient inference with high accuracy.

3.2 Convert a Transformer into an RNN

The transformer architecture consists of multihead attention, feedforward, and layer normalization modules (Vaswani et al., 2017). When a transformer is *trained* for a sequence generation task with teacher forcing (Williams and Zipser, 1989), the attention can be parallelized over positions because the target sequence is fully available. During *generation*, on the other hand, the output is incrementally constructed. As a result, the attention becomes an inference bottleneck for long sequences. This chapter presents a method to eliminate this bottleneck by converting a pretrained transformer into an efficient RNN of linear time and constant space complexity. We provide a detailed complexity analysis in terms of the sequence length and model dimensions.

3.2.1 Multihead Attention

The attention module takes as input sequences of *source* and *target* vectors. The source vectors are used to produce *key* and *value* features, while the target vectors are mapped to *query* vectors. More formally, denote by $\{\mathbf{x}_i^{\text{tgt}}\}_{i=1}^N$ and $\{\mathbf{x}_j^{\text{src}}\}_{j=1}^M$ the target and source vectors, where $\mathbf{x}_i^{\text{tgt}}, \mathbf{x}_j^{\text{src}} \in \mathbb{R}^h$ and h is the model dimensionality. We assume r attention heads of d dimensions ($h = dr$). For each head, the input vectors are first mapped to d dimensional *query*, *key*, and *value* features by learned affine transformations with $\mathbf{W}_* \in \mathbb{R}^{d \times h}$ and $\mathbf{b}_* \in \mathbb{R}^d$:

$$\mathbf{q}_i = \mathbf{W}_q \mathbf{x}_i^{\text{tgt}} + \mathbf{b}_q, \tag{3.1a}$$

$$\mathbf{k}_j = \mathbf{W}_k \mathbf{x}_j^{\text{src}} + \mathbf{b}_k, \quad \mathbf{v}_j = \mathbf{W}_v \mathbf{x}_j^{\text{src}} + \mathbf{b}_v. \tag{3.1b}$$

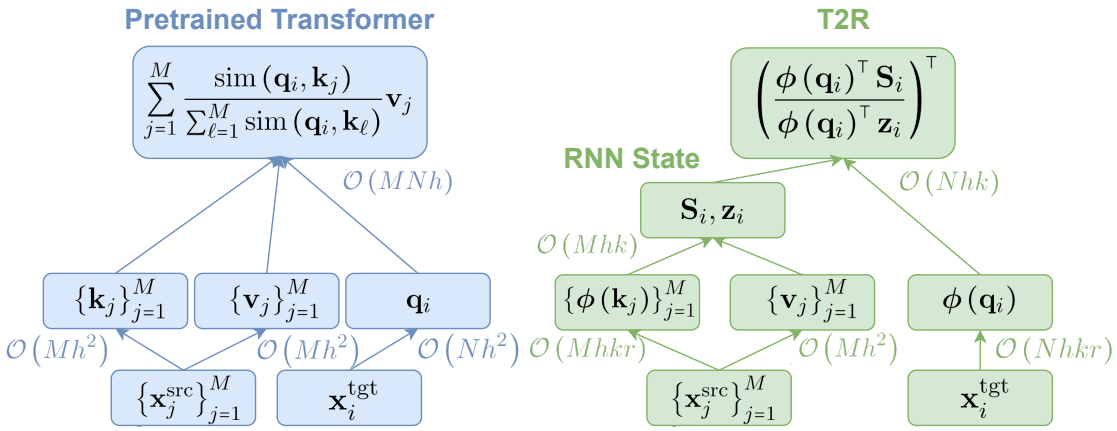


Figure 3.1: Attention computation steps and their time complexity in pretrained transformer and T2R models during inference generation. Features $\phi(\mathbf{q}_i)$ and $\phi(\mathbf{k}_j)$ are directly computed from input vectors, and $\mathbf{q}_i$ and $\mathbf{k}_j$ are never constructed. M: source length; N: target length; h: model dimensions; k: feature size; r: # heads.

The similarities of each query vector $\mathbf{q}_i$ with all M key vectors are computed and normalized to produce attention coefficients, which are then used to output a weighted average of the value vectors (Vaswani et al., 2017):

$$\mathbf{x}_i^{\mathrm{out}} = \sum_{j=1}^{M} \frac{\mathrm{sim}(\mathbf{q}_i, \mathbf{k}_j)}{\sum_{\ell=1}^{M} \mathrm{sim}(\mathbf{q}_i, \mathbf{k}_\ell)} \mathbf{v}_j, \tag{3.2a}$$

$$\mathrm{sim}(\mathbf{x}, \mathbf{y}) = \exp\left(\mathbf{x} \cdot \mathbf{y} / \sqrt{d}\right). \tag{3.2b}$$

Multihead attention runs this procedure for each of the r heads in parallel and concatenates r output vectors to get the final h dimensional vector.[1]

Generation Speed Overhead Fig. 3.1 depicts the transformer computation steps from input vectors and their time complexity. We assume that the time complexity of multiplying an $n \times m$ matrix by an $m \times k$ is $\mathcal{O}(nmk)$ as implemented in cuBLAS (NVIDIA, 2014).[2] It consists of the following two stages.

- **Feature Mapping**: computation of $\{\mathbf{q}_i\}_{i=1}^{N}$, $\{\mathbf{k}_j\}_{j=1}^{M}$, and $\{\mathbf{v}_j\}_{j=1}^{M}$ for all r heads from input vectors (Eqs. 3.1a-3.1b). Time complexity of $\mathcal{O}(Nh^2)$, $\mathcal{O}(Mh^2)$, and $\mathcal{O}(Mh^2)$.
- **Attention**: weighted average over the value vectors (Eq. 3.2a). $\mathcal{O}(MNh)$, quadratic in

[1] Layer normalization (Ba et al., 2016), residual connection (He et al., 2016), and projection are suppressed for brevity.

[2] If the batch size is small enough, parallelization can speed up matrix multiplication.

sequence length (M, N).

Generation Memory Overhead In autoregressive generation, query, key, and value vectors consume space complexity of $\mathcal{O}(h)$, $\mathcal{O}(Mh)$, and $\mathcal{O}(Mh)$ in every generation step. Every step's attention weight (Eq. 3.2a) spans over M source positions, taking $\mathcal{O}(Mr)$ space, linear in sequence length M.

3.2.2 Converting Transformers to RNNs

To address this generation bottleneck of quadratic time and linear space, this chapter proposes Transformer-to-RNN (T2R), a method to convert a pretrained transformer to an RNN inference model of linear time and constant memory complexity in sequence length (Fig. 3.1). T2R follows a swap-then-finetune procedure that modifies the attention computation of a pretrained transformer, and finetunes the model with the task objective.

We first replace the dot-then-exponential similarity function in a pretrained transformer (Eq. 3.2b) by

$$\widetilde{\mathrm{sim}}\left(\mathbf{x}, \mathbf{y}\right) = \boldsymbol{\phi}\left(\mathbf{x}\right) \cdot \boldsymbol{\phi}\left(\mathbf{y}\right), \tag{3.3a}$$

$$\boldsymbol{\phi}\left(\mathbf{x}\right) = \mathrm{ReLU}\left(\mathbf{W}_{\boldsymbol{\phi}}\mathbf{x} + \mathbf{b}_{\boldsymbol{\phi}}\right). \tag{3.3b}$$

Here $\mathbf{W}_{\boldsymbol{\phi}} \in \mathbb{R}^{k \times d}$ and $\mathbf{b}_{\boldsymbol{\phi}} \in \mathbb{R}^{k}$ are learned parameters of a single-layer MLP. They map a d dimensional vector to a k dimensional kernel feature space. The ReLU activation (Fukushima, 1980) ensures that the features are non-negative.[3] Different MLP parameters are used for different attention heads, and thus we add a total of $rk(d+1)$ learnable parameters per layer (less than 0.2% parameter increase in our language model, §3.3). We then finetune all parameters in this modified network, including the MLP parameters, with the original task objective.[4]

[3] We found that ReLU stabilized training by prohibiting negative similarities $\boldsymbol{\phi}(\mathbf{q}) \cdot \boldsymbol{\phi}(\mathbf{k})$. Other activation functions, such as cos, tanh, and elu, did not improve performance.

[4] We tried training the MLP parameters only, but this setting resulted in degraded development performance.

During inference generation, we reformulate the attention computation (Eq. 3.2a) as

$$\begin{aligned}\widetilde{\mathbf{x}}_i^{\text{out}} &= \sum_{j=1}^{M} \frac{\widetilde{\text{sim}}\left(\mathbf{q}_i, \mathbf{k}_j\right)}{\sum_{\ell=1}^{M} \widetilde{\text{sim}}\left(\mathbf{q}_i, \mathbf{k}_\ell\right)} \mathbf{v}_j \\ &= \left(\frac{\boldsymbol{\phi}\left(\mathbf{q}_i\right) \cdot \sum_{j=1}^{M} \boldsymbol{\phi}\left(\mathbf{k}_j\right) \otimes \mathbf{v}_j}{\boldsymbol{\phi}\left(\mathbf{q}_i\right) \cdot \sum_{\ell=1}^{M} \boldsymbol{\phi}\left(\mathbf{k}_\ell\right)}\right)^{\top}\end{aligned} \tag{3.4}$$

by the associativity of matrix multiplication. This formulation lends itself to recurrent computation. In causal attention where each query only attends to its prefix to predict the next word ($M = i$), define states:

$$\mathbf{S}_i = \sum_{j=1}^{i} \boldsymbol{\phi}\left(\mathbf{k}_j\right) \otimes \mathbf{v}_j, \quad \mathbf{z}_i = \sum_{j=1}^{i} \boldsymbol{\phi}\left(\mathbf{k}_j\right) \tag{3.5}$$

where $\mathbf{S}_i, \mathbf{z}_i \in \mathbb{R}^{k \times d}, \mathbb{R}^k$. These states can be computed recurrently (Katharopoulos et al., 2020):

$$\mathbf{S}_i = \mathbf{S}_{i-1} + \boldsymbol{\phi}\left(\mathbf{k}_i\right) \mathbf{v}_i^{\top} \quad \mathbf{z}_i = \mathbf{z}_{i-1} + \boldsymbol{\phi}\left(\mathbf{k}_i\right) \tag{3.6}$$

In the self-attention or encoder-to-decoder (cross) attention of a sequence-to-sequence model, $\mathbf{S}_i$ and $\mathbf{z}_i$ are constant with respect to i and only need to be computed once. Given the two states at position i, we can obtain the output vector:

$$\widetilde{\mathbf{x}}_i^{\text{out}} = \left(\frac{\boldsymbol{\phi}\left(\mathbf{q}_i\right)^{\top} \mathbf{S}_i}{\boldsymbol{\phi}\left(\mathbf{q}_i\right)^{\top} \mathbf{z}_i}\right)^{\top} \tag{3.7}$$

This avoids quadratic computation with respect to the input sequence length. We also speed up inference by merging the MLP feature map with the affine feature maps that produce queries and keys.

$$\boldsymbol{\phi}\left(\mathbf{q}_i\right) = \text{ReLU}\left(\widetilde{\mathbf{W}}_q \mathbf{x}_i^{\text{tgt}} + \widetilde{\mathbf{b}}_q\right), \tag{3.8a}$$

$$\boldsymbol{\phi}\left(\mathbf{k}_j\right) = \text{ReLU}\left(\widetilde{\mathbf{W}}_k \mathbf{x}_j^{\text{src}} + \widetilde{\mathbf{b}}_k\right), \tag{3.8b}$$

$$\text{where} \quad \widetilde{\mathbf{W}}_q = \mathbf{W}_\phi \mathbf{W}_q, \quad \widetilde{\mathbf{W}}_k = \mathbf{W}_\phi \mathbf{W}_k, \tag{3.8c}$$

$$\widetilde{\mathbf{b}}_q = \mathbf{b}_\phi + \mathbf{W}_\phi \mathbf{b}_q, \quad \widetilde{\mathbf{b}}_k = \mathbf{b}_\phi + \mathbf{W}_\phi \mathbf{b}_k. \tag{3.8d}$$

After the model is trained, Eqs. 3.8c–3.8d are computed once before generation; the intermediate features of $\mathbf{q}_i$ and $\mathbf{k}_j$ are never computed during inference.

Generation Speed Overhead The time complexity of each step in a T2R model is shown in Fig. 3.1. Similar to the transformer, it proceeds over two stages.

- **Feature Mapping**: computation of $\{\phi(\mathbf{q}_i)\}_{i=1}^N$, $\{\phi(\mathbf{k}_j)\}_{j=1}^M$, and $\{\mathbf{v}_j\}_{j=1}^M$ for all r heads (Eqs. 3.8a–3.8b). Time complexity of $\mathcal{O}(Nhkr)$, $\mathcal{O}(Mhkr)$, and $\mathcal{O}(Mh^2)$.
- **Attention**: the RNN states and the outputs for r heads (Eqs. 3.5–3.7) are computed with $\mathcal{O}(Mhk)$ and $\mathcal{O}(Nhk)$.

Comparing this with the pretrained transformer, we see that if the feature size is much smaller than input sequence lengths ($k \ll M, N$), the change in the attention stage from $\mathcal{O}(MNh)$ to $\mathcal{O}(hk(M+N))$ in T2R brings a substantial speedup.

Generation Memory Overhead T2R only needs to store the RNN state, and thus its space complexity is $\mathcal{O}(hk)$, constant in sequence length. This implies reduction in memory footprint when $k \ll M$, compared to the transformer's $\mathcal{O}(Mh)$.

3.2.3 Autoregressive Linear Transformers

In principle, any kernel function can be used as the similarity function in Eq. 3.2a (Tsai et al., 2019). Previous work proposed several untrainable feature map functions ϕ and developed autoregressive transformer variants with linear time and constant space complexity in sequence length (Katharopoulos et al., 2020; Peng et al., 2021; Choromanski et al., 2021). While those models follow similar computation steps to T2R, there are several differences in generation efficiency. Since the feature map in Katharopoulos et al. (2020) preserves input dimensions, the feature size is always the same as the head dimensions ($k = d$). This means that the speedup and memory savings from using a small feature size are restricted by design. In our experiments (§3.3.3), our T2R models gain further efficiency by using a feature size that is even smaller than the head

dimensions ($k = 32$ and $d = 128$ for language modeling). Peng et al. (2021) and Choromanski et al. (2021) scale query and key vectors by their norms before the random approximation to bound the error. Consequently, the feature mapping stage needs additional steps of producing intermediate $\mathbf{q}$ and $\mathbf{k}$ and scaling them. T2R suppresses these steps and speeds up generation further (§3.3.3).

3.3 Experiments

We present extensive experiments on standard benchmarks for language modeling and machine translation. Our results show that T2R achieves efficient autoregressive generation while retaining high accuracy.

3.3.1 Baselines and Comparison

We compare performance with previous transformer models for autoregressive generation with linear time and constant space complexity in input sequence length.[5] As discussed in §3.2.3, those prior methods correspond to two different untrainable feature maps ϕ. We experiment with two types of feature maps for comparisons: **ELU** ($\phi(\mathbf{x}) = \text{elu}(\mathbf{x}) + 1$, Katharopoulos et al., 2020); **RFA** (random feature approximation with softmax temperature reparameterization, Peng et al., 2021). Each feature map is evaluated in two settings: random initialization and pretrain. Random initialization is our reimplementation of the experiments in Katharopoulos et al. (2020) and Peng et al. (2021). The pretrain setting follows the same protocol as T2R except that we use different feature maps ϕ than our proposed one-layer MLP with ReLU activation. Positive orthogonal random features (**Performer**, Choromanski et al., 2021) provide similar random approximation to RFA and were evaluated in the biology domain, but we found that this method caused training divergence in the language modeling task.[6]

[5] See §3.5 for our discussion on more transformer variants with linear time complexity, but most of those variants need modifications for autoregressive modeling and have yet to be empirically evaluated in autoregressive generation tasks.

[6] Our implementation closely follows the code released by the authors (https://github.com/lucidrains/performer-pytorch/blob/main/performer_pytorch/performer_pytorch.py#L75-L81), but does *not* subtract the maximum logit; otherwise it would disallow the linear complexity in causal attention. We conjecture that this is the reason why Performer becomes less stable in our experiments. We suspect that some techniques are necessary to improve numerical stability in language modeling and machine translation.

3.3.2 Setup and Implementations

We apply our method to causal attention in language models and both cross and causal attention in machine translation. For language modeling, we use a 32-dimensional feature map function. We do not modify the encoder in machine translation as its generation speed overhead is much less significant than the decoder (Kasai et al., 2021a). Our exploration showed that reducing the feature size of causal attention tends to have less impact on the final translation accuracy as opposed to cross attention; we use feature sizes of 32 and 4 for cross and causal attention, respectively. This observation is consistent with previous work that showed that causal attention can be more drastically simplified than cross attention in transformer machine translation models (You et al., 2020; Tay et al., 2021).

Language Modeling

We use the WikiText-103 benchmark, which consists of 103M tokens sampled from English Wikipedia (Merity et al., 2017). We choose similar hyperparameters to prior work (Baevski and Auli, 2019; Fan et al., 2020): 32 layers, 8 heads, 128 head dimensions, 1024 model dimensions, 4096 fully connected dimensions and dropout (Srivastava et al., 2014) and layer dropout rates of 0.2. We partition the training data into non-overlapping blocks of 512 contiguous tokens ignoring document boundaries and train the model to predict each token from left to right (Baevski and Auli, 2019). Validation and test perplexity are measured by predicting the last 256 words out of the input of 512 consecutive words to avoid evaluating tokens in the beginning with limited context (*early token curse*, Press et al., 2021). We generally follow the optimization method from Baevski and Auli (2019), but some hyperparameters, such as the learning rate for the T2R fine-tuning, are adjusted for better convergence than randomly initialized training. See Appendix B.1 for more details.

Machine Translation

We experiment with 3 translation benchmarks: WMT14 EN-DE (4.5M train pairs, Bojar et al., 2016), WMT14 EN-FR (36M, Bojar et al., 2014), and WMT17 ZH-EN (20M, Bojar et al., 2017). We follow the preprocessing and data splits by previous work (EN-DE: Vaswani et al., 2017; EN-

FR: Gehring et al., 2017; EN-ZH: Hassan et al., 2018). We use the hyperparameters of the large sized transformer (Vaswani et al., 2017): 6 layers, 16 attention heads, 1024 model dimensions, and 4096 hidden dimensions for both the encoder and decoder. We apply dropout with 0.3 and label smoothing with $\varepsilon = 0.1$. Following Ott et al. (2018), we use an increased batch size of approximately 460K tokens. Each randomly initialized model is trained for 30K (60K for the large EN-FR dataset) steps using Adam with a learning rate of $5 \cdot 10^{-4}$ and $\beta = (0.9, 0.98)$ (Kingma and Ba, 2015). We observed that convergence of the T2R conversion can be achieved with 20K (40K for EN-FR) steps and a reduced learning rate of $2 \cdot 10^{-4}$. We average the checkpoints from the last five epochs to obtain the final model (Vaswani et al., 2017). In inference, we apply beam search with size 5 and length penalty 0.6. Consistent with previous practice, we evaluate with tokenized BLEU (Papineni et al., 2002). Further details are described in Appendix B.1.

3.3.3 Results

		Perplexity		**Training Time**
Model	k	dev.	test	(GPU Hours)
ELU + Random Init.	128	22.0	22.8	470h
RFA + Random Init.	32	20.4	21.3	512h
T2R + Random Init.	32	20.1	20.8	474h
ELU + Pretrain	128	21.5	22.2	97h
RFA + Pretrain	32	20.8	21.6	104h
T2R + Pretrain	32	**19.0**	**19.6**	98h
T2R 75% + Pretrain	32	17.9	18.5	95h
Pretrained Transformer	–	17.9	18.5	–
Baevski and Auli (2019)	–	–	18.7	–

Table 3.1: WikiText-103 language modeling results (perplexity). Training time is measured in GPU hours. The top two rows are our reimplementations of Katharopoulos et al. (2020) and Peng et al. (2021). Pretrain indicates initialization with a pretrained transformer for language modeling. T2R 75% indicates a model where every fourth layer from the top is kept as the original transformer layer. Perplexity is measured by predicting the last 256 words out of the input of 512 consecutive words. All models use 128 head dimensions. We assume access to a pretrained transformer model and measure the finetuning time in GPU hours.

Language Modeling Seen in Table 3.1 are language modeling results in perplexity. We observe that T2R with the learnable MLP feature map outperforms the other two linear transformer models by more than 2.0 perplexity points in the pretrain setting. Unlike the other linear transformer

models, T2R greatly benefits from pretraining (T2R + Pretrain: 19.6 vs. T2R + Random Init.: 20.8 test perplexity points). We attribute this advantage of T2R to the fact that the MLP feature map is able to learn attention patterns that are similar to those of the pretrained transformer, as evidenced in §3.4. Notice also that the T2R conversion is ∼5x faster (measured in GPU hours) than training a model from scratch. These results illustrate that a lightweight model can be obtained without repeating the expensive training of large-scale pretrained language models such as GPT-2 and GPT-3 (Radford et al., 2019; Brown et al., 2020). T2R's generation speedup (∼4x when producing 512 consecutive words) and memory savings are later benchmarked with varying sequence lengths. There remains a gap of 1.1 perplexity points between the T2R and pretrained transformer models (19.6 vs. 18.5). However, the gap can be closed when every fourth layer from the top is kept as the original transformer layer and the model is finetuned in the same way (T2R 75%). This suggests that keeping a small fraction of the quadratic attention layers can provide an effective middle ground between efficiency and accuracy.[7]

	Feature Size k		**WMT14**		**WMT17**	**Train Time**
Model	cross	causal	EN-DE	EN-FR	ZH-EN	(GPU hours)
ELU + Random Init.	64	64	28.4	*	23.4	120h
RFA + Random Init.	32	4	28.1	41.7	23.4	135h
T2R + Random Init.	32	4	27.5	39.8	23.1	123h
ELU + Pretrain	64	64	28.4	41.8	**23.8**	80h
RFA + Pretrain	32	4	27.6	41.8	23.2	90h
T2R + Pretrain	32	4	**28.7**	**42.1**	**23.8**	82h
Pretrained Transformer Large	–	–	28.9	42.2	24.2	–
Vaswani et al. (2017)	–	–	28.4	41.8	–	–

Table 3.2: Machine translation test results in BLEU scores. The top two rows are our reimplementations of Katharopoulos et al. (2020) and Peng et al. (2021). Pretrain indicates initialization with a trained transformer-large model. *: diverged even when running with multiple random seeds and smaller learning rates. We assume access to a pretrained transformer model and measure the finetuning time in GPU hours.

Machine Translation Seen in Table 3.2 are machine translation results in BLEU from various configurations. Departing from the language modeling experiments, the T2R model underperforms the other two linear transformer models when initialized randomly. However, consistent with language modeling, the T2R model substantially benefits from pretraining (e.g., 28.7 vs. 27.5

[7]Concurrent work (Lei, 2021) also explores reducing the number of attention layers for efficiency.

BLEU points in EN-DE). As a result, the T2R model achieves similar BLEU scores to the original transformer across all language pairs. ELU trained from the pretrained transformer yields comparable performance to T2R, but the feature size is much larger (64 vs. 32 and 64 vs. 4 in cross and causal attention), thus leading to increased overhead, as shown later. Note that the T2R finetuning time is only moderately smaller than that of randomly initialized training here, but further speedup in conversion can be potentially achieved with more extensive hyperparameter tuning.[8]

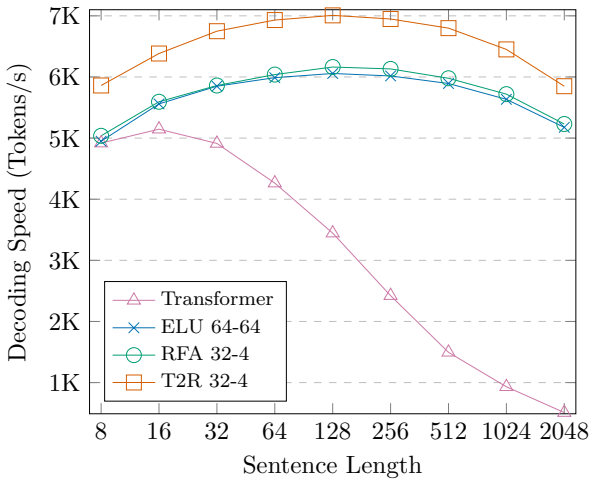


Figure 3.2: Machine translation speed of various models. Speed is measured on a single TPU v2 accelerator with batch size 16 and beam size 1, following Peng et al. (2021). 32-4 indicates the feature sizes of 32 and 4 for cross and causal attention, respectively.

Speedup and Memory Savings in Generation We run a conditional generation experiment to compare the decoding speed of the models in Table 3.2 (Fig. 3.2). Here we assume the input and output sequences are of the same length. All models are tested using greedy decoding with the same batch size of 16 on a TPU v2 accelerator.[9] We see that indeed the linear transformer models can generate an almost constant number of tokens per second regardless of the sequence length and outpace the transformer model dramatically as the sequence becomes longer. The T2R model achieves a 15%+ speedup over ELU and RFA due to its smaller feature sizes and faster feature mapping respectively; this confirms our analysis on T2R's speed advantage over them

[8] We found that the batch size could be reduced for T2R conversion without hurting accuracy, while randomly initialized models deteriorate with small batch sizes. This suggests that the computational cost for conversion can be much lighter than training from scratch, and T2R is advantageous when only a limited number of GPUs are available.

[9] https://opensource.google/projects/jax.

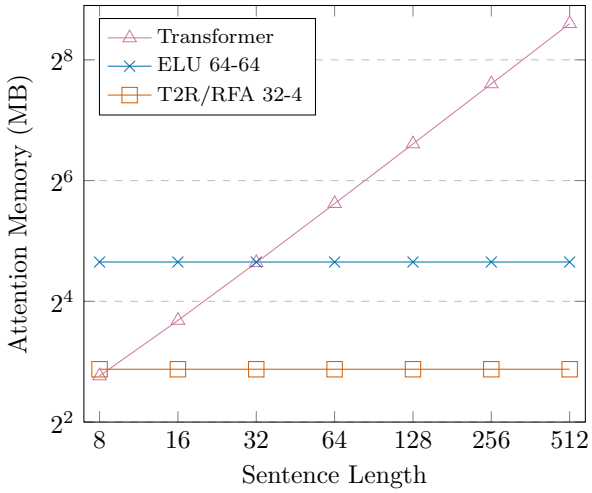


Figure 3.3: Memory consumption from the attention computation of various machine translation models in inference with batch size 16 and beam size 1.

(§3.2.3). Fig. 3.3 plots memory consumption from the attention computation during decoding for machine translation. Since the T2R, RFA, and ELU models compress keys and values into a $k \times d$ matrix $\mathbf{S}$ and a k dimensional vector $\mathbf{z}$ (§3.2.2), the required memory at each decoding step is constant over varying sequence lengths. It is also roughly proportional to the feature size k. The MLP feature map in the T2R model allows for small feature dimensions than the ELU feature of the head dimensions, resulting in a 70% memory reduction. The attention computation in the standard transformer, on the other hand, consumes memory linearly in sequence length at each decoding step because all previous key and value vectors have to be stored. We also found a similar speedup and memory savings in unconditional generation with the T2R language model ($\sim$4x speedup in generating 512 consecutive words over the transformer).

3.4 Analysis and Ablations

We presented T2R, a method to convert a pretrained transformer into an efficient RNN. In this section, we analyze our conversion approach by examining the impact of the feature size and induced attention weight distributions. Our analysis shows that T2R implicitly learns attention distributions similar to the original transformer.

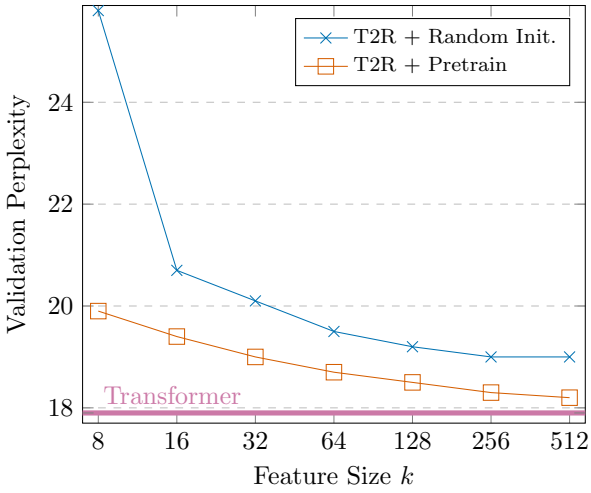


Figure 3.4: WikiText-103 validation perplexity with varying feature sizes.

Feature Size and Pretraining We saw that T2R benefits substantially from transformer pretraining. Fig. 3.4 compares T2R with pretraining and random initialization in terms of the relation between the validation perplexity from WikiText-103 and the feature sizes. We see that as the feature size (RNN state size) becomes smaller, pretraining becomes particularly important to achieve low perplexity. Transformer pretraining achieves a Pareto improvement over random initialization in the tradeoff between efficiency (small feature size) and accuracy (low perplexity).

Attention Distribution T2R is not explicitly trained to mimic the original attention distributions, and there is no guarantee that the MLP feature map approximates the exponential similarity function, unlike previous approximation approaches (Peng et al., 2021; Choromanski et al., 2021). Here, we analyze the properties of the attention weight distributions that are induced by finetuning. We use the validation data from WikiText-103 and run language models to predict the next word given the input of 512 contiguous words. We compute the attention weight distribution over the 512 words for each attention head in the model layers.

Fig. 3.5 compares the attention distributions from T2R in various configurations. T2R MLP frozen indicates a model that is finetuned with the MLP parameters frozen. Euclidean distances in attention distributions between the original transformer and each model are averaged across validation samples, model layers, and attention heads.[10] Comparing T2R before finetuning and

[10]We do not consider random initialization baselines here because random initialization makes it impossible to align attention heads and layers between models.

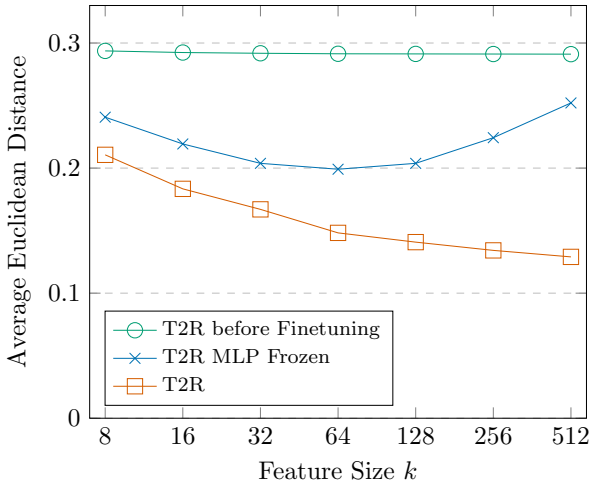


Figure 3.5: Average Euclidean distance of T2R models from the transformer attention weights with varying feature sizes. The distances are computed on the Wikitext-103 validation data for predicting a word given the preceding 512 words. All models are initialized with a pretrained transformer model.

the full T2R model, we see that the finetuning process induces much more similar attention distributions, and the distance diminishes as the feature size increases (and the perplexity approaches the original transformer, Fig. 3.4). We also observed that when the MLP parameters are not trained (T2R MLP frozen), the distance from the original attention distributions increases. These results suggest that finetuning of the whole network in T2R implicitly develops similar attention distributions to the original transformer even though the training supervision comes solely from language modeling.

3.5 Related Work

In addition to the work we already discussed, we highlight related methods from prior work that make transformer models efficient.

3.5.1 Knowledge Distillation

Knowledge distillation (Hinton et al., 2015) is closely related to our T2R conversion and uses a similar pipeline: a teacher model with large capacity is first trained and is used to generate *silver* training data for a new lightweight inference model. It has been successfully applied to machine translation (e.g., Kim and Rush, 2016; Gu et al., 2018) to make generation efficient. In particular, several prior works distill a transformer translation model to an RNN (Senellart et al.,

2018; Kim et al., 2019). We share the same motivation toward fast generation with light memory, but our approach differs in two ways: the original training data are used for finetuning an RNN model, and its model parameters are initialized with the "teacher" transformer. Our method does not use the computationally expensive teacher model to generate new training data. While data generation is a one-time computational cost, it becomes expensive as the teacher model size and training data increase. Moreover, since the pretrained parameters can be directly used, conversion requires fewer GPU hours than training a brand new lightweight model from scratch (§3.3.3).

3.5.2 Efficient Transformers

Prior work suggested many other strategies to improve efficiency in transformers, such as weight sharing and factorization (Dehghani et al., 2019; Lan et al., 2020), weight and layer pruning (Michel et al., 2019; Fan et al., 2020), quantization (Zafrir et al., 2019; Shen et al., 2020), and modifying the combination of sublayers (Press et al., 2020; Mandava et al., 2020). Some of these methods present orthogonal design choices and can be integrated into our T2R model to gain further efficiency. For a more comprehensive survey, see Tay et al. (2020b). Below we describe several prior works along two major strategies: compressing the attention context and sparsifying the attention patterns.

Attention Context Compression This strand of methods compresses the context that is attended to, thereby reducing the time and memory overhead in the attention. RNN models that we converted pretrained transformers into compress the context into a recurrent state. Other approaches include low rank approximation of the attention computation (Wang et al., 2020a; Tay et al., 2021) and adding a memory module that can access multiple tokens at once (Liu et al., 2018; Dai et al., 2019; Lee et al., 2019; Ainslie et al., 2020; Rae et al., 2020; Beltagy et al., 2020; Zaheer et al., 2020).

Sparse Attention Patterns Another approach to reducing the time and memory overhead from the attention computation is to limit the tokens that are attended to by sparsifying the attention patterns. These patterns can be set in advance or learned during training (Tay et al., 2020b).

For example, prior works introduced fixed patterns of blockwise attention (Qiu et al., 2020) and strided attention (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020). Other previous works presented methods to learn attention patterns from data (Sukhbaatar et al., 2019; Roy et al., 2020; Tay et al., 2020a).

It should be noted that significant modifications are necessary to apply many of these methods to autoregressive generation tasks such as language modeling and machine translation, and their empirical evaluation in these generation settings has yet to be conducted (Peng et al., 2021). This chapter presents extensive empirical evaluation in autoregressive generation settings.

3.6 Summary

In this chapter, we presented T2R, a method that converts a pretrained transformer to a recurrent neural network that reduces the time and memory cost of autoregressive generation. Our experiments in language modeling and machine translation demonstrated that our model produces an improved tradeoff between efficiency and accuracy over randomly initialized training and previous models with lightweight attention. Our results provide further support for the claim that large-scale pretrained models can be compressed into efficient inference models that facilitate downstream applications.

Part II

Customizable Inference Algorithms for Natural Language Generation

Chapter 4

Twist Decoding: Diverse Generators Guide Each Other

Modern artificial intelligence (AI) and natural language processing (NLP) models undergo two stages: **training**, where model parameters are learned based on large datasets, and **inference**, where the model is used to generate desired outputs. Inference is thus a core algorithmic component of NLP systems that has been actively studied in NLP research. Generation of language handles complex structure and exponentially many possibilities; for a given output length N and vocabulary size V, there are V^N possibilities, making it impossible to enumerate and find the best one. Several inference algorithms (e.g., greedy/beam search) have proven effective on a variety of language generation tasks and continue to be used in recent large-scale models, such as GPT-3 (Brown et al., 2020) and Google Translate (Wu et al., 2016). Overcoming the limitations of commonly used algorithms, I develop Twist decoding, a simple yet effective inference algorithm that enables customization with additional training.

4.1 Introduction

Natural language generation is an important building block for many applications, such as machine translation, summarization, and question answering (Ng et al., 2019; Lewis et al., 2020; Raffel et al., 2020; Brown et al., 2020; Asai et al., 2021b, *inter alia*). Researchers have recently

The material in this chapter is adapted from Kasai et al. (2022b).

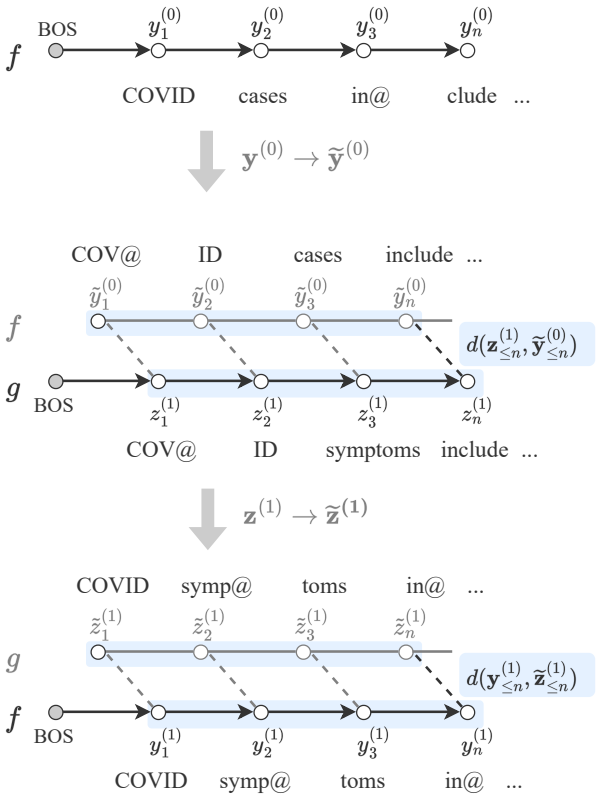


Figure 4.1: TWIST decoding of two generation models, f and g, that does not assume a shared vocabulary, tokenization, or generation order. Beam search is first applied to f to generate $\mathbf{y}^{(0)}$, followed by output mapping to $\widetilde{\mathbf{y}}^{(0)}$ (e.g., f's detokenization and g's tokenization or sequence reversal). g is then decoded with beam search augmented with distances from the set of previously-generated outputs (here only one sequence $\mathbf{y}$ is shown): $d(\mathbf{z}^{(1)}_{\leq n}, \widetilde{\mathbf{y}}^{(0)}_{\leq n})$. Subsequently, f is similarly decoded with g's guidance. Here we show one iteration that already achieves substantial improvements (§4.4). @ indicates the BPE separator.

explored and advanced models for generation in various aspects, including model architecture (Bahdanau et al., 2015; Vaswani et al., 2017), domain adaptation (Chu and Wang, 2018; Bapna and Firat, 2019), prompting (Brown et al., 2020), and even generation order (Gu et al., 2018). The resulting generation models are diverse, trained on different data, with different assumptions, at different times. We hypothesize that diverse generation models may achieve better results through ensembling, if the various approaches have complementary strengths. Given the high cost of unifying approaches during training time (Strubell et al., 2019; Schwartz et al., 2019), inference-time combination of existing models is an attractive alternative.

One well-established ensembling technique is "shallow fusion" (Sutskever et al., 2014; Gulcehre et al., 2015; Firat et al., 2016, *inter alia*), which aggregates models' scores during beam search. This approach requires, however, that the models use the same vocabulary/tokenization

scheme and organize the search in the same way (e.g., autoregressive, left-to-right factorization).

We introduce a new inference algorithm, Twist decoding (Fig. 4.1), that enables more diverse generators to guide each other. Twist decoding can combine generators with different vocabularies, (de)tokenization, and even generation order without any additional training or finetuning. Our method decodes a model by standard beam search, but the scores at every step incorporate a simple function that measures the distance from outputs of the other model. We run this procedure on each generation model in turn, so that both can benefit from each other.

We present extensive experiments on machine translation and scientific paper summarization and show that Twist decoding can improve performance over each model decoded in isolation across several scenarios: combining 1) generic and domain-specific models, 2) left-to-right and right-to-left generation models, and 3) models that generate using different conditioning inputs. Our results show consistent performance gains from combining generic and domain-specific translation models over a wide range of domains, including medical and legal translation. Applications in these domains require particularly high accuracy, and Twist decoding is a desirable alternative to standard beam search on a single model. Interestingly, we find that Twist decoding between generic and domain models is effective even when parallel data from the domain are scarce and the domain model yields poor performance by itself, suggesting complementary strengths of diverse generators (§4.3.4).

Twist decoding can be seen as a generalization of reranking heuristics that have proven effective in syntactic parsing (Shen and Joshi, 2003; Charniak and Johnson, 2005; Collins and Koo, 2005), speech recognition (Collins et al., 2005), and machine translation (Shen et al., 2004; Och et al., 2004): one model generates candidate sequences, followed by rescoring from another model. We present extensive comparisons with reranking baselines and demonstrate that Twist decoding outperforms reranking consistently. We also observe that since the encoder computations on two models can be parallelized, the inference time required for Twist decoding is much shorter than the sum of the two models, resulting only in a 50% increase, relative to decoding of a single model in isolation (§4.4). Twist decoding is therefore a viable alternative to standard beam search on a single model and the widespread reranking heuristic.

Twist Decoding
g generates $\mathbf{z}$ with guidance from f at iteration t

k: beam size. M: maximum length.
$\mathcal{V}_g$: vocabulary of g. $g(\cdot)$: scoring function.
$\mathcal{Y}^{(t-1)}$: set of output sequences from f. $\mathcal{Z}^{(t)}$: new outputs.
B_n: beam of continuing sequences.
H: expanded hypotheses before beam selection.
$d(\cdot,\cdot)$: distance between partial sequences.
λ_f: scalar coefficient for the distance.

$\widetilde{\mathcal{Y}}^{(t-1)} = \left\{\widetilde{\mathbf{y}} = \text{map\_output}(\mathbf{y}) \mid \mathbf{y} \in \mathcal{Y}^{(t-1)}\right\}$
$B_0 \leftarrow \{\text{BOS}\},\ \mathcal{Z}^{(t)} \leftarrow \varnothing$
for $n \in \{1, \ldots, M\}$ **do**
  $H \leftarrow \varnothing$
  for $\mathbf{z} \in B_{n-1}$ **do** # Expansion.
    for $z \in \mathcal{V}_g$ **do**
      $s \leftarrow g(\mathbf{z} \circ z) - \lambda_f \min_{\widetilde{\mathbf{y}} \in \widetilde{\mathcal{Y}}^{(t-1)}} d\left(\mathbf{z} \circ z, \widetilde{\mathbf{y}}_{\leq n}\right)$
      H.add$(\langle s, \mathbf{z} \circ z \rangle)$
    end for
  end for
  $B_n \leftarrow \text{topk}(H),\ \mathcal{Z}^{(t)}$.add $(\text{finished}(H))$
end for
return $\mathcal{Z}^{(t)}$

Figure 4.2: Twist decoding when g is guided by f. Swap f and g and $\mathbf{y}$ and $\mathbf{z}$ to obtain $\mathcal{Y}^{(t)}$. map_output converts outputs from f to g; e.g., f's detokenization followed by g's tokenization. It would also include sequence reversal if f or g is a right-to-left model. The highlighted line is the *only* modification that Twist decoding introduces to standard beam search. The input sequence to g is omitted. See also Kasai et al. (2022c) for the stopping criterion and implementation details (the *first come, first served* heuristic).

4.2 Twist Decoding

We propose Twist decoding, a general decoding algorithm that generates text from diverse models without assumptions of a shared vocabulary, tokenization, or generation order. At the core of the algorithm is a simple modification in standard beam search (highlighted in Fig. 4.2); we incorporate into a scoring function the distance from outputs that are previously generated by another model.

4.2.1 Initial Decoding

Let us assume that we have two generation models: f and g.[1] Both f and g assign scores to output sequences.[2] For example, f can be a domain-specific translation model and g a generic one. f and g perform their own pre/postprocessing (e.g., (de)tokenization) and factorization (e.g., left-to-right or right-to-left factorization). Here we suppress for brevity the conditional dependence on the input (e.g., machine translation input from the source language). Standard beam search with a beam size k is first applied to f to produce a set of k output sequences: $\mathcal{Y}^{(0)}$. This approximately solves $\text{topk}_{\mathbf{y}} f(\mathbf{y})$ by pruned breadth-first search, and often returns higher-quality outputs than the exact search counterpart (Stahlberg and Byrne, 2019; Meister et al., 2020a).

4.2.2 Mutually-Guided Decoding

Once $\mathcal{Y}^{(0)}$ is obtained, we proceed with decoding generators with mutual guidance (Fig. 4.2; $t \geq 1$).

Output Sequence Mapping. The commonly-used technique of ensembling (Sutskever et al., 2014) or shallow fusion (Gulcehre et al., 2015; Stahlberg et al., 2018) adds scores from f and g at every step and executes the same search algorithm to approximately solve $\text{topk}_{\mathbf{y}} f(\mathbf{y}) + g(\mathbf{y})$. This method thus necessitates a shared vocabulary, tokenization, and generation order (Imamura and Sumita, 2017). We relax this assumption and first map the candidates in $\mathcal{Y}^{(t-1)}$ to output sequences for g: $\widetilde{\mathcal{Y}}^{(t-1)}$ (Line 1 in Fig. 4.2). This mapping (map_output) typically involves deterministic operations of f's detokenization followed by g's tokenization. Sequence reversal is also performed if f and g generate in the opposite order. For example, if g uses byte-pair encoding (Sennrich et al., 2016b), but f does not, we might have $\mathbf{y}$=*John does n't like Mary* mapped to $\widetilde{\mathbf{y}}$=*Jo@ hn doesn't like Mar@ y*, where @ denotes subword separation.

[1] The algorithm can be readily extended to three or more generators. We also abuse f or g to mean both the generator and its scoring function.

[2] They typically assign log-probabilities, but it is not necessary to assume the scores form a valid probability distribution.

Decoding with Distance Terms. We then decode g with guidance from $\widetilde{\mathcal{Y}}^{(t-1)}$. Specifically, we perform beam search with a simple modification in scoring (Line 7). In this chapter, we use a simple distance measure that adds binary distances at all positions (i.e., the Hamming distance):

$$d\left(\mathbf{z}_{\leq n}, \widetilde{\mathbf{y}}_{\leq n}\right) = \sum_{i \leq n} \mathbb{1}\left\{z_i \neq \widetilde{y}_i\right\}$$

We also explored using the distance between (sub)word embeddings from the model: $\sum_{i \leq n} \| e(z_i) - e(\widetilde{y}_i) \|_2$, but this did not bring improvements (§4.4). Note also that when i exceeds the length of $\widetilde{\mathbf{y}}$, we assume $\widetilde{y}_i = \textsc{eos}$. The overall distance term is then

$$\min_{\widetilde{\mathbf{y}} \in \widetilde{\mathcal{Y}}^{(t-1)}} d\left(\mathbf{z}_{\leq n}, \widetilde{\mathbf{y}}_{\leq n}\right)$$

Here we minimize over the output sequences to compute the distance to the closest candidate. These candidates from $\widetilde{\mathcal{Y}}^{(t-1)}$ can be equally good outputs but differ only by one word; in such cases, this minimization operation avoids overestimation of the distances. The new score at step n in beam search is now computed by:

$$g(\mathbf{z}_{\leq n}) - \lambda_f \min_{\widetilde{\mathbf{y}} \in \widetilde{\mathcal{Y}}^{(t-1)}} d\left(\mathbf{z}_{\leq n}, \widetilde{\mathbf{y}}_{\leq n}\right),$$

where λ_f is a scalar coefficient for the distance term that controls the importance of f relative to g. We tune $\lambda_f \in \{0.1, 0.3, 1.0, 3.0\}$ during development. After this beam search, we obtain a new candidate set, $\mathcal{Z}^{(t)}$. We then run the same beam search (Fig. 4.2) with the roles of f, $\mathcal{Y}$ and g, $\mathcal{Z}$ swapped.[3] Namely, we decode f with distance terms from $\mathcal{Z}^{(t)}$ at each step of beam search:

$$f(\mathbf{y}_{\leq n}) - \lambda_g \min_{\widetilde{\mathbf{z}} \in \widetilde{\mathcal{Z}}^{(t)}} d\left(\mathbf{y}_{\leq n}, \widetilde{\mathbf{z}}_{\leq n}\right)$$

Finally, the highest-scoring sequence from $\mathcal{Y}^{(t)}$ is output. This process of mutually-guided decoding can be repeated multiple times. We observe, however, that one iteration ($t = 1$) suffices to bring performance gains (§4.4). We also present detailed sensitivity analysis over varying λ_f and λ_g and find that Twist decoding is particularly effective when $\lambda_g > \lambda_f$ (i.e., initial exploration

[3] We can stop inference with $\mathcal{Z}^{(t)}$, but we found that led to performance degradation in preliminary development.

by g is encouraged with relatively little guidance from f's original outputs; see §4.4).

Reranking Heuristic as a Special Case. Notice that as $\lambda_f \to \infty$, g's generation falls back to a reranking heuristic: top k sequences from the initial f decoding are reranked according to g. This reranking heuristic has proven successful in a wide range of sequence generation tasks, including machine translation (Shen et al., 2004), syntactic parsing (Collins and Koo, 2005), and speech recognition (Collins et al., 2005). Reranking is performed in many strong machine translation systems to use a right-to-left model to improve a left-to-right model; e.g., top-performing systems in recent WMT competitions (Ng et al., 2019; Kiyono et al., 2020; Wang et al., 2021; Akhbardeh et al., 2021). In our experiments, we extensively compare performances of Twist decoding and reranking and demonstrate that the former consistently outperforms the latter.

4.3 Experiments

We present experiments across three scenarios: combining domain and generic models for machine translation (§4.3.1), left-to-right and right-to-left machine translation models (§4.3.2), and scientific paper summarization models that take as input different parts of the paper (§4.3.3). We empirically compare Twist decoding with decoding in isolation and the widely-adopted reranking baselines, illustrating that Twist decoding offers performance improvements in various situations *without* any change to the trained models.

4.3.1 Domain and Generic Models

Machine translation has now been used for many domains, ranging from everyday conversations to medical documents. Machine translation models are often trained on large amounts of parallel data, such as the Europarl corpus (Koehn, 2005) and the OPUS data (Tiedemann, 2012). Applying these models to out-of-domain data remains a challenge (Koehn and Knowles, 2017; Chu and Wang, 2018), and users for some of these domains require high accuracy in translation (e.g., medical and legal documents). We will demonstrate that Twist decoding between general-purpose and domain-specific models is a viable approach to tackle this problem.

Setups. We use machine translation datasets over diverse domains from prior work (Koehn and Knowles, 2017; Hu et al., 2019): German→English over medical (1.1M training sentence pairs), legal (720K pairs), Koran (religious text, 480K pairs), and subtitles (14M pairs) domains.[4] For the domain-specific models, we train a base-sized transformer model (Vaswani et al., 2017) with a 6-layer encoder and a 6-layer decoder on the training data of each domain. The top-performing German→English system from WMT19 (Barrault et al., 2019; Ng et al., 2019)[5] is used as the generic model. This generic model is a large-sized transformer trained on a concatenation of publicly available parallel data, including the Europarl (Koehn, 2005) and UN (Ziemski et al., 2016) corpora with the backtranslation technique (Sennrich et al., 2016a). We follow (de)tokenization (Koehn et al., 2007) and byte-pair encoding (Sennrich et al., 2016b) of previous work (Koehn and Knowles, 2017; Hu et al., 2019).[6]

For every domain, we evaluate a total of six configurations: decoding of the generic and domain models each in isolation; the reranking baseline and Twist decoding with f being the generic model and g being the domain model, as well as the versions where f and g are swapped to see the effect of the two roles. In all cases, we use beam size 5 (Freitag and Al-Onaizan, 2017) and length penalty 1 (Wu et al., 2016) and conduct all experiments using the `fairseq` library (Ott et al., 2019). All performance is measured with the COMET score (Rei et al., 2020a,b) and the SacreBLEU implementation (Post, 2018) of the BLEU score (Papineni et al., 2002). Note that COMET is based on crosslingual contextual representations (Conneau et al., 2020), and recent work showed that it achieves significantly higher correlation with expert human judgment than BLEU and other n-gram-based metrics (Kasai et al., 2022a,d). More experimental details are described in Appendix §C.1.1.

Results. Seen in Table 4.1 are the results from our experiments over various domains. Firstly, given two translation models f and g, Twist decoding outperforms the reranking baseline in all configurations (indicated in blue) with only one exception (a small drop in BLEU in the subtitles domain). Particularly noteworthy are the gains in the medical domain: Twist decoding outperforms the reranking heuristic by 5.8 COMET and 1.4 BLEU points when f is the domain

[4] We excluded the IT domain because we found significant overlap between training and dev./test data.
[5] https://github.com/pytorch/fairseq/tree/main/examples/wmt19.
[6] https://github.com/JunjieHu/dali.

German→English			Medicine		Law		Koran		Subtitles	
Method	f	g	COMET	BLEU	COMET	BLEU	COMET	BLEU	COMET	BLEU
Isolation	Generic	–	44.5	41.2	30.6	34.8	14.4	16.6	34.4	31.3
Isolation	Domain	–	80.7	48.3	60.7	40.9	8.7	17.0	32.3	29.0
Rerank	Generic	Domain	59.6	43.5	56.4	36.1	14.7	17.0	40.3	**32.3**
Twist	Generic	Domain	71.6	47.5	61.4	40.2	**16.5**	18.5	**41.0**	32.2
Δ (**Twist**− Rerank)			+12.0	+4.0	+5.0	+4.1	+1.8	+1.5	+0.7	−0.1
Rerank	Domain	Generic	75.8	48.7	59.9	40.8	12.3	18.1	36.5	30.3
Twist	Domain	Generic	**81.6**	**50.1**	**61.6**	**41.3**	15.3	**18.7**	37.3	31.0
Δ (**Twist**− Rerank)			+5.8	+1.4	+1.7	+0.5	+3.0	+0.6	+0.8	+0.7

Table 4.1: Combination of generic and domain-specific translation models. The generic model is the top-performing translation model in WMT19 (Ng et al., 2019) that is trained on a collection of parallel corpora, such as the Europarl and the UN corpora. Two settings are considered for the reranking baseline and Twist decoding: f is the generic model and g is the domain model or the reverse. The best scores are in **bold**. COMET (Rei et al., 2020a,b) uses crosslingual contextual representations (Conneau et al., 2020) and achieves significantly higher correlation with expert human judgment than BLEU (Papineni et al., 2002) and other alternative metrics (Kasai et al., 2022a,d).

model and g is the generic model. Twist decoding is thus an effective generalization over the reranking heuristic commonly used in the literature across domains.

Comparing the performance of decoding in isolation and Twist decoding, we observe that the best score from Twist decoding substantially outperforms each individual model over all domains: e.g., 81.6 vs. 80.7 (domain model) and 81.6 vs. 44.5 (generic model) COMET points in the medical domain. In both medical and legal domains, the generic model underperforms the domain model by a large margin. Nonetheless, Twist decoding between the two improves over the domain model, suggesting that Twist decoding makes use of their complementary strengths. Finally, we see a consistent pattern regarding f and g: both Twist decoding and the reranking baseline perform better when the higher-performing model is chosen as f. (e.g., the domain model performs better in medicine and law, and vice versa in subtitles.) This is expected because f is used both for initial decoding and final decoding with g's guidance (Fig. 4.1).

4.3.2 Left-to-Right and Right-to-Left Models

Language generation models usually factorize sequences autoregressively in a left-to-right order, but previous work showed that left-to-right (L2R) models can be improved by reranking their outputs with a separate right-to-left (R2L) model (Imamura and Sumita, 2017; Ng et al., 2019;

Kiyono et al., 2020, *inter alia*). Twist decoding can be readily applied to such scenarios since it does not assume shared generation order between models.

Setups. We experiment with two language pairs from the WMT 2020 news translation task (Barrault et al., 2020): Chinese→English (**WMT20 ZH-EN**, 48M training sentence pairs) and English→German (**WMT20 EN-DE**, 48M pairs). Submissions for these language pairs to the shared task have human evaluations from professional translators (Freitag et al., 2021), and the correlation between automatic metrics and the human ratings are studied in subsequent work (Kasai et al., 2022a); COMET (Rei et al., 2020b,a) achieves the highest correlation out of the 15+ metrics.

Similar to the previous experiments, we measure all performance using COMET and BLEU scores. Note that we use two reference translations per instance for WMT20 ZH-EN and three for WMT20 EN-DE, following Kasai et al. (2022a). They both have reference translations from two different services, and WMT20 EN-DE has an additional translation created by linguists who are asked to paraphrase the two translations as much as possible. These paraphrased translations are shown to increase correlation with human judgments by mitigating the translationese effect (Graham et al., 2020) and diversifying the reference (Freitag et al., 2020). On each dataset, we follow the preprocessing and tokenization (Koehn et al., 2007; Sennrich et al., 2016b) from Kasai et al. (2022a)[7] and train a large-sized transformer model for left-to-right and right-to-left translation, in which the output English/German sequences are reversed after tokenization. We implement all models and decoding with `fairseq` and apply beam search with beam size 5 and length penalty 1. We again consider a total of six settings: reranking and Twist decoding with L2R as f and R2L as g or the reverse, as well as the individual models. Further details can be found in Appendix §C.1.2.

Results. Table 4.2 shows the results from L2R and R2L translation models. Twist decoding again outperforms the reranking counterpart by a considerable margin in COMET and BLEU on both language pairs; e.g., 43.1 vs. 41.2 COMET points on WMT20 ZH-EN when f is R2L and g is L2R. The best performance is achieved by Twist decoding on both datasets and improves over

[7] https://github.com/jungokasai/billboard/tree/master/baselines.

WMT20 Test			ZH→EN		EN→DE	
Method	f	g	COMET	BLEU	COMET	BLEU
Isolation	L2R	–	40.8	35.5	42.9	45.5
Isolation	R2L	–	40.4	35.0	43.3	44.9
Rerank	L2R	R2L	41.4	36.1	43.7	46.0
Twist	L2R	R2L	42.8	**36.8**	**45.4**	**46.7**
Δ (**Twist** − Rerank)			+1.4	+0.7	+1.7	+0.7
Rerank	R2L	L2R	41.2	35.4	44.7	45.2
Twist	R2L	L2R	**43.1**	**36.8**	44.8	46.0
Δ (**Twist** − Rerank)			+1.9	+1.4	+0.1	+0.8

Table 4.2: Combination of left-to-right (L2R) and right-to-left (R2L) transformer translation models. ZH: Chinese. DE: German. Two settings are considered for reranking and our Twist decoding each: L2R or R2L as f. The best scores are in **bold**.

the individual models by more than 1 BLEU point. The reranking baseline, on the other hand, does not outperform L2R in BLEU when f is R2L: 35.4 vs. 35.5 (ZH-EN) and 45.2 vs. 45.5 (EN-DE). This result illustrates that Twist decoding is a more effective approach to combine models with different generation order than the popular reranking.

4.3.3 Summarization with Different Input

We also experiment with strong models on a highly abstractive scientific paper summarization task: **SciTLDR** (Cachola et al., 2020). Specifically, we use two BART-based models from prior work (Cachola et al., 2020) that differ in input type: one that only takes as input the paper abstract (Abst.) and the other a concatenation of the abstract, introduction, and conclusion (AIC).[8]

Setups. We use the train/dev./test split from Cachola et al. (2020). Again following Cachola et al. (2020), we use all human-written summaries (written either by authors or undergraduate computer science students) as the reference and evaluate performance in terms of the ROUGE score (Lin, 2004).[9] We average the instance-level scores from the Python rouge-score implementation.[10] Similar to our previous experiments, we use beam size 5 and length penalty 1. See more detail in Appendix C.1.3.

[8] https://github.com/allenai/scitldr.

[9] We release our models and their outputs, so other metrics can be readily used as well in the future.

[10] https://pypi.org/project/rouge-score/.

Results. Table 4.3 presents our results. TWIST decoding substantially outperforms the reranking baseline when f is the AIC model (e.g., +0.5 ROUGE-L points), but they yield (almost) the same performance when f is the Abst. model. Nonetheless, TWIST decoding achieves the best performance out of all configurations. Our small improvements might be attributed to the fact that the input to the Abst. model is a strict subset of the AIC model and there are only limited benefits from combining them.

SciTLDR Summ. Test			ROUGE		
Method	f	g	R-1	R-2	R-L
Isolation	Abst.	–	39.9	21.1	34.5
Isolation	AIC	–	40.2	21.3	34.9
Rerank	Abst.	AIC	40.5	21.7	35.1
Twist	Abst.	AIC	40.5	21.7	35.0
Δ (**Twist** − Rerank)			0.0	0.0	−0.1
Rerank	AIC	Abst.	40.1	21.2	34.8
Twist	AIC	Abst.	**40.7**	**22.1**	**35.3**
Δ (**Twist** − Rerank)			+0.6	+0.9	+0.5

Table 4.3: Combination of scientific paper summarization models. Both models are BART-based models from prior work (Cachola et al., 2020) with different input: abstract only (Abst.) or abstract, introduction, and conclusion (AIC). The best scores are in **bold**. The ROUGE scores (R-1, R-2, and R-L) are computed by averaging instance-level scores from the Python rouge-score implementation.

4.3.4 Low-Resource Scenarios

In our experiments over four diverse domains (§4.3.1), we assumed that plenty of parallel data is available in every domain, and the domain model generally outperformed the generic model. Concretely, we used 1.1M and 720K training sentence pairs for the medical and legal domains, based on the data splits from previous work (Koehn and Knowles, 2017; Hu et al., 2019). In real-world applications, however, these domain-specific translation data are often scarce since they need to be annotated by bilingual speakers with expertise in those domains. The question arises: *can a domain model trained on small parallel data still help the generic model by its complementary strengths?* To simulate such low-resource scenarios, we randomly sample {10k, 20k, 40k, 80k} sentence pairs and conduct the same evaluations with the generic and domain models as f and g, respectively.

Fig. 4.3 plots COMET scores of various decoding methods on the medical and legal domains.

The score from the generic model is constant because we only change the domain training data. There is a striking trend: even though the domain model performs poorly by itself, it improves the generic model through TWIST decoding over varying sizes. Reranking also helps the generic model as the data size increases, but the improvement is less pronounced than that of TWIST decoding.

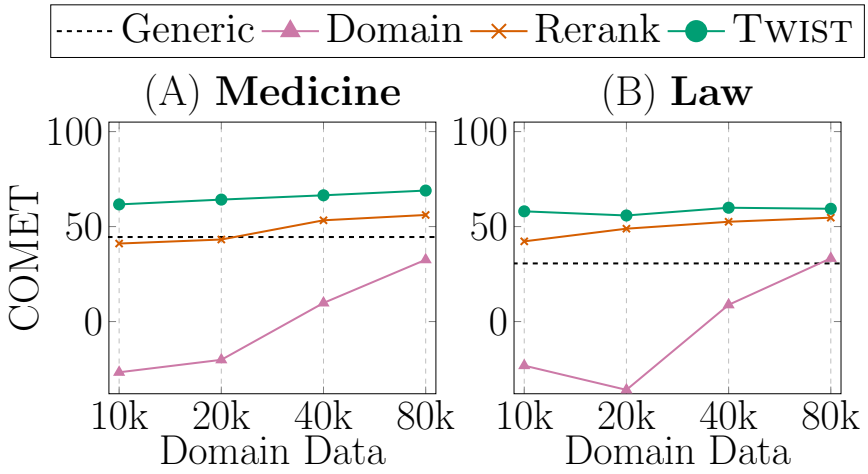


Figure 4.3: Results when parallel data are scarce in the target domain. Both TWIST decoding and reranking use the generic model as f and the domain model as g. COMET (Rei et al., 2020a) is a regression-based metric that can take negative values. λs are tuned in each case, and we found that as the domain model (g) gets stronger, λ_g increases, relative to λ_f. This observation is aligned with the intuition that λ_g indicates the relative importance of g's guidance.

4.4 Analysis

Iterations. So far, we have only applied one iteration of TWIST decoding, but Fig. 4.4 plots performance over multiple iterations. Iteration 0 signifies f's initial decoding ($\mathbf{y}^{(0)}$ in Fig. 4.1), and every iteration involves g's decoding with f's guidance ($\mathbf{z}^{(t)}$) and its reverse ($\mathbf{y}^{(t)}$). We observe that the first iteration brings most of the performance gains. This makes TWIST decoding practically appealing, as it improves performance without much increase in the computation or inference time (see below).

Inference Time. Table 4.4 reports the runtime of each decoding method, relative to f's decoding in isolation. We use batch size 1 on the same single A100-SXM GPU and measure the wall-clock time from when all models are loaded until all outputs are obtained. As expected, TWIST

decoding results in a slowdown compared to decoding in isolation, but the increase in time is only 50%. The inference time for Twist decoding is much shorter than the sum of f and g in isolation (1.4× vs. 2.1× on medical translation) because 1) the encoder computation for f and g can be parallelized and 2) the encoder computation for f is done only once while we need two runs of f's decoder. We leave it to future work to further speed up Twist decoding; since the slowdown of Twist decoding primarily comes from the decoder, it can be sped up by best-first beam search (Meister et al., 2020b), a deep-encoder, shallow-decoder strategy (Kasai et al., 2021a), or a fast, linear-complexity variant of the transformer decoder (Peng et al., 2021; Kasai et al., 2021b) that is shown to retain the performance of the standard encoder-decoder transformer. Another approach could be sequence-level knowledge distillation (Kim and Rush, 2016), which has proven successful in speeding up an ensemble translation model (Freitag et al., 2017).

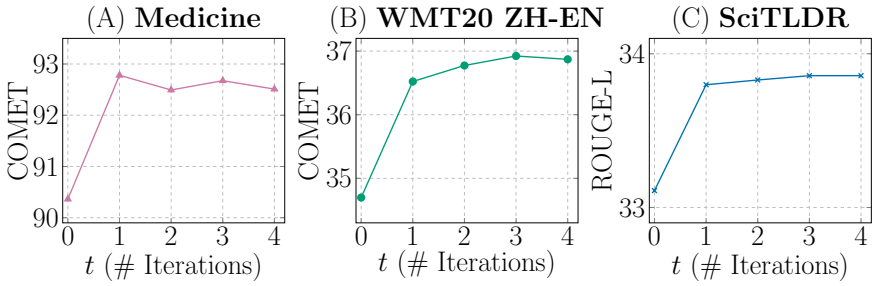


Figure 4.4: Effects of iterations on dev. performance. Iteration 0 refers to the initial decoding from f. Every iteration consists of g's decoding with f's guidance followed by f's decoding with g's guidance. The values of λs are kept the same over all iterations for simplicity. Initially, we explored gradually increasing the λs as f and g's outputs become closer, but we found no substantial performance gain.

Inference	**Medicine**			**WMT20 ZH→EN**		
Method	f	g	Time	f	g	Time
Isolation	Domain	–	1.0×	R2L	–	1.0×
Isolation	Generic	–	1.1×	L2R	–	1.0×
Rerank	Domain	Generic	1.0×	R2L	L2R	1.0×
Twist	Domain	Generic	1.4×	R2L	L2R	1.5×

Table 4.4: Inference time relative to a single model decoded in isolation. It is measured on the same single Nvidia A100-SXM GPU with batch size 1. We measure the wall-clock time from when the models are loaded until the last sentence is translated on the test data.

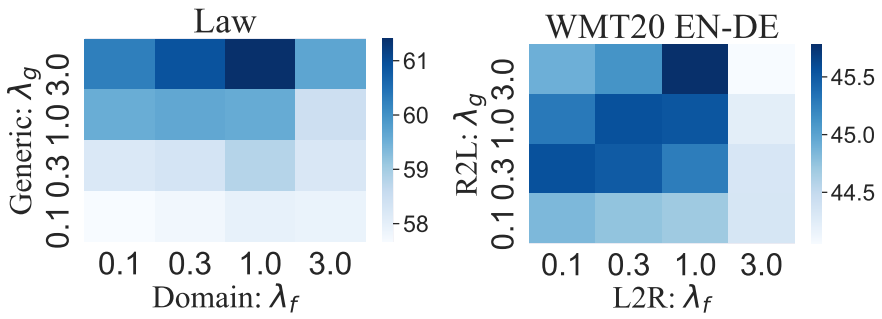


Figure 4.5: Dev. set performance measured in the COMET score (Rei et al., 2020a,b) with varying λ_f and λ_g. See Appendix §C.2 for other configurations.

Dev. Set Results	**Medicine**		**WMT20 ZH-EN**	
Distance Function	COMET	BLEU	COMET	BLEU
Original	92.8	58.1	**36.5**	**26.4**
One Candidate	**93.2**	**58.2**	35.4	25.8
Embed. Distance	92.8	57.9	**36.5**	26.2

Table 4.5: Variants of the distance function in Twist decoding. f is the domain model and g is the generic model for medical translation (German→English). f is R2L and g is L2R for WMT20 Chinese→English.

Sensitivity Analysis on Distance Coefficients. As discussed in §4.2.2, λ_f and λ_g weight the distance terms *from* f and g respectively. We tuned λ_f and λ_g on the dev. set from the range of $\{0.1, 0.3, 1.0, 3.0\}$. Fig. 4.5 visualizes how they affect the overall performance on the dev. sets. $\lambda_g > \lambda_f$ generally yields good performance, suggesting the effectiveness of the initial exploration by g with relatively weaker guidance from f.

Variants of Distance Functions. We experiment with two variants of distance terms (Table 4.5): 1) *one candidate*, which measures the distance from the 1-best candidate from the other model (vs. minimization over multiple candidates; §4.2.2) and 2) *embed. distance*, which calculates the distance based on the Euclidean distance between the embeddings. Here the embeddings are taken from the output layer of the decoder. Overall, both variants yield similar performance to the original distance function, but the *one candidate* method has a substantial performance drop on WMT20 ZH-EN. Note also that the *embed. distance* method necessitates additional distance computations between the token embeddings. This result illustrates that our original distance function is a simple yet effective design choice.

Medicine	Domain (Isolation) 👍 Generic (Isolation) 👎	Domain (Isolation) 👎 Generic (Isolation) 👍
Reference	If signs and symptoms of tardive dyskinesia appear in a patient on ABILIFY, dose reduction or discontinuation should be considered.	In placebo-controlled trials, the incidence of akathisia in bipolar patients was 12.1% with aripiprazole and 3.2% with placebo.
Domain	If signs and symptoms of tardive dyskinesia appear in one patient on ABILIFY, a dose reduction or discontinuation should be considered.	In placebo-controlled trials, the incidence of akathisia in bipolar disorder was 12.1% and 3.2% with aripiprazole.
Generic	If a patient treated with ABILIFY shows signs and symptoms of late dyskinesia, it should be considered to reduce the dose or stop treatment.	In placebo-controlled studies, the incidence of akathisia in bipolar patients was 12.1% with aripiprazole and 3.2% with placebo.
Twist f: Domain g: Generic	If signs and symptoms of tardive dyskinesia appear in one patient on ABILIFY, a dose reduction or discontinuation should be considered.	In placebo-controlled trials, the incidence of akathisia in bipolar patients was 12.1% with aripiprazole and 3.2% with placebo.

Table 4.6: Example outputs from machine translation on the medical domain. For Twist decoding, f is the domain model, and g is the generic model. In the left section, the generic model fails to capture technical terminology (late dyskinesia vs. tardive dyskinesia for the German term, *Spätdyskinesie*), and Twist decoding chooses the correct term of tardive dyskinesia from the domain model. In the right example, the domain model has a problem in coordination (12.1% and 3.2% with aripiprazole vs. 12.1% with aripiprazole and 3.2% with placebo), and Twist decoding successfully benefits from the accurate translation of the generic model.

Examples. Seen in Table 4.6 are example German→English translations from the medical domain. The left section presents a case where the domain model translates the technical term, *Spätdyskinesie*, into the corresponding English term: tardive dyskinesia. The generic model, on the other hand, generates a literal translation: late dyskinesia. In the right section, the domain model fails to handle the coordinate structure: 12.1% and 3.2% with aripiprazole vs. 12.1% with aripiprazole and 3.2% with placebo. Further, the final output has wording closer to the reference translation: trials vs. studies and bipolar patients vs. bipolar disorder. These examples illustrate that Twist decoding benefits from the complementary strengths of the domain and generic models.

4.5 Related Work

4.5.1 Decoding from Multiple Models.

Much early work proposed methods to generate text from multiple models especially for machine translation (often called *consensus-based decoding*; Bangalore et al., 2001, 2002; Matusov et al., 2006; Rosti et al., 2007; Sim et al., 2007; Hildebrand and Vogel, 2008). Most of these methods limit their search space to n-best candidates from individual translation models (Li et al., 2009), contrasting with our Twist decoding where one model can update its translation outputs under the guidance of another model. *Collaborative decoding* (Li et al., 2009) trains a separate feature-based scorer that measures the consensus between phrase-based Chinese-to-English translation models. Several recent works proposed inference algorithms for decoding from multiple generators for specific tasks, such as detoxification and abductive reasoning (West et al., 2021; Liu et al., 2021).

4.5.2 Alternatives to Left-to-Right Decoding.

We showed that Twist decoding can be used to benefit from models with diverging generation order. Several prior works proposed approaches for generating text in a different fashion than the standard left-to-right order. For example, much recent work explored non-autoregressive generation (Gu et al., 2018; Lee et al., 2018; Mansimov et al., 2019; Ghazvininejad et al., 2019; Kasai et al., 2020, *inter alia*) primarily to parallelize and speed up inference. More specifically, several works introduced training and/or inference algorithms that combine left-to-right and right-to-left models for machine translation (Zhou et al., 2019) and commonsense inference (Zaidi et al., 2020). Qin et al. (2020) incorporated right (future) context into a left-to-right language model by iterative gradient-based updates on the output representations. Those algorithms are designed specifically for the combination of left-to-right and right-to-left generation and cannot be easily extended to more general situations, such as diverging tokenization and vocabularies where Twist decoding has been shown effective.

4.6 Summary

This chapter presented Twist decoding, a general inference algorithm that generates text from diverse models without the assumption of a shared vocabulary, tokenization, or generation order. Our method enables diverse models to guide each other, thereby outperforming individual models over various scenarios, even when one of the models is much weaker because of limited data. We also demonstrated that Twist decoding can be viewed as a generalization and improvement of the commonly-adopted reranking heuristic. As it only requires a small change in code, we hope that researchers and practitioners will explore complementary strengths of diverse generation models through Twist decoding.

Part III

Transparent and Communal Evaluation Methodologies for Natural Language Processing

Chapter 5

Transparent Human Evaluation for Image Captioning

Today people talk about the rapid progress in natural language processing (NLP) and artificial intelligence (AI). But what are we measuring? Are we making progress from everyone's point of view? Evaluating NLP systems presents a serious challenge, particularly for language generation tasks such as machine translation and summarization. Language generation is inherently open-ended, and generation quality cannot be measured using a simple metric like classification accuracy or F1 score. As AI technology becomes further enmeshed in our society, it is vital to systematically collect feedback about models from human users. This chapter introduces and adopts an interpretable evaluation rubric for state-of-the-art image captioning models. Unlike evaluations conducted by previous work (e.g., evaluations without any interpretable rubric), our human-in-the-loop evaluation demonstrates that machine-generated captions continue to fall short of human-written ones.

5.1 Introduction

Recent progress in large-scale training has pushed the state of the art in vision-language tasks (Li et al., 2020b; Zhang et al., 2021, *inter alia*). One of these tasks is image captioning, whose objective is to generate a caption that describes the given image. The performance in image captioning

The material in this chapter is adapted from Kasai et al. (2022d).

Machines	P	R	CIDEr
A red fire hydrant spewing water on a street.	5	3	139.2
A red fire hydrant spraying water on a street.	5	3	205.2
Human			
A busted red fire hydrant spewing water all over a street creating a rainbow.	5	5	120.5

Figure 5.1: These machine captions are *precise* (in the scale of 1–5) but lose points in recall (i.e., coverage of salient information); they both ignore the rainbow in the picture. Automatic metrics, such as CIDEr, do not capture this failure.

has been primarily measured in automatic metrics (e.g., CIDEr, Vedantam et al., 2015; SPICE, Anderson et al., 2016) on popular benchmarks, such as MSCOCO (Lin et al., 2014) and Flickr8k (Hodosh et al., 2013). Use of these metrics is justified based on their correlation with human judgments collected in previous work (Hodosh et al., 2013; Elliott and Keller, 2014; Kilickaya et al., 2017, *inter alia*).

Continuous use of these previous human judgments, however, raises significant concerns for development of both captioning models and automatic metrics because of their lack of transparency. In previous work, annotators (crowdworkers, typically) rate image captions directly (Hodosh et al., 2013), pairwise (Vedantam et al., 2015), or along multiple dimensions such as thoroughness (Aditya et al., 2015) and truthfulness (Yatskar et al., 2014). These scoring judgments depend highly on individual annotators' discretion and understanding of the annotation scheme (Freitag et al., 2021; Clark et al., 2021), making it difficult to decompose, interpret, and validate annotations. This lack of transparency also makes it difficult to interpret evaluation results for downstream applications where some aspects are particularly important (e.g., accessibility for people with visual impairments; Gleason et al., 2019, 2020). Further, these annotations were done only on relatively old models (e.g., MSCOCO leaderboard submissions in 2015; Anderson et al., 2016). Correlations of automatic metrics with human judgments can break down

especially when model types change (Callison-Burch et al., 2006), or generation models become increasingly powerful (Ma et al., 2019a; Edunov et al., 2020). We thus develop an up-to-date, transparent human evaluation protocol to better understand how current models perform and how automatic metrics are correlated when applied to current models.

At the core of our rubrics are two main scores in a tradeoff: *precision* and *recall* (Fig. 5.1). The former measures accuracy of the information in a caption, and the latter assesses how much of the salient information in the image is covered. We then penalize a caption if we find a problem in *fluency*, *conciseness*, or *inclusive language*. Two or more authors evaluate every instance and collaborate to resolve disagreements, ensuring high quality of the annotations. We assess outputs from four strong models as well as human-generated reference captions from MSCOCO. We call our scores THUMB 1.0 (**T**ransparent **Hum**an **B**enchmark), and release them publicly.

Key Findings We made several key observations from the evaluations.

- Machine-generated captions from recent models have been claimed to achieve superhuman performance using popular automatic metrics (human performance is ranked at the 250th place in the MSCOCO leaderboard),[1] but they still show substantially lower quality than human-generated ones.

- Machines fall short of humans, especially in recall (Fig. 5.1), but most automatic metrics say the opposite. This finding is consistent with prior work that showed that machines tend to produce less diverse captions than humans (van Miltenburg et al., 2018).

- Human performance is underestimated in the current leaderboard paradigm, and there is still much room for improvement on MSCOCO captioning.

- CLIPScore and RefCLIPScore (Hessel et al., 2021), recently proposed metrics that use image features, improve correlations particularly in recall. While they fail to score human generation much higher than machine one, they capture an aspect that is less reflected in text-only metrics.

- Currently available strong captioning models generate highly fluent captions. Fluency evalu-

[1] https://competitions.codalab.org/competitions/3221#results.

ation is thus no longer crucial in ranking these models.

5.2 Evaluation Protocol

We establish a transparent evaluation protocol for English image captioning models. Our rubrics and rules are developed through discussions among all annotators (first four authors of this paper) and designed to increase the reliability of evaluation (Jonsson and Svingby, 2007)

5.2.1 Evaluation Setups and Quality Control

We used images from the test data in the standard *Karpathy split* (Karpathy and Fei-Fei, 2015) of the MSCOCO dataset (Lin et al., 2014). The dataset consists of 113K, 5K, and 5K train/dev./test everyday-scene photos sampled from Flickr. We randomly sampled 500 test images and prepared one human- and four machine-generated captions for every image (§5.2.3). We first performed developmental evaluations of 250 captions for 50 images and created rubrics. We then proceeded with the rest of the captions. For every image, captions were shuffled, and thus annotators did not know which caption corresponded to which model, thereby avoiding a potential bias from knowledge about the models. We conducted two-stage annotations: the first annotator scores all captions for given images, and the second annotator checks and modifies the scores when necessary. After the developmental phase, the κ coefficient (Cohen, 1960) was 0.86 in precision and 0.82 in recall for the rest of the evaluated captions (§5.2.2).[2] The first four authors of this paper conducted all evaluations; none of them are color blind or low vision, two are native English speakers, and one is a graduate student in linguistics. We finally ensured that at least one native speaker evaluated the fluency of every caption (§5.2.2), meaning that if a caption is annotated by the two non-native speakers, one native speaker checks the fluency in an additional round.

[2]Furthermore, we found that a third annotator did not change the scores for all 100 captions randomly sampled for meta-evaluations, confirming the sufficiently high quality of our two-stage annotations. Disagreement in ratings can also result from a certain degree of subjectivity (Misra et al., 2016).

5.2.2 THumB 1.0

Similar to the framework of the automatic SPICE metric (Anderson et al., 2016), we base our manual evaluations on two main scores: **precision** and **recall**. We also consider three types of penalty: **fluency**, **conciseness**, and **inclusive language**. The overall score is computed by averaging precision and recall and deducting penalty points.

Image	Caption	P	R	Flu.	Total
	**1-A**: Up-Down ***A dog playing with a frisbee on the ground.***	3	4	0	3.5
	1-B: VinVL-base ***A otter is laying on the sand next to two frisbees.***	5	4	0.1	4.4
	1-C: VinVL-large ***A small animal laying on a rock with three frisbees.***	4	3	0	3.5
	**2-A**: Up-Down ***A close up of a plate of broccoli.***	5	3	0	4
	2-B: Unified-VLP, VinVL-base, VinVL-large ***A plate of pasta and broccoli on a table.***	4	4	0	4
	2-C: Human ***A multi colored dish with broccoli and white twisted pasta in it.***	5	5	0.1	4.9
	**3-A**: Unified-VLP ***A little girl holding a video game controller.***	3	4	0	3.5
	3-B: VinVL-large ***A little girl is blow drying her hair on a couch.***	4	5	0	4.5
	3-C: Human ***A little girl holding a blow dryer next to her head.***	5	5	0	5
	**4-A**: Up-Down ***A black cat laying in a red suitcase.***	3	5	0	4
	4-B: Unified-VLP, VinVL-base, VinVL-large ***A black cat sitting on top of a red suitcase.***	5	5	0	5
	4-C: Human ***A large black cat laying on top of a pink piece of luggage.***	4	5	0	4.5
	**5-A**: Up-Down, Unified-VLP ***A man standing in front of a display of donuts.***	3	2	0	2.5
	5-B: VinVL-large ***A woman standing behind a counter at a donut shop.***	5	3	0	4
	5-C: Human ***Woman selling doughnuts with doughnut stock in the background.***	5	5	0.3	4.7

Table 5.1: Example evaluations of machine- and human-generated captions. None of these captions get penalties in conciseness and inclusive language. Evaluated captioning models are described in §5.2.3. All MSCOCO images are provided under a Creative Commons Attribution 4.0 License (Lin et al., 2014).

Main Scores

The two main scores are assessed in the scale of 1–5. They balance information accuracy and coverage.

Precision Precision (P) measures how precise the caption is given the image. For instance, Caption 1-B in Table 5.1 is perfectly precise, while 1-A (*dog* vs. *otter*, *one* vs. *two frisbees*) and 1-C (*three* vs. *two frisbees*) are not precise. Precision guards against hallucinations from the language model (*table* in 2-B) that are known to be common failures of image captioning models (Rohrbach et al., 2018). The score of 4 is reserved for relatively minor issues, such as attributes that are almost correct (e.g., *pink* vs. *red* in 4-C, Table 5.1) or cases where the caption does not contradict the image but is not guaranteed to be true (e.g., it is unclear whether the girl is sitting on a couch in 3-B). In addition to objects themselves, precision deals with information like properties, attributes, occasions, locations, and relations between objects (e.g., *in a red suitcase* vs. *on a red suitcase* in 4-A).

Recall Recall (R) measures how much of the salient information (e.g., objects, attributes, and relations) from the image is covered by the caption. This includes color (e.g., color of the frisbees in 1-A, 1-B, and 1-C) and guards against generic, uninformative captions that machines tend to produce (Wang et al., 2020b). For instance, an otter is a *small animal*, and thus *small animal* is *precise* (1-C); however, it is much less informative (and less natural; Ordonez et al., 2013) than saying an otter. Similarly, Caption 5-B only says a woman is standing behind a counter at a donut shop, but she is selling donuts, not buying or looking at donuts, which is salient information from the picture. We do not take a point off if missing information is already expected from the caption (e.g., a double-decker bus is typically red). We often find it useful to take a generative approach when evaluating recall: *what image does the caption lead us to imagine?* When the caption entails many potential images that substantially diverge from the given image, the recall score should be low.[3]

[3]Prior work found recall (or *specificity*) can vary across cultures or languages (van Miltenburg et al., 2017). We focus on the English language in this chapter.

Penalties

Fluency Fluency (Flu.) measures the quality of captions as English text regardless of the given image. Initially, we scored fluency in the scale of 1–5, similar to P and R, but we found most captions from modern neural network models were highly fluent. Thus, we instead decided to take points off from the average of P and R if there's a fluency problem to account for minor issues that are much less problematic than losing one P/R point. The four annotators had extensive discussions and developed rubrics for fluency. Similar to recent work on professional evaluations for machine translation (Freitag et al., 2021), we evaluated under the following principle: if a fluency problem is expected to be easily corrected by a text postprocessing algorithm (e.g., grammatical error correction: Yuan and Briscoe, 2016; Sakaguchi et al., 2017), the penalty should be 0.1. This includes obvious misspellings or grammatical errors (e.g., *A otter* in 1-B) and missing determiners/hyphens (*multi colored* in 2-C). 0.5+ points were subtracted for more severe problems, such as duplication (e.g., *A display case of donuts and doughnuts*), ambiguity (e.g., *A cat is on a table with a cloth on* ***it***), and broken sentences (e.g., *A large concrete sign small buildings behind it.*). See Table D.1 in §D.1 for more extensive fluency rubrics. Note that the average fluency penalty was 0.01; this confirms that fluency is no longer crucial in ranking models for MSCOCO captioning and contrasts with human evaluations previously done for older captioning models.

Conciseness The scores so far do not take into account conciseness of captions. Specifically, a model could simply increase all scores by describing every detail in a picture. For instance, the following caption is overly repetitive: *a woman lying on her back with knees bent on a beach towel under a multicolored, striped beach umbrella, surrounded by sand, and with clear blue sky above*. We subtract 0.5 points for these captions. Note that most machine captions were short, and this penalty was only applied to two human-generated captions. It might become more crucial for future models with a more powerful object detection module that catches many objects in the picture.

Inclusive Language We found that some instances substantially diverge from inclusive language when humans are described (van Miltenburg, 2020), raising a concern for downstream applications. In these cases, we added a penalty: 0.5 points were deducted for a subjective

comment about appearance (e.g., *very pretty girl*), and 2 points for more severe problems (e.g., *beautiful breasts*).

Rules of THumB

In our development phase, we established the following additional rules to clarify our annotation scheme.

Avoiding Double Penalties When an error is accounted for in precision, we correct the error before scoring the recall, thereby avoiding penalizing the precision and recall for the same mistake. For example, P=3 is given to Caption 1-A in Table 5.1 because of its wrong detection (*dog* vs. *otter*; *one* vs. *two frisbees*), but we score the recall assuming that the caption is now *an* ***otter*** *playing with two frisbees on the ground*. This ensures that a generic, useless caption, such as *there is something on something* (P=5, R=1), would be ranked considerably lower than *a dog on the beach with two pink and yellow frisbees* (P=3, R=5). Similarly, the wrong detection in 5-A (*man* vs. *woman*) is handled only in precision. Note that such error correction is not applicable to hallucinations because there is no alignment between a part of the image and a hallucinated object (e.g., *table* in 2-B). This rule departs from the definition of recall in SPICE (Anderson et al., 2016), an automatic metric that measures the F_1 score in scene graphs predicted from reference and generated captions; their alignment is limited to WordNet synonyms (Miller, 1995). This means that classifying an otter as a dog or even a small animal would result in cascading errors both in precision and recall, overrating captions that completely overlook the otter or ones that make a more severe classification error (e.g., miscategorize the otter as a car, compared to a dog).

Object Counts as Attributes All counts are considered as object attributes, and wrong counts are handled in precision. This simplifies the distinction between precision and recall. For instance, both *a frisbee* (1-A) and *three frisbees* (1-C) are precision problems, while saying *some frisbees* would be a recall problem when it is clear that there are exactly two frisbees. Note that this is in line with SPICE, which treats object counts as attributes in a scene graph, rather than duplicating a scene graph for every instance of an object (Anderson et al., 2016).

Black and White Photo MSCOCO contains black and white or gray-scale pictures. Some captions explicitly mention that they are black and white, but we disregard this difference in our evaluations. The crowdsource instructions for creating reference captions do not specify such cases (Chen et al., 2015). Further, we can potentially run postprocessing to determine whether it is black and white to modify the caption accordingly, depending on the downstream usage.

Text Processing Image captioning models often differ slightly in text preprocessing. As a result, we found that generated captions were sometimes slightly different in format (e.g., tokenized or detokenized; lowercased or not). For better reproducibility, we follow the spirit of SacreBLEU (Post, 2018), which has become the standard package to compute BLEU scores for machine translation: all evaluations, including automatic metrics, should be done on clean, untokenized text, independently of preprocessing design choices. We apply the following minimal postprocessing to the model outputs and human captions.

- Remove unnecessary spaces at the start or end of every caption.
- Uppercase the first letter.
- Add a period at the end if it doesn't exist, and remove a space before a period if any.

We keep the postprocessing minimal and encourage future model developers to follow the standard practice in machine translation: every model has to output clean, truecased, untokenized text that is ready to be used in downstream modules. This also improves the transparency and reproducibility of automated evaluations (Post, 2018).

5.2.3 Evaluated Captions

We evaluated the following four strong models from the literature as well as human-generated captions. They share similar pipeline structure: object detection followed by crossmodal caption generation. They vary in model architecture, (pre)training data, model size, and (pre)training objective. Evaluating captions from them will enable us to better understand what has been improved and what is still left to future captioning models.

- **Up-Down** (Anderson et al., 2018) trains Faster R-CNN (Ren et al., 2015) on the Visual Genome datset (Krishna et al., 2016) for object detection. It then uses an LSTM-based crossmodal generation model.

	THumB 1.0						Automatic Metrics						
Model	P↑	R↑	Flu.↓	Con.↓	Inc.↓	Total↑	BLEU	ROUGE	BERT-S	SPICE	CIDEr	CLIP	RefCL
Human	4.82	4.35	0.019	0.02	0.00	$\mathbf{4.56}^{+0.03}_{-0.03}$	26.2	50.4	0.938	23.7	111.5	0.791	0.834
VinVL-large	4.54	3.97	0.005	0.00	0.00	$4.25^{+0.04}_{-0.04}$	33.3	56.5	0.946	26.4	141.8	0.784	0.834
VinVL-base	4.47	3.95	0.001	0.00	0.00	$4.21^{+0.04}_{-0.04}$	32.3	55.9	0.945	25.6	138.4	0.779	0.830
Unified-VLP	4.35	3.77	0.004	0.00	0.00	$4.06^{+0.04}_{-0.04}$	31.6	55.8	0.945	24.3	128.5	0.771	0.821
Up-Down	4.29	3.50	0.014	0.00	0.00	$3.88^{+0.05}_{-0.05}$	28.4	52.2	0.939	21.0	110.7	0.746	0.803

Table 5.2: Performance of image captioning models with respect to THUMB 1.0 (left) and automatic metrics (right). All scores are averaged over 500 images randomly sampled from the Karpathy test split. P: precision; R: recall; Flu.: fluency; Con.: conciseness; Inc.: inclusive language. 90% confidence intervals for total scores are calculated by bootstrapping (Koehn, 2004). All reference-based metrics take as input the same four crowdsourced captions that are not used in Human for fair comparisons. **THumB 1.0 scores Human substantially higher than the machines, unlike all automatic metrics.**

- **Unified-VLP** (Zhou et al., 2020b) uses the same object detection model as Up-Down. The transformer-based generation model is initialized with base-sized BERT (Devlin et al., 2019) and further pretrained with 3M images from Conceptual Captions (Sharma et al., 2018).
- **VinVL-base** and **VinVL-large** (Zhang et al., 2021) train a larger-scale object detection model with the ResNeXt-152 C4 architecture (Xie et al., 2017) on ImageNet (Deng et al., 2009). The transformer generation model is initialized with BERT and pretrained with 5.7M images.
- **Human** randomly selects one from the five human-generated reference captions in MSCOCO. Those captions were created by crowdworkers on Amazon Mechanical Turk (Chen et al., 2015).

Further details are described in §D.1.2 of Appendix.

5.3 Results and Analysis

We present results and analysis from our evaluations. Our transparent evaluations facilitate assessments and analysis of both captioning models (§5.3.1) and automatic metrics (§5.3.2).

5.3.1 Comparing Models

Seen in Table 5.2 (left section) is the model performance that is averaged over the 500 test images and broken down by the rubric categories. Overall, Human substantially outperforms all machines in the P, R, and total scores. In particular, we see a large gap between Human and the machines in recall (e.g., Human 4.35 vs. VinVL-large 3.97). This contrasts with the automatic

metric-based ranking of the MSCOCO leaderboard, where Human is ranked at the 250th place.[4] This result questions claims about human parity or superhuman performance on MSCOCO image captioning. The four machine captioning models are ranked in the expected order, though the small difference between VinVL-large and VinVL-base suggests that simply scaling up models would not lead to a substantial improvement. We see that the three models that are initialized with pretrained BERT (VinVL-large/base, Unified-VLP) are particularly fluent, but the problem is small in the other models as well.

While we compute representative, total scores, our transparent rubrics allow for adjusting weighting of the categories depending on the application of interest. For instance, in the social media domain, recall can be more important than precision to make captions engaging to users (Shuster et al., 2019). To assess the models independently of these aggregation decisions, we count the number of times when each model outperforms/underperforms all the others both in P and R (*strictly* best/worst, Table 5.3). We see patterns consistent with Table 5.2. For example, Human is most likely to be strictly best and least likely to be strictly worst. This suggests that machine captioning models would still fall short of crowdworkers in a wide range of downstream scenarios.

Model	Human	Vin-large	Vin-base	U-VLP	Up-Down
# Best ↑	**327**	180	161	112	74
# Worst ↓	**65**	128	150	190	269

Table 5.3: # times when each captioning model is *strictly* best/worst in the caption set (i.e., best/worst both in precision and recall).

5.3.2 Comparing Automatic Metrics

While carefully-designed human judgments like ours should be considered more reliable, automatic metrics allow for faster development cycles. Our transparent evaluations can also be used to analyze how these automatic metrics correlate with different aspects of image captioning. Table 5.2 (right section) shows automatic scores of the captioning models over 7 popular metrics for image captioning. CLIP (CLIPScore; Hessel et al., 2021) is a *referenceless* metric that uses

[4]The official leaderboard ranks submissions using CIDEr (Vedantam et al., 2015) with 40 references on the hidden test data. We use the public Karpathy test split instead, but we suspect the same pattern would hold on the hidden data as well, given the large gap between machines and Human.

image features from CLIP (Radford et al., 2021), a crossmodal retrieval model trained on 400M image-caption pairs from the web. RefCL (RefCLIPSCore) augments CLIPScore with similarities between the generated and reference captions. All other metrics, such as SPICE (Anderson et al., 2016) and CIDEr (Vedantam et al., 2015), only use reference captions without image features.

These automatic metrics generally agree with our evaluations in ranking the four machines, but completely disagree in the assessment of Human. Most metrics rank Human near the bottom, showing that they are not reliable in evaluating high-quality, human-generated captions. The two metrics with powerful image and text features (CLIPScore and RefCLIPScore) give high scores to Human compared to the other metrics, but they still fail to score Human substantially higher than VinVL-large. This suggests that automatic metrics should be regularly updated as our models become stronger (and perhaps more similar to humans), and raises a significant concern about the current practice that fixes evaluation metrics over time (Kasai et al., 2022a).

	w/o Human			w/ Human		
Metric	P	R	Total	P	R	Total
RefCLIPScore	0.34	**0.27**	**0.44**	0.31	0.26	$\mathbf{0.41}^{+0.05}_{-0.05}$
RefOnlyC	**0.42**	0.14	0.41	**0.37**	0.11	$0.34^{+0.04}_{-0.05}$
CLIPScore	0.18	**0.27**	0.32	0.17	**0.28**	$0.32^{+0.05}_{-0.05}$
CIDEr	0.27	0.18	0.33	0.21	0.11	$0.23^{+0.04}_{-0.04}$
BERT-S	0.27	0.18	0.33	0.20	0.10	$0.21^{+0.04}_{-0.04}$
SPICE	0.26	0.15	0.30	0.20	0.09	$0.21^{+0.04}_{-0.04}$
ROUGE-L	0.26	0.17	0.31	0.18	0.07	$0.18^{+0.04}_{-0.04}$
BLEU	0.21	0.13	0.25	0.15	0.04	$0.13^{+0.04}_{-0.04}$

Table 5.4: Instance-level correlations of automatic evaluation scores. RefCLIPScore and CLIPScore use image features unlike the others, and all but CLIPScore require references. All of these reference-based metrics use the same subset of four captions as in Table 5.2 that exclude Human. All metrics had correlations lower than 0.1 for fluency.

Seen in Table 5.4 are instance-level Pearson correlation scores between automatic scores and our evaluations.[5] We also add an ablation study: RefOnlyC removes image features from RefCLIPScore to quantify the effect of image features. We consider two types of scenarios: one *with* Human and one *without*. Correlations drop from the latter to the former for all metrics and aspects except CLIPScore, again showing that the metrics are not reliable in assessing human-

[5]Instance-level Pearson correlations with human judgments were often computed in prior work to compare automatic metrics for image captioning (e.g., Hessel et al., 2021). An alternative is system-level correlations, but they would be uninformative with five systems only.

generated captions. Interestingly, CLIPScore correlates best in recall (0.28 w/ Human) but suffers in precision (0.17 w/ Human). RefOnlyC, in contrast, achieves the best correlations in P at the expense of R. RefCLIPScore balances the two and achieves the best correlation in total scores. This indicates that the CLIP image features particularly help assess coverage of salient information that can be ignored in some reference captions from crowdworkers.[6] Prior work (Hessel et al., 2021) found that SPICE can still improve correlations when combined with CLIPScore, even though CLIPScore better correlates with human judgments than SPICE. This implies that image-based and reference-only metrics capture different aspects of image captioning. Our analysis indeed agrees with their finding and, further, identifies that recall is one such aspect. For an extensive description of these metrics and their configurations, see §D.1.1.

5.3.3 Score Distributions

Seen in Fig. 5.2 are distributions of precision and recall scores for human and machine-generated captions. We see that the precision distribution looks similar between Human and machines, but not recall. This provides further support for our claim that current machines fall short of humans particularly in recall.

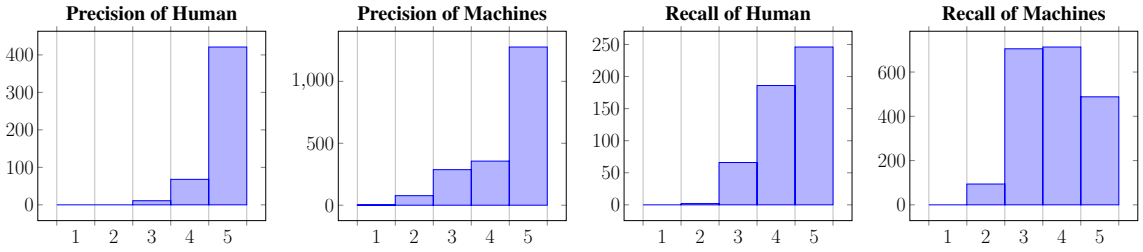


Figure 5.2: Precision/recall histograms for human- and machine-generated captions.

5.3.4 Machine vs. Human Examples

Table 5.5 provides examples that contrast machine- and human-generated captions. We see that machine-generated captions ignore salient information or make critical errors for these images.

[6]The low recall correlations of reference-only metrics can be partly because the maximum (as opposed to minimum or average) is typically taken over multiple reference captions (e.g., BERTScore, Zhang et al., 2020b). Nevertheless, this alone does not explain the recall gap from image-based metrics because RefCLIPScore also takes the maximum score over all references. Future work can explore the relation between precision/recall and different treatments of multiple references.

These problems often occur in relatively rare cases: a tennis player is showing excitement rather than hitting a ball; a bride and groom are cutting a wedding cake; a boy is wearing a tie without a shirt; a man is putting clothing and a tie on a dummy instead of a person. But these situations are exactly the most important information because of their *atypicality* (Feinglass and Yang, 2021). This illustrates fundamental problems of current image captioning models that are left to future work.

5.4 Related Work

Human Evaluations for Image Captioning Several prior works conducted human evaluations for image captioning with varying models, datasets, and annotation schemes. Much work used crowdworkers from Amazon Mechanical Turk on Flickr-based datasets, including the PASCAL (Rashtchian et al., 2010), Flickr8k/30k (Hodosh et al., 2013; Young et al., 2014), and MSCOCO datasets. Annotators scored the overall quality directly (Kulkarni et al., 2011; Hodosh et al., 2013), pairwise (Vedantam et al., 2015), or along multiple dimensions, such as truthfulness/correctness (Yatskar et al., 2014; Anderson et al., 2016), thoroughness (Aditya et al., 2015), relevance (Yang et al., 2011; Li et al., 2011), and grammaticality/readability (Mitchell et al., 2012; Elliott and Keller, 2013). There are similarities between our rubrics and previous annotations, but our framework defines every dimension in a decomposable way through discussions among all annotators, while focusing on outputs from strong models currently available. Apart from these conventional Flickr-based datasets, some other work evaluated image captions for social media (engagingness, Shuster et al., 2019; accessibility for Twitter users with vision impairments, Gleason et al., 2019, 2020) and news articles (Biten et al., 2019). Our transparent evaluations would enable us to adjust the aggregation method based on the nature of downstream applications. More specializing categories can be added for these applications in later versions (e.g., THumB 2.0).

Human Evaluations for Other Generation Tasks Much previous work explored human evaluations for other language generation tasks than image captioning. The WMT shared task (Akhbardeh et al., 2021) conducts human evaluations of state-of-the-art machine translation systems every year; participants or crowdworkers directly rate a translation in a 100-point scale,

which is a method developed by Graham et al. (2013, 2014, 2017). GENIE takes a similar approach but hosts human evaluations in leaderboards for machine translation, summarization, and commonsense reasoning (Khashabi et al., 2022). Kryscinski et al. (2019) and Fabbri et al. (2021) assessed many summarization models in a similar annotation scheme to the DUC 2006/2007 evaluations (Dang, 2006). Our transparent evaluation framework is inspired by rubric-based machine translation judgments by professional translators (Freitag et al., 2021), which resulted in different system rankings than the WMT evaluations. As top-performing models and automatic metrics are becoming increasingly similar across various natural language generation tasks, our findings on image captioning may be useful for other generation tasks as well.

5.5 Summary

This chapter introduced THUMB 1.0, transparent evaluations for the MSCOCO image captioning task. We refined our rubrics through extensive discussions among all annotators, and ensured the high quality by two-stage annotations. Our evaluations demonstrated critical limitations of current image captioning models and automatic metrics. While recent image-based metrics show promising improvements, they are still unreliable in assessing high-quality captions from crowdworkers. We hope that our annotation data will help future development of better captioning models and automatic metrics, and THUMB 1.0 will become a basis for transparent human evaluations for the image captioning task and beyond.

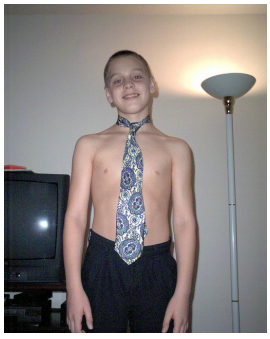
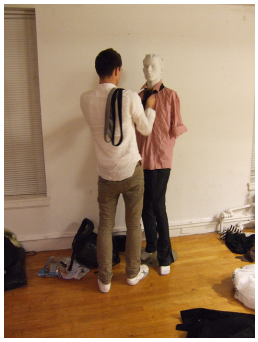
Image	Caption	P	R	Flu.	Total
	**6-A**: Up-Down ***A man holding a tennis racquet on a tennis court.***	5	3	0	4
	6-B: Unified-VLP, VinVL-base, VinVL-large ***A man holding a tennis racket on a tennis court.***	5	3	0	4
	6-C: Human ***A tennis player shows controlled excitement while a crowd watches.***	5	5	0	5
	**7-A**: Up-Down ***A person cutting a cake with a knife.***	3	3	0	3
	7-B: Unified-VLP ***A person cutting a wedding cake with a knife.***	3	5	0	4
	7-C: VinVL-base ***A couple of cakes on a table with a knife.***	5	3	0	4
	7-D: VinVL-large ***A woman cutting a cake with a knife.***	3	3	0	3
	7-E: Human ***Bride and grooms arms cutting the wedding cake with fruit on top.***	5	5	0.1	4.9
	**8-A**: Up-Down ***A young boy wearing a blue shirt and a blue tie.***	3	3	0	3
	8-B: Unified-VLP ***A young boy wearing a shirt and a tie.***	3	3	0	3
	8-C: VinVL-base ***A young boy wearing a tie standing in front of a lamp.***	5	3	0	4
	8-D: VinVL-large ***A young man wearing a tie and a shirt.***	3	3	0	3
	8-E: Human ***A man wearing only a tie standing next to a lamp.***	4	5	0	4.5
	**9-A**: Up-Down ***A couple of men standing next to each other.***	2	2	0	2
	9-B: Unified-VL ***Two men standing in a room.***	2	2	0	2
	9-C: VinVL-base ***A couple of men standing in a room.***	2	2	0	2
	9-D: VinVL-large ***Two men standing next to each other in a room.***	2	2	0	2
	9-E: Human ***A man standing next to a dummy wearing clothes.***	5	3	0	4

Table 5.5: Examples that contrast machine- and human-generated captions. All machine-generated captions overlook or misinterpret salient information: the excitement the tennis player expresses, the bride and groom cutting a wedding cake, the boy not wearing a shirt, and the man putting a tie on a dummy. None of these captions are penalized for conciseness or inclusive language. See §D.1.3 in Appendix for more examples.

Chapter 6

Bidimensional Leaderboards: Generate and Evaluate Language Hand in Hand

This chapter introduces an abstraction of leaderboards, called **bidimensional leaderboards** (BILLBOARDS), an interface that bridges modeling and evaluation research. BILLBOARDS simultaneously facilitate progress in natural language generation and its evaluation. BILLBOARDS accept two types of submissions related to a given task and dataset: generators and metrics. Unlike conventional leaderboards, BILLBOARDS do not tie model ranking to a predetermined set of metrics: generators are ranked based on the most reliable metric currently available. BILLBOARDS' built-in analysis shows that most automatic evaluations overrate machine over human-written generation, demonstrating the importance of updating metrics as generation models become stronger (and perhaps more similar to humans) in the future.

6.1 Introduction

Recent modeling advances have led to improved natural language generation in applications such as machine translation and summarization (Ng et al., 2019; Raffel et al., 2020; Brown et al., 2020, *inter alia*). This progress is typically measured with automatic scores, such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), executed by modeling researchers themselves. These metrics allow for fast, inexpensive development cycles. They were adopted based on reported

The material in this chapter is adapted from Kasai et al. (2022a).

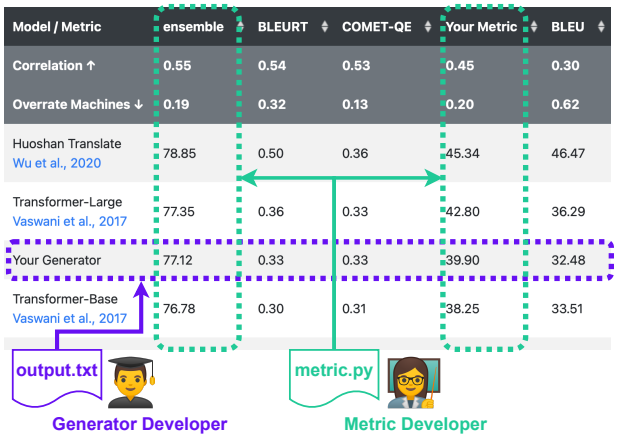


Figure 6.1: Bidimensional leaderboard (BILLBOARD). When a generator developer submits output text (`output.txt`), BILLBOARD computes all metric scores. When a metric developer submits an executable program (e.g., `metric.py`), BILLBOARD computes correlation with the human judgments, updates the ensemble metric (§6.2.2), and measures how much the metric overrates machines (§6.2.3).

correlations with human judgments at the time the metrics were introduced, but it has since been established that the correspondence can collapse when models of different types are compared (Callison-Burch et al., 2006) or models become increasingly powerful (Ma et al., 2019a; Edunov et al., 2020).

Meanwhile, many evaluation metrics that improve correlation with human judgments have been proposed (Clark et al., 2019; Zhang et al., 2020b; Sellam et al., 2020; Hessel et al., 2021, *inter alia*), but this progress has yet to be broadly adopted by the community of researchers focused on advancing models. Indeed, consistent with prior meta-evaluations (Marie et al., 2021), we found that 68% of the machine translation papers from NAACL and ACL 2021 evaluated their models solely by BLEU, and only 5% measured the performance using recent metrics with contextual representations such as COMET (Rei et al., 2020a). Similarly, automatic evaluation in 66% of the summarization papers was done only in terms of ROUGE.[1] We believe this separation between generation modeling and automatic evaluation represents a missed opportunity for each subcommunity to more rapidly benefit from the advances of the other.

We therefore propose an abstraction of conventional leaderboards, **bidimensional leaderboards** (BILLBOARDS), that simultaneously facilitates progress in natural language generation and its evaluation (Fig. 6.1). A BILLBOARD accepts two types of submissions related to a given task and

[1]We examined all papers whose title contains "machine translation" and "summarization." See Appendix E.1 for details.

dataset: **generators** and **metrics**. Unlike conventional leaderboards, model ranking is not tied to a predetermined set of metrics; the generators are ranked based on the metric that currently correlates best with human judgments. Metric submissions are ranked by their correlations to human judgments, and each is stored as an executable program, which will then be used to evaluate future generation submissions. Our BILLBOARD includes a sparse regression that selects and linearly combines three existing metrics, revealing complementary strengths. All leaderboard scores are readily reproducible, allowing research on generation models and automatic metrics to benefit from each other.

We release four BILLBOARD interfaces (https://nlp.cs.washington.edu/billboard/) spanning three generation tasks: the WMT20 EN-DE and WMT20 ZH-EN machine translation tasks (Barrault et al., 2020), the CNNDM summarization task (Hermann et al., 2015), and the MSCOCO image captioning task (Lin et al., 2014).

Key Findings Using the collective analyses of BILLBOARDS, our main findings are as follows.

- A simple linear combination of a few (diverse) metrics can sometimes improve correlation. This finding quantifies complementary effects of different metrics and encourages metric developers to seek out aspects of generated text quality not yet measured by existing metrics.

- Using linear mixed-effects models, we find that most automatic metrics, especially conventional, reference-based ones such as BLEU and ROUGE, *overrate* machines over humans in all tasks. This result provides further support for the claim that the metrics should be continually evaluated and updated as our generation models become stronger (and perhaps, closer to humans).

- When only one reference is available per instance, COMET-QE (a strong *referenceless* metric with crosslingual contextual representations; Rei et al., 2020a) achieves higher correlation with human judgments than all reference-based metrics. This raises a concern about the current standard evaluation practice in machine translation and summarization that uses reference-based metrics with a single reference per instance.

- Our findings confirm many others who report that recent metrics achieve substantially higher

correlation with human judgments than popular metrics like BLEU and ROUGE in BILLBOARDS. We believe these older metrics continue to be used mainly because modeling researchers value consistency and accessibility of evaluation practice over long periods of time. BILLBOARDS provide a way to maintain long-term comparability of system output while also drawing better conclusions about system quality, using advances in evaluation. All generators continue to be evaluated with new metrics on BILLBOARDS.

6.2 Bidimensional Leaderboards

We propose BILLBOARDS to simultaneously drive progress in natural language generation and its evaluation, which are often disconnected in current research. We first describe the general framework (§6.2.1) and the automatic analyses they provide (§6.2.2-6.2.3). We then discuss our design choices (§6.2.4) and the rubric-based, human judgment data necessary to initialize BILLBOARDS (§6.2.5).

6.2.1 Billboard Framework

The leaderboard paradigm has driven research on state-of-the-art model performance on many tasks in various fields (e.g., ImageNet, Russakovsky et al., 2015; SQuAD, Rajpurkar et al., 2016). As applications and tasks become more diverse, however, the conventional leaderboard paradigm presents a serious challenge: the assumption becomes too strong that predetermined, automatic metrics can reliably score the system performance *over time*. In particular, scores from automatic metrics often diverge from human judgments in language generation tasks, especially when models become increasingly powerful (Ma et al., 2019a).

Much recent work proposed new evaluation metrics that improve correlations with human judgments in certain generation tasks (Clark et al., 2019; Zhang et al., 2020b; Sellam et al., 2020; Hessel et al., 2021, *inter alia*), but most developers of generation models are not benefiting from them (See Appendix E.1 for our analysis of papers from NAACL/ACL 2021). From the perspective of generation model developers, it is not clear which of these many metrics in the literature is most reliable in which generation task or dataset, resulting in community-wide overuse of long-standing metrics like BLEU and ROUGE. Developers of evaluation metrics, on the other

hand, are missing the opportunity to apply their metrics to new generation models and compare them with the existing ones. We propose BILLBOARDS that bridge this gap between generation modeling and evaluation development.

Generators, Metrics, and Scores A BILLBOARD for a language generation task consists of sets of generators and evaluation metrics: $\mathcal{G} = \{G_i\}_{i=1}^{I}, \mathcal{M} = \{M_j\}_{j=1}^{J}$. Each generator G_i takes as input X_k (e.g., source text in machine translation) and generates text: $Y_{i,k} = G_i(X_k)$. A metric M_j assigns a score to each generated text given the generation input and the corresponding set of references $\mathcal{R}_k$: $s_{i,j,k} = M_j(Y_{i,k}, \mathcal{R}_k, X_k)$. The last two arguments to the function are optional; some metrics do not require references (i.e., *referenceless* or *quality estimation* metrics) or the generation input (e.g., BLEU). We then compute the aggregate score $s_{i,j}$ by averaging $s_{i,j,k}$ over K test examples.

Rankings In contrast to standard leaderboards, BILLBOARDS have a dynamic set of evaluation metrics, and generators are not ranked by a predefined metric. We first rank the metrics by measuring their correlations to human judgments as commonly done in the generation evaluation literature (Zhang et al., 2020b; Sellam et al., 2020). Let $h_{i,k}$ be a human score for $Y_{i,k}$ (i.e., output from generator G_i on input X_k). We compute the instance-level Pearson correlation for every metric M_j between $h_{i,k}$ and $s_{i,j,k}$ (M_j score for $Y_{i,k}$). All metrics are ranked by their correlations. We then use the top metric M_{j^*} to rank the generators in the descending order of s_{i,j^*}. We defer our discussions on alternative design choices (§6.2.4) and human evaluations (§6.2.5). We note, however, that the overall framework of BILLBOARDS still holds regardless of these decisions.

6.2.2 Ensemble of Metrics

So far, we have assumed that metrics are used individually in isolation, but BILLBOARDS provide a unique opportunity to examine metrics collectively. Different metrics can capture different aspects of generation quality; even if a metric is not sufficiently informative in isolation, it might reflect an important aspect of text quality that the existing metrics overlook. Here we consider a straightforward and interpretable ensemble of metrics using a regression model with ℓ_1 regular-

ization (Tibshirani, 1994). Let the ensemble's score be

$$\hat{h}_{i,k} = \sum_{j=1}^{J} w_j \cdot s_{i,j,k},$$

where w_j is a scalar coefficient associated with the jth metric and the intercept term is suppressed. We optimize the vector of coefficients $\mathbf{w}$ with the pairs of output text and a human score $\{Y_{i,k}, h_{i,k}\}_{k=1}^{K}$ from the test data:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{k=1}^{K} \left(h_{i,k} - \hat{h}_{i,k}\right)^2 + \lambda \|\mathbf{w}\|_1$$

The ℓ_1 regularization produces sparse coefficients and improves interpretability by removing highly correlated metrics. Moreover, it avoids the need for practitioners to run many metrics to obtain an ensemble score when used outside our BILLBOARDS. Our goal for the ensemble is to provide a useful signal to the research community, rather than to achieve the best possible correlation with human judges at a given time; we tune λ to get three non-zero coefficients. Every metric is standardized by its mean and standard deviation on the test data.

Similar to the individual metrics, we rank this ensemble metric by its correlation to the human judgments. To make fair comparisons, we simulate situations where the ensemble is applied to a newly submitted generator that has no human evaluations. Specifically, we perform cross validation that holds out the human judgments for each generator G_i and runs regression on the rest; we then apply these I regression models to the corresponding held-out data and calculate the overall correlation. We will see that the ensemble metric outperforms all individual metrics in some cases, suggesting that different metrics can capture different aspects.

Reproduciblity The ensemble metric is updated every time a new metric is submitted (Fig. 6.1). For reproducibility, we keep track of every past ensemble metric with a signature that indicates its coefficients, λ, and input metrics in the backend. Similar to SACREBLEU (Post, 2018), model developers can report the signature for easy replication of their scores from the ensemble metric.[2] Further, all generation outputs are saved on the leaderboards, so model developers can download outputs from all past models and compare in any way.

[2]E.g., ensemble.wmt20-zh-en+refs.AB+version.1.

6.2.3 Mixed-Effects Model Analysis

Recent work (Kasai et al., 2022d) observed that automatic metrics tend to *overrate* machine-generated text over human one on the MSCOCO image captioning task (Chen et al., 2015). This problem is particularly severe in conventional metrics that are based on n-gram overlap such as BLEU and CIDEr (Vedantam et al., 2015). This raises a significant concern about the continuous use of these conventional metrics in generation tasks as models become increasingly powerful (and more similar to humans); those metrics unintentionally discourage researchers from developing human-like, strong generation models. To quantify this undesirable property, we propose a linear mixed-effects model that compares the two groups of machine- and human-generated text. The underlying model assumes that $s_{i,j,k}$, the score from metric M_j for generator G_i and test example k, can be expressed as (the intercept term is suppressed for brevity):

$$s_{i,j,k} = \beta_0^j \mathbb{1}\{G_i \text{ is machine}\} + \beta_1^j h_{i,k} + \gamma_k + \epsilon_{i,j,k}$$

where γ_k is the random effect for example k, and $\epsilon_{i,j,k}$ is Gaussian noise. Intuitively, β_0^j measures how much metric M_j *overrates* machine generation over human one, compared against the human judgment $h_{i,k}$. $\beta_0^j = 0$ means being neutral, and indeed we will find that β_0^j is significantly positive in most cases (§6.4). We standardize all metric scores over the test samples to compare the size of β_0^j. We apply the *lme4* package (Bates et al., 2015).

6.2.4 Design Choices and Discussion

In our current setup, we make several design choices for metrics and their rankings:

- **M.1** Metrics are expected to positively correlate with the generation output quality.
- **M.2** By default, metrics are ranked based on their instance-level Pearson correlations with human judgments. We also compute and present their system-level Kendall rank correlations.
- **M.3** When available, reference-based metrics use multiple references per instance.

M.1 implies that we need to take the negative of metric scores that are intended to negatively correlate (e.g., TER, Snover et al., 2006). This normalization is also done in WMT metric competitions

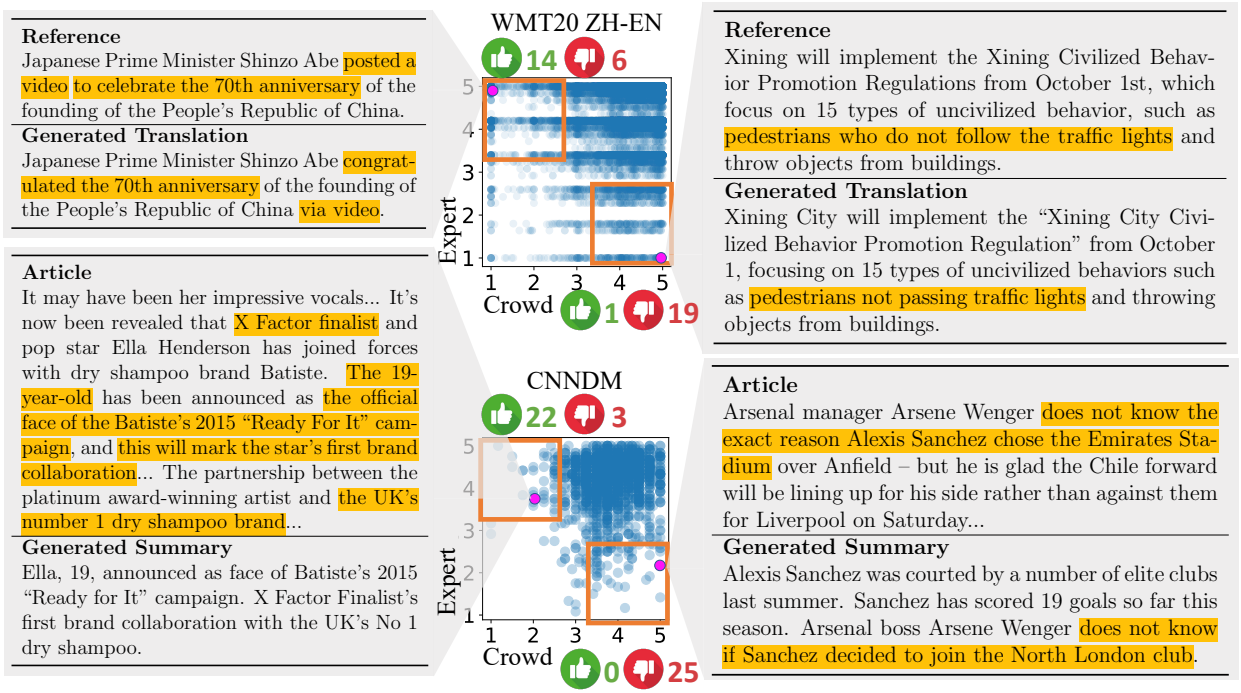


Figure 6.2: Comparisons and meta-evaluations of crowdworker and rubric-based, expert evaluations for WMT20 ZH-EN and CNNDM summarization. Every dot represents one test instance that is evaluated by the same numbers of experts and crowdworkers (one for WMT20 ZH-EN and three for CNNDM) for fair comparisons. We randomly sampled instances with diverging evaluations in two areas ☐ and conducted binary meta-evaluations (good 👍 or bad quality 👎). **Meta-evaluations agree more with the expert evaluations**: 👍 > 👎 in the upper left squares and 👎 > 👍 in the lower right squares. We suspect that the highlighted text might have caused the disagreement.

(Callison-Burch et al., 2007, 2008, *inter alia*).

While instance-level correlations are commonly used to evaluate and compare automatic metrics for various language generation tasks (Sellam et al., 2020; Fabbri et al., 2021; Hessel et al., 2021, *inter alia*), there are several alternatives to M.2. For example, Pearson, Spearman's rank, or Kendall rank correlations can be used on a system (i.e., generator) level (Callison-Burch et al., 2007; Macháček and Bojar, 2014; Mathur et al., 2020b). However, such system-level correlations would substantially reduce data points to compare automatic scores, resulting in many ties in the ranking. Spearman's and Kendall rank correlations become brittle when multiple generators are similar in overall output quality; penalizing a metric for swapping two similar generators is misleading (Macháček and Bojar, 2014). Moreover, if a metric can perform well on an instance level, it can be used to augment human judgments by, for example, flagging likely wrong ratings (Mathur et al., 2020b). Thus, we encourage researchers to develop metrics that correlate well with human judgments on an instance level. Prior work also points out other problems in

ranking metrics like *outlier effects* where outlier systems have a disproportionately large effect on the overall correlation (Mathur et al., 2020a,b). We therefore assume M.2 in the current version of BILLBOARDS, but this can be modified in a future version.

M.3 is supported by our experimental results in §6.4 that multiple references substantially improve reference-based metrics, and a single reference is often insufficient to outperform strong referenceless metrics. Some metrics have specifications for multiple references (e.g., BLEU, CIDEr). In the other cases, we evaluate outputs against every reference and take the maximum score, following prior work on image captioning evaluation (Zhang et al., 2020b; Hessel et al., 2021).[3]

6.2.5 Human Evaluation

Human evaluations are required to initialize BILLBOARDS; they are used to rank metrics, train the metric ensembling model, and assess how much each metric overrates machines. Recent work, however, points out problems when evaluations are done by crowdworkers even when extensive quality controls are performed (Gillick and Liu, 2010; Toral et al., 2018; Freitag et al., 2021; Clark et al., 2021). Freitag et al. (2021) show that rubric-based machine translation evaluations by professional translators led to substantially different generator rankings from the crowdsource evaluations in WMT 2020 (Barrault et al., 2020), where WMT participants or Amazon Mechanical Turkers directly assess each translation's adequacy by a single score (*direct assessment*). These crowdworker evaluations depend highly on individual annotators' discretion and understanding of the annotation scheme (Freitag et al., 2021; Clark et al., 2021), making it difficult to decompose, interpret, and validate (Kasai et al., 2022d). Moreover, these direct assessment scores make it difficult to interpret evaluation results for downstream applications where some aspects are particularly important (e.g., accessibility for people with visual impairments in image captioning, Gleason et al., 2020; gender bias in machine translation, Stanovsky et al., 2019).

Motivated by this line of work, we perform meta-evaluations to compare crowdsourced and rubric-based expert evaluations. Fig. 6.2 plots overall scores for test examples from WMT20 ZH-EN (Barrault et al., 2020; Freitag et al., 2021) and CNNDM summarization (Fabbri et al., 2021). Each instance is evaluated by averaging the same number of crowdworkers and expert scores for

[3]Intuitively, the maximum score measures the distance to the closest out of equally valid generations.

fair comparisons. We see that substantially many instances fall into disagreement: crowdworkers give much higher scores than experts (lower right square) or the reverse (upper left square). We sample and shuffle 20/25 examples from either type and ask a meta-evaluator to make a binary decision (good or bad quality).[4] Meta-evaluations agree more with the expert evaluations (e.g., 22 and 0 in the upper left and lower right squares for CNNDM, respectively). In the examples on the left, crowdworkers fail to properly assess a valid translation with different structure than the reference (*posted a video to celebrate* vs. *congratulated via video*) or a summary that combines information from different parts of the article. The examples on the right illustrate that crowdworkers can be fooled by inaccurate yet fluent generations (*does not know the reason* vs. *does not know if Sanchez decided*). Given this result, we decide to initialize our BILLBOARDS with rubric-based expert evaluations for all generation tasks. We still encourage future work to explore ways to improve crowdsourced evaluations for scalability.

6.3 Experiments

Having established the framework, we set up BILLBOARDS for three natural language generation tasks: machine translation, summarization, and image captioning. To maximize the performance of reference-based metrics, we use as many references as possible for each task. See §6.4 for an analysis on the effect of varying numbers of references.

6.3.1 Tasks

Machine Translation We experiment with two language pairs from the WMT 2020 news translation task (Barrault et al., 2020): Chinese→English (**WMT20 ZH-EN**) and English→German (**WMT20 EN-DE**). We use outputs from all submitted translation systems.[5] These two language pairs have expert, rubric-based scores (MQM) from Freitag et al. (2021) for a subset of 10 submitted systems, including the top-performing systems and human translations. Each output sentence is evaluated by three professional translators. Following Freitag et al. (2021), the three scores are averaged to get an instance-level score.

[4]The meta-evaluations were done by a bilingual speaker (WMT20 ZH-EN) and the first author of this paper (CNNDM).

[5]https://www.statmt.org/wmt20/translation-task.html.

We use all human translations available as a reference set for reference-based metrics. Concretely, every test instance in WMT20 ZH-EN has two translations provided by different human translation services: Human-A and Human-B (Barrault et al., 2020). In addition to Human-A and Human-B, WMT20 EN-DE provides a translation that is created by linguists who are asked to paraphrase Human-A and Human-B as much as possible (Human-P, Freitag et al., 2020). These paraphrased translations are shown to increase correlations with human judgments by mitigating the *translationese effect* and diversifying the reference when the generation quality is measured by reference-based metrics (Freitag et al., 2020).

				Single Metrics		**Ensemble of Metrics**	
Dataset	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{M}\rvert$	Top Gen.	Top Metric	Corr.	Linear Combination	Corr.
ZH-EN	19	15	Huoshan	COMET	0.55	1.72·COMET-QE+1.48·COMET+1.21·BLEURT	**0.61**
EN-DE	17	11	Tohoku	COMET	0.49	1.19·COMET+0.36·COMET-QE+0.02·Prism-ref	**0.51**
CNNDM	26	15	Lead-3	COMET	**0.41**	2.85·COMET+0.26·COMET-QE+0.01·BERTScore	0.29
MSCOCO	4	15	VinVL-large	RefCLIP-S	**0.45**	2.08·RefCLIP-S+1.51·RefOnlyC+0.82·CIDEr	**0.45**

Table 6.1: Summary of BILLBOARDS as of Jan. 10, 2022. Huoshan: Wu et al. (2020a); Tohoku: Kiyono et al. (2020); VinVL-large: Zhang et al. (2021); COMET, COMET-QE: Rei et al. (2020a); BLEURT: Sellam et al. (2020); Prism-ref: Thompson and Post (2020); BERTScore: Zhang et al. (2020b); RefCLIP-S: Hessel et al. (2021); RefOnlyC: Kasai et al. (2022d). COMET-QE is a *referenceless* metric. BLEURT is specifically trained to evaluate into-English translations. RefCLIP-S uses image features unlike most metrics for image captioning. RefOnlyC removes image features from RefCLIP-S and only uses reference text features from CLIP (Radford et al., 2021).

Along with all submitted generators in WMT20 ZH-EN and WMT20 EN-DE, we train three transformer baselines with the `fairseq` library (Ott et al., 2019) and place them in our BILLBOARDS: **transformer-base**, **transformer-large**, and **transformer-large-ensemble** with similar hyperparameters (e.g., 6-layer encoder and decoder) to the ones trained on the WMT16 EN-DE data in Vaswani et al. (2017).[6] These baselines allow researchers to compare their translation models without resource-intensive techniques such as backtranslation (Sennrich et al., 2016a), model ensembling, and deep encoders (Kasai et al., 2021a). These techniques are all used in top-performing systems of WMT20 (Wu et al., 2020a; Kiyono et al., 2020) but might be infeasible in many research settings. See Appendix E.2 for a list of all hyperparameters for the baselines.

[6]Data and models are available at https://github.com/jungokasai/billboard/tree/master/baselines.

Summarization We use the CNN/DailyMail corpus (**CNNDM**, Hermann et al., 2015; Nallapati et al., 2016). We use the standard train/dev./test split and 24 models from Fabbri et al. (2021). 100 test articles are annotated with 10 summaries written by humans (Kryscinski et al., 2019). For those 100 articles, rubric-based, expert evaluations for 18 generators, including human-written highlights, are provided by Fabbri et al. (2021).[7] Each output summary is evaluated by three experts along four dimensions: *coherence* (collective quality of all summary sentences), *consistency* (factual alignment with the article, penalizing for hallucinations), *fluency* (quality of the individual sentences), and *relevance* (selection of important content). An instance-level score is computed by averaging scores over all these categories and the three experts. Note that this aggregation method can be modified, depending on the downstream task of interest (Kasai et al., 2022d). All 10 human-written summaries are used as the reference set for reference-based metrics.[8]

Image Captioning We use the **MSCOCO** dataset (Lin et al., 2014) that consists of everyday-scene photos sampled from Flickr. Every image is annotated with five captions written by crowdworkers (Chen et al., 2015). We apply the standard *Karpathy split* (Karpathy and Fei-Fei, 2015). For each of 500 test images, rubric-based evaluations (THumB 1.0) are available for five systems, including one caption from a crowdworker (Kasai et al., 2022d). Similar to machine translation and summarization, we use all five crowdworker captions as a reference set for reference-based metrics.

6.3.2 Mixed-Effects Models

Our mixed-effects model analyzes how much every automatic metric overrates machines over humans (§6.2.3). This means that we need to free up one human generation per instance to measure its scores in the reference-based metrics. For machine translation, we score Human-B using the reference set of Human-A (WMT20 ZH-EN) or Human-A and Human-P (WMT20 EN-DE). For CNNDM, we use concatenated highlights as human-generated summaries and use the

[7] Some of the outputs are lowercased and/or tokenized. In these cases, we apply the NLTK detokenizer (Bird et al., 2009) and/or the Stanford CoreNLP truecaser (Manning et al., 2014). We encourage, however, future model developers to provide clean, untokenized output to improve the reproducibility and transparency of evaluation results (Post, 2018; Kasai et al., 2022d).

[8] Prior work used a concatenation of author-written highlights as a reference, but here we do not add it to the reference set. This is because these highlights are sometimes noisy (e.g., containing URLs) or lack coherence (Fabbri et al., 2021).

10 human-written summaries from Kryscinski et al. (2019) as the reference. We follow Kasai et al. (2022d) for MSCOCO and score their randomly-selected *Human* caption using the other four as the reference. As the distinction between the *reference* and *human generation* (e.g., Human-A vs. Human B on WMT20 ZH-EN) is arbitrary, we found that swapping the roles would still lead to similar results (See Appendix E.5).

6.4 Results and Analysis

Here we discuss the current results and make several key observations about the state of language generation evaluation. Table 6.1 summarizes the four BILLBOARDS. It is particularly noteworthy that COMET, a metric designed for machine translation, achieves the best correlation on the CNNDM summarization task as well. COMET evaluates the similarity between the crosslingual representations from XLM-RoBERTa (Conneau et al., 2020) for input text and its translation candidate. But these crosslingual representations can, of course, be used *monolingually* for English summarization. This illustrates an additional benefit of BILLBOARDS that centralize different generation tasks and find surprising task transferability of learning-based metrics. See Appendices E.2 and E.3 for lists of all participating generators and metrics.

Ensemble Metric The rightmost section of Table 6.1 shows the chosen metrics and their coefficients in the ensemble (§6.2.2). On the machine translation tasks, the ensemble metric outperforms the top individual metric.[9] In particular, we see a substantial gain of 0.06 points in WMT20 ZH-EN. The *referenceless* metric of COMET-QE is selected both for WMT20 ZH-EN and WMT20 EN-DE, suggesting complementary effects of diverse metrics. To further test this hypothesis, we perform ablations that drop one out of the three metrics at a time (Table 6.2). We see that only dropping COMET-QE would result in a decrease in the correlation score. This implies that the referenceless metric provides important information that the others do not.

[9]We found a major reason for the anomaly in CNNDM; an outlier generator that does not use the standard CNNDM training data (the GPT-2 zero-shot model; Ziegler et al., 2019) has a disproportionately large effect on the regression models. The ensemble metric outperformed the top individual metric of COMET when the zero-shot model was removed.

Removed Metric	–	COMET	COMET-QE	BLEURT
Correlation	0.61	0.61	0.57	0.61

Table 6.2: Ensemble ablation studies on WMT20 ZH-EN. Only removing COMET-QE leads to a correlation drop. See Appendix E.4 for the other datasets.

Mixed-Effects Models Seen in Table 6.3 are the results from our analysis that measures how much metrics *overrate* machines over humans (§6.2.3). We see that the fixed-effect coefficient β_0 is significantly positive in most cases. Referenceless metrics tend to have smaller coefficients. This can be due to the more diverse nature of human text than machine-generated text; reference-based metrics give a low score to human text that differs from the references even if it is of high quality. The conventional n-gram overlap-based metrics (BLEU, ROUGE, and CIDEr) have particularly large coefficients. These results suggest that the evaluation practice should be regularly updated as our generation models become stronger (and perhaps, more similar to human generation) in the future. Note that unlike the other tasks, "human-generated text" for CNNDM summarization is an automatic concatenation of author highlights, which contains substantial noise (Fabbri et al., 2021). This might explain the neutral and negative coefficients.

ZH-EN	COMET	COMET-QE	BLEURT	BLEU
	$0.27_{\pm 0.02}$	$0.13_{\pm 0.01}$	$0.32_{\pm 0.02}$	$0.62_{\pm 0.02}$
EN-DE	COMET	COMET-QE	Prism-ref	BLEU
	$0.08_{\pm 0.03}$	$-0.17_{\pm 0.02}$	$0.44_{\pm 0.02}$	$0.33_{\pm 0.03}$
CNNDM	COMET	COMET-QE	BERTScore	ROUGE-L
	$-0.17_{\pm 0.12}$	$0.02_{\pm 0.11}$	$-0.04_{\pm 0.12}$	$0.33_{\pm 0.13}$
COCO	RefCLIP-S	RefOnlyC	CIDEr	CLIP-S
	$0.09_{\pm 0.06}$	$0.24_{\pm 0.06}$	$0.43_{\pm 0.06}$	$-0.04_{\pm 0.05}$

Table 6.3: β_0 fixed-effect coefficients from the linear mixed-effects models, quantifying how much automatic metrics **overrate** machines over humans, relative to human raters. $\beta_0 = 0$ is neutral, and statistical significance is indicated by red (positive) or blue text (negative). The subscripts indicate 90% confidence intervals. Three metrics that correlate best with the human judgments are shown as well as one popular metric. COMET-QE and CLIP-S are *referenceless*. See §E.5 for the other metrics.

Effects of the Number of References Fig. 6.3 plots correlations over varying numbers of references. COMET was the top-performing *reference-based* metric regardless of the number of refer-

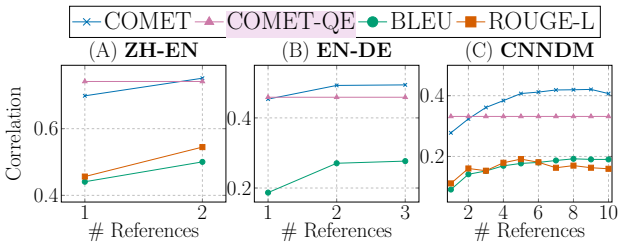


Figure 6.3: Correlations with varying numbers of references. In all cases, one reference is not sufficient to outperform the referenceless COMET-QE metric. The default ROUGE assumes English input.

ences, but we observe that it underperforms the refererenceless metric when only one reference is given. Model performance in machine translation and summarization is commonly measured by applying reference-based metrics against one reference per instance in the research community. Our finding thus raises a further concern about the current evaluation practice. Finally, we see that popular choices of BLEU and ROUGE metrics have much lower correlations than the recent metrics over various numbers of references, in line with the recent studies (Mathur et al., 2020a, *inter alia*).

6.5 Related Work

6.5.1 Related Benchmarks

WMT organizes the metric competition track in parallel with the translation task every year (Mathur et al., 2020b; Barrault et al., 2020, *inter alia*). Participants submit automatic scores for the translation outputs from the parallel translation task. Unfortunately, most of these new metrics are not used by subsequent machine translation work, perhaps because they are tested solely against the concurrent translation submissions and it is up to model developers to execute or even implement new metrics. The GEM workshop (Gehrmann et al., 2021) conducts extensive analysis of models and evaluation methods over a wide set of generation tasks. Billboards ease the burden through *standard* leaderboard experience where generator developers only need to upload generation outputs for the test split. Billboards also offer automatic ensembling of metrics and quantify the diversity that a new metric adds. The human-in-the-loop Genie leaderboard (Khashabi et al., 2022) centralizes crowdsourced evaluations for generation tasks. The current Billboard setup is based on rubric-based, expert evaluation data from previous

work, but future work can explore ways to improve crowdsourced evaluations and use them to update BILLBOARDS.

6.5.2 From Bidimensional to Multidimensional

BILLBOARDS lend themselves to a natural extension: *multidimensional leaderboards*. In particular, generation models have more aspects than generation quality, such as training and inference efficiency, sample efficiency, and robustness. These aspects are often ignored in the current leaderboard paradigm but are important to better serving practitioners' needs (Schwartz et al., 2019; Ethayarajh and Jurafsky, 2020; Mishra and Arunkumar, 2021). There are ongoing modeling and benchmarking efforts especially for efficient machine translation (Heafield et al., 2020; Peng et al., 2021; Kasai et al., 2021b, *inter alia*). We leave this extension to future work and specifically target the gap between generation modeling and evaluation.

6.6 Summary

We introduced BILLBOARDS, a simple yet powerful generalization of leaderboards that bridges the gap between generation modeling and evaluation research. We established and released four BILLBOARDS on machine translation, summarization, and image captioning tasks. We demonstrated that their built-in analysis of metric ensembling and mixed-effects modeling revealed key insights into the current state of natural language generation and its evaluation methods. BILLBOARDS allow for a standard leaderboard experience both on the modeling and evaluation sides. We invite submissions from researchers through our website.

Chapter 7

Conclusion

This thesis presents my key contributions towards the long-term goal of efficient, customizable, and communal natural language processing. We explored efficient methods for machine translation and transformer models, customization algorithms for language generation, and communal platforms for robust evaluation.

As shown in Part I, the simple layer allocation strategy of deep encoder, shallow decoder can be applied to any sequence-to-sequence scenario with a encoder-decoder architecture, though we focused on machine translation. Such scenarios include large-scale pretraining, such as T5 (Raffel et al., 2020) and BART (Lewis et al., 2020). T2R made further steps by target the fundamental inefficiency of transformers that scale quadratically with respect to sequence length. In Part II, we explored the idea of model customization at inference generation time. This avoided expensive additional training typically needed for customization. Finally, Part III presents two related efforts towards communal, rebust evaluation of language generation models.

7.1 Future Directions

Dynamic and real-world evaluations. The AI train has already left the station: ChatGPT is used by more than 100 million people, and Google Translate translates more than 100 billion words per day. At present, most AI and NLP models are evaluated statically and only once, when papers are written. Since AI models have proven useful in many tasks, there has never been a more opportune time for the community to explore dynamic and real-world evaluations

beyond fixed test data. My collaborators and I recently started REALTIME QA, a dynamic benchmarking effort that evaluates question answering systems in real time (Kasai et al., 2022e).[1] Such dynamic evaluations benefit from interdisciplinary collaborations; real-time question answering systems can, for instance, facilitate emergency management of natural disasters and pandemics. I am excited to contribute to this effort from NLP and machine learning perspectives. I believe that real-world evaluations will reveal crucial challenges that guide our future research in many aspects. For example: How should a real-time system combat fake news or toxic content? How can we update an NLP system quickly and efficiently to fulfill real-time information needs? I believe that insights from evaluation methodologies, efficient NLP, and language generation will add valuable insights to this research area.

Massively multilingual NLP. Current NLP data creation and model development focus heavily on the English language, leading to over-representation of English-centric problems (e.g., information needs about American politics). This lack of multilingual NLP research and resources limits the diversity and accessibility of AI technology and heightens the barriers to the use of many AI applications in the world. Indeed, English covers only one quarter of global web users.[2] The NLP community should pursue massively multilingual processing that benefits people around the world; we need to develop linguistically diverse resources and advance models that can be used for applications in many languages. Indeed, my early work showed the possibility of effectively expanding AI models to diverse languages with smaller or even no labeled training data by using our multilingual vector representation (Mulcaire et al., 2019; Mulcaire* et al., 2019). Multilingual vector representations continue to be studied in more recent work (e.g., Xue et al., 2021). I believe that it is promising to explore multilingual processing from the perspectives of inference algorithms and efficiency. Many global languages lack large knowledge sources, such as Wikipedia, but much current work assumes that user questions can be answered based solely on a knowledge source in the users' own language (Asai et al., 2021a,b, 2022). Can we find an inference algorithm that systematically combines knowledge sources from various languages and provides answers to users regardless of their language? Can we develop

[1] https://realtimeqa.github.io/.
[2] https://www.internetworldstats.com/stats7.htm.

models that efficiently process many languages? With the rich body of tools and methods currently available, the time is ripe to tackle these challenging research problems and improve the inclusiveness of language technologies for the billions of speakers of the 7,000+ languages other than English.

Bibliography

Somak Aditya, Yezhou Yang, Chitta Baral, Cornelia Fermüller, and Yiannis Aloimonos. 2015. From images to sentences through scene description graphs using commonsense reasoning and knowledge.

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proc. of EMNLP*.

Farhad Akhbardeh, Arkady Arkhangorodsky, Magdalena Biesialska, Ondřej Bojar, Rajen Chatterjee, Vishrav Chaudhary, Marta R. Costa-jussa, Cristina España-Bonet, Angela Fan, Christian Federmann, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Leonie Harter, Kenneth Heafield, Christopher Homan, Matthias Huck, Kwabena Amponsah-Kaakyire, Jungo Kasai, Daniel Khashabi, Kevin Knight, Tom Kocmi, Philipp Koehn, Nicholas Lourie, Christof Monz, Makoto Morishita, Masaaki Nagata, Ajay Nagesh, Toshiaki Nakazawa, Matteo Negri, Santanu Pal, Allahsera Auguste Tapo, Marco Turchi, Valentin Vydrin, and Marcos Zampieri. 2021. Findings of the 2021 conference on machine translation (WMT21). In *Proc. of WMT*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: semantic propositional image caption evaluation. In *Proc. of ECCV*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proc. of CVPR*.

Akari Asai, Jungo Kasai, Jonathan H Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. 2021a. XOR QA: Cross-lingual open-retrieval question answering. In *Proc. of NAACL*.

Akari Asai, Shayne Longpre, Jungo Kasai, Chia-Hsuan Lee, Rui Zhang, Junjie Hu, Ikuya Yamada, Jonathan H. Clark, and Eunsol Choi. 2022. MIA 2022 shared task: Evaluating cross-lingual open-retrieval question answering for 16 diverse languages. In *Proc. of MIA*.

Akari Asai, Xinyan Yu, Jungo Kasai, and Hannaneh Hajishirzi. 2021b. One question answering model for many languages with cross-lingual dense passage retrieval. In *Proc. of NeurIPS*.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *Proc. of ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proc. of ASRU*.

Srinivas Bangalore, Vanessa Murdock, and Giuseppe Riccardi. 2002. Bootstrapping bilingual data using consensus translation for a multilingual instant messaging system. In *Proc. of COLING*.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proc. of EMNLP*.

Antonio Valerio Miceli Barone, Jindřich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017. Deep architectures for neural machine translation. In *Proc. of WMT*.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi,

Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proc. of WMT*.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proc. of WMT*.

Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. 2015. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*.

Rachel Bawden, Giorgio Maria Di Nunzio, Cristian Grozea, Inigo Jauregi Unanue, Antonio Jimeno Yepes, Nancy Mah, David Martinez, Aurélie Névéol, Mariana Neves, Maite Oronoz, Olatz Perez-de Viñaspre, Massimo Piccardi, Roland Roller, Amy Siu, Philippe Thomas, Federica Vezzani, Maika Vicente Navarro, Dina Wiemann, and Lana Yeganova. 2020. Findings of the WMT 2020 biomedical translation shared task: Basque, Italian and Russian as new additional languages. In *Proc. of WMT*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Steven Bird, Evan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. Cambridge University Press.

Ali Furkan Biten, Lluís Gómez, Marçal Rusiñol, and Dimosthenis Karatzas. 2019. Good news, everyone! context driven entity-aware captioning for news images. In *Proc. of CVPR*.

Florian Böhm, Yang Gao, Christian M. Meyer, Ori Shapira, Ido Dagan, and Iryna Gurevych. 2019. Better rewards yield better summaries: Learning to summarise without references. In *Proc. of EMNLP*.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia

Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proc. of WMT*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proc. of WMT*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proc. of WMT*.

Léo Bouscarrat, Antoine Bonnefoy, Thomas Peel, and Cécile Pereira. 2019. STRASS: A light and effective method for extractive summarization based on sentence embeddings. In *Proc. of ACL*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proc. of NeurIPS*.

Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel Weld. 2020. TLDR: Extreme summarization of scientific documents. In *Findings of EMNLP*.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (meta-) evaluation of machine translation. In *Proc. of WMT*.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proc. of WMT*.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proc. of EACL*.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019a. KERMIT: Generative insertion-based modeling for sequences.

William Chan, Mitchell Stern, Jamie Ryan Kiros, and Jakob Uszkoreit. 2019b. An empirical study of generation order for machine translation.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. of ACL*.

Ciprian Chelba, Mia Chen, Ankur Bapna, and Noam Shazeer. 2020. Faster transformer decoding: N-gram masked self-attention.

Tanfang Chen, Weiwei Wang, Wenyang Wei, Xing Shi, Xiangang Li, Jieping Ye, and Kevin Knight. 2020. DiDi's machine translation system for WMT2020. In *Proc. of WMT*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proc. of ACL*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proc. of SSST-8*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2021. Rethinking attention with Performers. In *Proc. of ICLR*.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. In *Proc. of COLING*.

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. All that's 'human' is not gold: Evaluating human evaluation of generated text. In *Proc. of ACL*.

Elizabeth Clark, Asli Celikyilmaz, and Noah A. Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proc. of ACL*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *CL*.

Michael Collins, Brian Roark, and Murat Saraclar. 2005. Discriminative syntactic language modeling for speech recognition. In *Proc. of ACL*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proc. of ACL*.

Raj Dabre and Atsushi Fujita. 2019. Recurrent stacking of layers for compact neural machine translation models. In *Proc. of AAAI*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. of ACL*.

Hoa Trang Dang. 2006. Overview of DUC 2006. In *Proc. of DUC*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz. 2019. Universal transformers. In *Proc. of ICLR*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proc. of CVPR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Proc. of NeurIPS*.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. BanditSum: Extractive summarization as a contextual bandit. In *Proc. of EMNLP*.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proc. of NAACL*.

Sergey Edunov, Myle Ott, Marc'Aurelio Ranzato, and Michael Auli. 2020. On the evaluation of machine translation systems trained with back-translation. In *Proc. of ACL*.

Desmond Elliott and Frank Keller. 2013. Image description using visual dependency representations. In *Proc. of EMNLP*.

Desmond Elliott and Frank Keller. 2014. Comparing automatic evaluation measures for image description. In *Proc. of ACL*.

Kawin Ethayarajh and Dan Jurafsky. 2020. Utility is in the eye of the user: A critique of NLP leaderboards. In *Proc. of EMNLP*.

Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. Question answering as an automatic evaluation metric for news article summarization. In *Proc. of NAACL*.

Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating summarization evaluation. *TACL*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *Proc. of ICLR*.

Joshua Feinglass and Yezhou Yang. 2021. SMURF: SeMantic and linguistic UndeRstanding fusion for caption evaluation via typicality analysis. In *Proc. of ACL*.

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proc. of EMNLP*.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proc. of WMT*.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation.

Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *TACL*.

Markus Freitag, David Grangier, and Isaac Caswell. 2020. BLEU might be guilty but references are not innocent. In *Proc. of EMNLP*.

Kunihiko Fukushima. 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *Proc. of ICML*.

Sebastian Gehrmann, Tosin P. Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh D. Dhole, Wanyu Du, Esin Durmus, Ondrej Dusek, Chris Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Dhruv Kumar, Faisal Ladhak, Aman Madaan, Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Rubungo Andre Niyongabo, Salomey Osei, Ankur P. Parikh, Laura Perez-Beltrachini, Niranjan Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc, Thibault Sellam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabezudo, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. 2021. The GEM benchmark: Natural language generation, its evaluation and metrics. In *Proc. of GEM*.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proc. of EMNLP*.

Ulrich Germann. 2020. The University of Edinburgh's submission to the German-to-English and English-to-German tracks in the WMT 2020 news translation and zero-shot translation robustness tasks. In *Proc. of WMT*.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020a. Aligned cross entropy for non-autoregressive machine translation.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke S. Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proc. of EMNLP*.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. 2020b. Semi-autoregressive training improves mask-predict decoding.

Dan Gillick and Yang Liu. 2010. Non-expert evaluation of summarization systems is risky. In *Proc. of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Cole Gleason, Patrick Carrington, Cameron Tyler Cassidy, Meredith Ringel Morris, Kris M. Kitani, and Jeffrey P. Bigham. 2019. "It's almost like they're trying to hide it": How user-provided image descriptions have failed to make Twitter accessible. In *Proc. of WWW*.

Cole Gleason, Amy Pavel, Emma McCamey, Christina Low, Patrick Carrington, Kris M. Kitani, and Jeffrey P. Bigham. 2020. Twitter a11y: A browser extension to make twitter images accessible. In *Proc. of CHI*.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous measurement scales in human evaluation of machine translation. In *Proc. of LAW*.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2014. Is machine translation getting better over time? In *Proc. of EACL*.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2017. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*.

Yvette Graham, Barry Haddow, and Philipp Koehn. 2020. Translationese in machine translation evaluation.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *Proc. of ICLR*.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019a. Insertion-based decoding with automatically inferred generation order. *TACL*.

Jiatao Gu, Changhan Wang, and Jake Zhao. 2019b. Levenshtein transformer. In *Proc. of NeurIPS*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proc. of ACL*.

Junliang Guo, Linli Xu, and Enhong Chen. 2020. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Proc. of ACL*.

Jeremy Gwinnup and Tim Anderson. 2020. The AFRL WMT20 news translation systems. In *Proc. of WMT*.

Mark Harris. 2007. Optimizing parallel reduction in CUDA.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mengnan Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic Chinese to English news translation.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. of CVPR*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention.

Kenneth Heafield, Hiroaki Hayashi, Yusuke Oda, Ioannis Konstas, Andrew Finch, Graham Neubig, Xian Li, and Alexandra Birch. 2020. Findings of the fourth workshop on neural generation and translation. In *Proc. of WNGT*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proc. of NeurIPS*.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. 2021. CLIPScore: A reference-free evaluation metric for image captioning. In *Proc. of EMNLP*.

Almut Silja Hildebrand and Stephan Vogel. 2008. Combination of machine translation systems via hypothesis selection from combined n-best lists. In *Proc. of AMTA: Student Research Workshop*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *Proc. of NeurIPS Deep Learning and Representation Learning Workshop*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *JAIR*.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proc. of ACL*.

Junjie Hu, Mengzhou Xia, Graham Neubig, and Jaime Carbonell. 2019. Domain adaptation of neural machine translation by lexicon induction. In *Proc. of ACL*.

Kenji Imamura and Eiichiro Sumita. 2017. Ensemble and reranking: Using multiple models in the NICT-2 neural machine translation system at WAT2017. In *Proc. of WAT*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proc. of ICLR*.

Yichen Jiang and Mohit Bansal. 2018. Closed-book training to improve summarization encoder memory. In *Proc. of EMNLP*.

Anders Jonsson and Gunilla Svingby. 2007. The use of scoring rubrics: Reliability, validity, and educational consequences. *Educational Research Review*.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature*.

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *Proc. of ICML*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proc. of EMNLP*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proc. of CVPR*.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Non-autoregressive machine translation with disentangled context transformer. In *Proc. of ICML*.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. 2021a. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *Proc. of ICLR*.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. 2021b. Finetuning pretrained transformers into RNNs. In *Proc. of EMNLP*.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Lavinia Dunagan, Jacob Morrison, Alexander R. Fabbri, Yejin Choi, and Noah A. Smith. 2022a. Bidimensional leaderboards: Generate and evaluate language hand in hand. In *Proc. of NAACL*.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Hao Peng, Ximing Lu, Dragomir Radev, Yejin Choi, and Noah A. Smith. 2022b. Twist decoding: Diverse generators guide each other. In *Proc. of EMNLP*.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Dragomir Radev, Yejin Choi, and Noah A. Smith. 2022c. Beam decoding with controlled patience.

Jungo Kasai, Keisuke Sakaguchi, Lavinia Dunagan, Jacob Morrison, Ronan Le Bras, Yejin Choi, and Noah A. Smith. 2022d. Transparent human evaluation for image captioning. In *Proc. of NAACL*.

Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2022e. RealTime QA: What's the answer right now? Under review.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proc. of ICML*.

Daniel Khashabi, Gabriel Stanovsky, Jonathan Bragg, Nicholas Lourie, Jungo Kasai, Yejin Choi, Noah A. Smith, and Daniel S. Weld. 2022. GENIE: Toward reproducible and standardized human evaluation for text generation. In *Proc. of EMNLP*.

Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. 2017. Re-evaluating automatic metrics for image captioning. In *Proc. of EACL*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proc. of EMNLP*.

Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. 2019. From research to production and back: Ludicrously fast neural machine translation. In *Proc. of WNGT*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *Proc. of ICLR*.

Shun Kiyono, Takumi Ito, Ryuto Konno, Makoto Morishita, and Jun Suzuki. 2020. Tohoku-AIP-NTT at WMT 2020 news translation task. In *Proc. of WMT*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT Summit*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proc. of NGT*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalanditis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. 2016. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*.

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proc. of EMNLP*.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proc. of EMNLP*.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and

Tamara L Berg. 2011. Baby talk: Understanding and generating simple image descriptions. In *Proc. of CVPR*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proc. of ICLR*.

Jason D. Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proc. of EMNLP*.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proc. of ICML*.

Tao Lei. 2021. When attention meets fast recurrence: Training language models with reduced compute. In *Proc. of EMNLP*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proc. of ACL*.

Mu Li, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009. Collaborative decoding: Partial hypothesis re-ranking using translation consensus between decoders. In *Proc. of ACL*.

Siming Li, Girish Kulkarni, Tamara L Berg, Alexander C Berg, and Yejin Choi. 2011. Composing simple image descriptions using web-scale n-grams. In *Proc. of CoNLL*.

Xiaoya Li, Yuxian Meng, Arianna Yuan, Fei Wu, and Jiwei Li. 2020a. LAVA NAT: A non-autoregressive translation model with look-around decoding and vocabulary attention.

Xiujun Li, Xi Yin, Chunyuan Li, Xiaowei Hu, Pengchuan Zhang, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020b. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Proc. of ECCV*.

Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Hint-based training for non-autoregressive machine translation. In *Proc. of EMNLP*.

Zuchao Li, Hai Zhao, Rui Wang, Kehai Chen, Masao Utiyama, and Eiichiro Sumita. 2020c. SJTU-NICT's supervised and unsupervised neural machine translation systems for the WMT20 news translation task. In *Proc. of WMT*.

Jindrich Libovický and Jindrich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proc. of EMNLP*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proc. of Text Summarization Branches Out*.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *Proc. of ECCV*.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proc. of ACL*.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by summarizing long sequences. In *Proc. of ICLR*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proc. of EMNLP*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *TACL*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv: 1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: stochastic gradient descent with restarts. In *Proc. of ICLR*.

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. 2019a. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In *Proc. of WMT*.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard H. Hovy. 2019b. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. In *Proc. of EMNLP*.

Matouš Macháček and Ondřej Bojar. 2014. Results of the WMT14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Swetha Mandava, Szymon Migacz, and Alex Fit Florea. 2020. Pay attention when required.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. of ACL System Demonstrations*.

Elman Mansimov, Alex Wang, and Kyunghyun Cho. 2019. A generalized framework of sequence generation with application to undirected sequence models.

Benjamin Marie, Atsushi Fujita, and Raphael Rubino. 2021. Scientific credibility of machine translation research: A meta-evaluation of 769 papers. In *Proc. of ACL*.

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2020a. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proc. of ACL*.

Nitika Mathur, Johnny Wei, Markus Freitag, Qingsong Ma, and Ondřej Bojar. 2020b. Results of the WMT20 metrics shared task. In *Proc. of WMT*.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation for multiple machine translation systems using enhanced hypothesis alignment. In *Proc. of EACL*.

Clara Meister, Ryan Cotterell, and Tim Vieira. 2020a. If beam search is the answer, what was the question? In *Proc. of EMNLP*.

Clara Meister, Tim Vieira, and Ryan Cotterell. 2020b. Best-first beam search. *TACL*.

Fandong Meng, Jianhao Yan, Yijin Liu, Yuan Gao, Xianfeng Zeng, Qinsong Zeng, Peng Li, Ming Chen, Jie Zhou, Sifan Liu, and Hao Zhou. 2020. WeChat neural machine translation systems for WMT20. In *Proc. of WMT*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proc. of ICLR*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Proc. of NeurIPS*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *Proc. of ICLR*.

George A. Miller. 1995. WordNet: A lexical database for English. *CACM*.

Emiel van Miltenburg. 2020. How do image description systems describe people? a targeted assessment of system competence in the PEOPLE-domain. In *Proc. of LANTERN*.

Emiel van Miltenburg, Desmond Elliott, and Piek Vossen. 2017. Cross-linguistic differences and similarities in image descriptions. In *Proc. of INLG*.

Emiel van Miltenburg, Desmond Elliott, and Piek Vossen. 2018. Measuring the diversity of automatic image descriptions. In *Proc. of COLING*.

Swaroop Mishra and Anjana Arunkumar. 2021. How robust are model rankings : A leaderboard customization approach for equitable evaluation. In *Proc. of AAAI*.

Ishan Misra, C. Lawrence Zitnick, Margaret Mitchell, and Ross B. Girshick. 2016. Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels. In *Proc. of CVPR*.

Margaret Mitchell, Jesse Dodge, Amit Goyal, Kota Yamaguchi, Karl Stratos, Xufeng Han, Alyssa Mensch, Alex Berg, Tamara Berg, and Hal Daumé III. 2012. Midge: Generating image descriptions from computer vision detections. In *Proc. of EACL*.

Alexander Molchanov. 2020. PROMT systems for WMT 2020 shared news translation task. In *Proc. of WMT*.

Phoebe Mulcaire*, Jungo Kasai*, and Noah A. Smith. 2019. Low-resource parsing with crosslingual contextualized representations. In *Proc. of CoNLL*. * equal contribution.

Phoebe Mulcaire, Jungo Kasai, and Noah A. Smith. 2019. Polyglot contextual representations improve crosslingual transfer. In *Proc. of NAACL*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proc. of CoNLL*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proc. of EMNLP*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proc. of NAACL*.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 news translation task submission. In *Proc. of WMT*.

NVIDIA. 2014. The NVIDIA CUDA basic linear algebra subroutines (CUBLAS).

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proc. of NAACL*.

Csaba Oravecz, Katina Bontcheva, László Tihanyi, David Kolovratnik, Bhavani Bhaskar, Adrien Lardilleux, Szymon Klocek, and Andreas Eisele. 2020. eTranslation's submissions to the WMT 2020 news translation task. In *Proc. of WMT*.

Vicente Ordonez, Jia Deng, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2013. From large scale image categorization to entry-level categories. In *Proc. of ICCV*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL Demonstrations*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proc. of WMT*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *Proc. of ICML*.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proc. of NAACL*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Proc. of NeurIPS*.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021. Random feature attention. In *Proc. of ICLR*.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proc. of WMT*.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proc. of WMT*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proc. of WMT*.

Ofir Press, Noah A. Smith, and Omer Levy. 2020. Improving transformer models by reordering their sublayers. In *Proc. of ACL*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2021. Shortformer: Better language modeling using shorter inputs. In *Proc. of ACL*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proc. of EACL*.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In *Proc. of EMNLP*.

Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. 2020. Blockwise self-attention for long document understanding. In *Proc. of EMNLP*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision.

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

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *Proc. of ICLR*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JLMR*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. of EMNLP*.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2019. Guiding non-autoregressive neural machine translation decoding with reordering information.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using Amazon's Mechanical Turk. In *Proc. of NAACL Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020a. COMET: A neural framework for MT evaluation. In *Proc. of EMNLP*.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020b. Unbabel's participation in the WMT20 metrics shared task. In *Proc. of WMT*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. of NeurIPS*.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object hallucination in image captioning. In *Proc. of EMNLP*.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007. Combining outputs from multiple machine translation systems. In *Proc. of NAACL*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient content-based sparse attention with routing transformers. *TACL*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet large scale visual recognition challenge. *IJCV*.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proc. of IJCNLP*.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. 2021. Linear transformers are secretly fast weight programmers. In *Proc. of ICML*.

Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D. Manning. 2015. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proc. of VL*.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. *CACM*.

Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2019. Answers unite! unsupervised metrics for reinforced summarization models. In *Proc. of EMNLP*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. of ACL*.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proc. of ACL*.

Jean Senellart, Dakun Zhang, Bo Wang, Guillaume Klein, Jean-Pierre Ramatchandirin, Josep Crego, and Alexander Rush. 2018. OpenNMT system description for WNMT 2018: 800 words/sec on a single-core CPU. In *Proc. of WNG*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proc. of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. 2020. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *Proc. of AAAI*.

Eva Sharma, Luyang Huang, Zhe Hu, and Lu Wang. 2019. An entity-driven framework for abstractive summarization. In *Proc. of EMNLP*.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual Captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proc. of ACL*.

Libin Shen and Aravind K. Joshi. 2003. An SVM-based voting algorithm with application to parse reranking. In *Proc. of NAACL*.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *Proc. of NAACL*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-BERT: hessian based ultra low precision quantization of BERT. In *Proc. of AAAI*.

Tingxun Shi, Shiyu Zhao, Xiaopu Li, Xiaoxue Wang, Qian Zhang, Di Ai, Dawei Dang, Xue Zhengshan, and Jie Hao. 2020. OPPO's machine translation systems for WMT20. In *Proc. of WMT*.

Xing Shi and Kevin Knight. 2017. Speeding up neural machine translation decoding by shrinking run-time vocabulary. In *Proc. of ACL*.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2020. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *Proc. of AAAI*.

Kurt Shuster, Samuel Humeau, Hexiang Hu, Antoine Bordes, and Jason Weston. 2019. Engaging image captioning via personality. In *Proc. of CVPR*.

Khe Chai Sim, William J. Byrne, Mark J. F. Gales, Hichem Sahbi, and Philip C. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *Proc. of ICASSP*.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? In *Proc. of EMNLP*.

Felix Stahlberg, James Cross, and Veselin Stoyanov. 2018. Simple fusion: Return of the language model. In *Proc. of WMT*.

Gabriel Stanovsky, Noah A. Smith, and Luke Zettlemoyer. 2019. Evaluating gender bias in machine translation. In *Proc. of ACL*.

Mitchell Stern, William Chan, Jamie Ryan Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: flexible sequence generation via insertion operations. In *Proc. of ICML*.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Proc. of NeurIPS*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proc. of ACL*.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *Proc. of ACL*.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. In *Proc. of NeurIPS*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NeurIPS*.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2021. Synthesizer: Rethinking self-attention in transformer models. In *Proc. of ICML*.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020a. Sparse sinkhorn attention. In *Proc of ICML*.

Yi Tay, M. Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient Transformers: A survey.

Brian Thompson and Matt Post. 2020. Automatic machine translation evaluation in many languages via zero-shot paraphrasing. In *Proc. of EMNLP*.

Robert Tibshirani. 1994. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proc. of LREC*.

Antonio Toral, Sheila Castilho, Ke Hu, and Andy Way. 2018. Attaining the unattainable? reassessing claims of human parity in neural machine translation. In *Proc. of WMT*.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proc. of EMNLP*.

Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. 2020. ENGINE: Energy-based inference networks for non-autoregressive machine translation. In *Proc. of ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Proc. of CVPR*.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. When a good translation is wrong in context: Context-aware machine translation improves on deixis, ellipsis, and lexical cohesion. In *Proc. of ACL*.

Longyue Wang, Mu Li, Fangxu Liu, Shuming Shi, Zhaopeng Tu, Xing Wang, Shuangzhi Wu, Jiali Zeng, and Wen Zhang. 2021. Tencent translation system for the WMT21 news translation task. In *Proc. of WMT*.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. 2019a. Learning deep transformer models for machine translation. In *Proc. of ACL*.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020a. Linformer: Self-attention with linear complexity.

Weiyue Wang, Jan-Thorsten Peter, Hendrik Rosendahl, and Hermann Ney. 2016. CharacTer: Translation edit rate on character level. In *Proc. of WMT*.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019b. Non-autoregressive machine translation with auxiliary regularization. In *Proc. of AAAI*.

Zeyu Wang, Berthy Feng, Karthik Narasimhan, and Olga Russakovsky. 2020b. Towards unique and informative captioning of images. In *Proc. of ECCV*.

Daimeng Wei, Hengchao Shang, Zhanglin Wu, Zhengzhe Yu, Liangyou Li, Jiaxin Guo, Minghan Wang, Hao Yang, Lizhi Lei, Ying Qin, and Shiliang Sun. 2020. HW-TSC's participation in the WMT 2020 news translation shared task. In *Proc. of WMT*.

Peter West, Ximing Lu, Ari Holtzman, Chandra Bhagavatula, Jena D. Hwang, and Yejin Choi. 2021. Reflective decoding: Beyond unidirectional generation with off-the-shelf language models. In *Proc. of ACL*.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*.

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. 2020. HuggingFace's transformers: State-of-the-art natural language processing. In *Proc. of EMNLP: System Demonstrations*.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *Proc. of ICLR*.

Liwei Wu, Xiao Pan, Zehui Lin, Yaoming Zhu, Mingxuan Wang, and Lei Li. 2020a. The Volctrans machine translation system for WMT20. In *Proc. of WMT*.

Shuangzhi Wu, Xing Wang, Longyue Wang, Fangxu Liu, Jun Xie, Zhaopeng Tu, Shuming Shi, and Mu Li. 2020b. Tencent neural machine translation systems for the WMT20 news translation task. In *Proc. of WMT*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Proc. of AAAI*.

Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proc. of CVPR*.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. In *Proc. of EMNLP*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proc. of NAACL*.

Yezhou Yang, Ching Teo, Hal Daumé III, and Yiannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Proc. of EMNLP*.

Mark Yatskar, Michel Galley, Lucy Vanderwende, and Luke Zettlemoyer. 2014. See no evil, say no evil: Description generation from densely labeled images. In *Proc. of *SEM*.

Weiqiu You, Simeng Sun, and Mohit Iyyer. 2020. Hard-coded Gaussian attention for neural machine translation. In *Proc. of ACL*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*.

Lei Yu, Laurent Sartran, Po-Sen Huang, Wojciech Stokowiec, Domenic Donato, Srivatsan Srinivasan, Alek Andreev, Wang Ling, Sona Mokra, Agustin Dal Lago, Yotam Doron, Susannah Young, Phil Blunsom, and Chris Dyer. 2020. The DeepMind Chinese–English document translation system at WMT2020. In *Proc. of WMT*.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proc. of NAACL*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8BERT: quantized 8bit BERT. In *Proc. of EMC*^{2}.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago

Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for longer sequences. In *Proc. of NeurIPS*.

Najam Zaidi, Trevor Cohn, and Gholamreza Haffari. 2020. Decoding as dynamic programming for recurrent autoregressive models. In *Proc. of ICLR*.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018a. Accelerating neural transformer via an average attention network. In *Proc. of ACL*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020a. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proc. of ICML*.

Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. VinVL: Making visual representations matter in vision-language models. In *Proc. of CVPR 2021*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. BERTScore: Evaluating text generation with BERT. In *Proc. of ICLR*.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018b. Neural latent extractive document summarization. In *Proc. of EMNLP*.

Chunting Zhou, Graham Neubig, and Jiatao Gu. 2020a. Understanding knowledge distillation in non-autoregressive machine translation. In *Proc. of ICLR*.

Jiawei Zhou and Phillip Keung. 2020. Improving non-autoregressive neural machine translation with monolingual data. In *Proc. of ACL*.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *TACL*.

Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J. Corso, and Jianfeng Gao. 2020b. Unified vision-language pre-training for image captioning and VQA. In *Proc. of AAAI*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proc. of ACL*.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences.

Michał Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The United Nations parallel corpus v1.0. In *Proc. of LREC*.

Chapter A

Supplementary Materials for Deep Encoder, Shallow Decoder: Reevaluating Non-autoregressive Machine Translation

A.1 Results

Model			WMT14 EN−DE						WMT16 EN−RO					
	T	E-D	→DE	S_1	S_{max}	→EN	S_1	S_{max}	→RO	S_1	S_{max}	→EN	S_1	S_{max}
CMLM	4	6-6	26.7	4.4×	0.3×	30.8	4.1×	0.3×	33.0	3.6×	0.3×	33.3	3.5×	0.2×
	10	6-6	27.4	1.9×	0.2×	31.2	1.8×	0.2×	33.3	1.6×	0.1×	33.7	1.5×	0.1×
	4	12-1	24.7	**7.6×**	0.4×	29.4	**6.9×**	0.4×	31.9	**6.6×**	0.3×	32.7	5.9×	0.3×
	10	12-1	26.3	4.3×	0.2×	30.3	4.0×	0.2×	32.4	3.5×	0.1×	33.0	3.4×	0.2×
DisCo		6-6	27.4	3.6×	0.3×	31.3	3.6×	0.3×	33.2	4.0×	0.2×	33.3	4.1×	0.2×
		12-1	26.8	6.4×	0.4×	30.6	6.2×	0.4×	32.6	6.0×	0.3×	32.6	**6.3×**	0.3×
AR		6-6	**28.3**	1.0×	1.0×	**31.8**	1.0×	1.0×	**34.6**	1.0×	1.0×	34.6	1.0×	1.0×
		6-1	27.4	2.7×	**1.4×**	30.8	2.6×	**1.5×**	33.2	3.0×	**2.0×**	34.3	2.9×	**2.0×**
		12-1	**28.3**	2.5×	**1.4×**	**31.8**	2.5×	1.4×	33.8	2.9×	**2.0×**	**34.8**	2.9×	**2.0×**

Table A.1: Test BLEU and speed comparisons with varying numbers of encoder (E) and decoder (D) layers.

Table A.1 provides comparisons of speed and quality in the WMT14 EN−DE and WMT16 EN−RO datasets.

A.1.1 Hyperparameters and Setting

All of our models are implemented in `fairseq` (Ott et al., 2019) and trained with 16 Telsa V100 GPUs CUDA 10.1, and cuDNN 7.6.3. We used mixed precision and distributed training over 16 GPUs interconnected by Infiniband (Micikevicius et al., 2018; Ott et al., 2018). Apart from EN↔ZH where we used separate BPE operations, we tie all embeddings (Press and Wolf, 2017; Inan et al., 2017).

We generally follow the hyperparameters chosen in Vaswani et al. (2017); Ghazvininejad et al. (2019); Kasai et al. (2020) regardless of the numbers of encoding and decoding layers.[1] Specifically, we list the hyperparameters in Table A.2 for easy replication. All other hyperparamter options are left as default values in `fairseq`.

Hyperparameter	**Value**
label smoothing	0.1
# max tokens	4096
dropout rate	[0.1, 0.2, 0.3]
encoder embedding dim	512
encoder ffn dim	2048
# encoder attn heads	8
decoder embedding dim	512
decoder ffn dim	2048
# decoder attn heads	8
max source positions	10000
max target positions	10000
Adam lrate	5×10^{-4}
Adam β_1	0.9
Adam β_2	0.98
lr-scheduler	inverse square
warm-up lr	1×10^{-7}
# warmup updates	4000
# max updates	300K, 500K (EN→FR)
length penalty	1.0

Hyperparameter	**Value**
label smoothing	0.1
# max tokens	8192
dropout rate	[0.1, 0.2, 0.3]
encoder embedding dim	512
encoder ffn dim	2048
# encoder attn heads	8
decoder embedding dim	512
decoder ffn dim	2048
# decoder attn heads	8
max source positions	10000
max target positions	10000
Adam lrate	5×10^{-4}
Adam β_1	0.9
Adam β_2	0.999
lr-scheduler	inverse square
warm-up lr	1×10^{-7}
# warmup updates	10000
# max updates	300K, 500K (EN→FR)

Table A.2: Autoregressive (**left**) and non-autoregressive (**right**) `fairseq` hyperparameters and setting.

[1] We use their code at https://github.com/facebookresearch/Mask-Predict and https://github.com/facebookresearch/DisCo.

A.1.2 Sample Outputs

Here we provide translation outputs randomly sampled from the validation data in ZH→EN. We do not find a qualitative difference between AR 6-6 and deep-shallow 12-1 models.

Source	Reference	AR 6-6	AR Deep-Shallow 12-1
上面所列出的当然不尽完整	The previous list is not exhaustive, of course	The list above is certainly incomplete	The list above is certainly not complete
一百万人，加拿大总人口的十分之一，依靠政府的救济过活。	One million people, a tenth of the entire Canadian population, were dependent on government relief.	One million people, one tenth of Canada's population, live on government aid.	One million people, one tenth of Canada's total population, depend on government aid for their survival.
妇女企业家们还透过 关于小企业管理的一系列培训课程得到援助,这种 培训课程最初于1994年1月在安曼新营开始。	Women entrepreneurs were also assisted through a series of training courses in small-business management, first conducted in January 1994 at Amman New Camp.	Women entrepreneurs were also assisted through a series of training courses on small-scale enterprise management, which began in January 1994 at Amman New Camp.	Women entrepreneurs have also been assisted through a series of training courses on small enterprise management, which began at Amman New Camp in January 1994.
事实上,必须在中期上保持对拯救生命和控制恐怖两方面所作的 投入 ,以有效地扭转这一持续悲剧的势头。	Indeed, the investment in both saving lives and reining in terror needs to be sustained over the medium-term in order to effectively turn the tide in this continuing tragedy.	In fact, the investment made in saving lives and controlling terror must be maintained in the medium term in order to effectively reverse the momentum of this continuing tragedy.	Indeed, investment in saving lives and controlling terror must be maintained in the medium term in order to effectively reverse the momentum of this continuing tragedy.
这一复杂和动荡的世界恐怕是多元化、民主和自由的,在这世界上,美国正在力图剥夺我国作为一个主权国家的合法位置,好象两国之间两百年的关系不算回事。	In this complex and convulsed world that is supposedly pluralistic, free and democratic , the United States is trying to deny my country, Cuba, its rightful place as a sovereign nation. It is as if two centuries of relations between the two countries meant nothing.	I am afraid that this complex and volatile world is pluralistic, democratic and free, in which the United States is trying to deprive my country of its rightful place as a sovereign State, just as two hundred years of relations between the two countries are not worth it.	This complex and volatile world, which is feared, pluralistic, democratic and free, the United States is seeking to deprive our country of its rightful place as a sovereign State, as if two hundred years of relations between the two countries were not a matter.

Table A.3: Sample translation outputs from the ZH→EN validation data.

Chapter B

Supplementary Materials for Finetuning Pretrained Transformers into RNNs

B.1 Hyperparameters and Setting

All training is implemented in `fairseq` (Ott et al., 2019) and run with PyTorch 1.7.1 (Paszke et al., 2019), 8 Telsa V100 GPUs, and CUDA 11.0. We used mixed precision and distributed training over 8 GPUs (Micikevicius et al., 2018; Ott et al., 2018). Apart from EN→ZH where we used separate BPE operations and only tied the decoder input and output embeddings, we tie all embeddings (Press and Wolf, 2017; Inan et al., 2017). We experimented with feature sizes of [16, 32, 64] and [4, 8, 16, 32] for language modeling and machine translation respectively, and chose the smallest feature sizes that retained the development performance compared to the standard transformer.

Language Modeling

We generally follow the optimization method from Baevski and Auli (2019). For optimizing a model from random initialization, the learning rate is linearly warmed up from 10^{-7} to 1 for the initial 16K steps and then annealed using a cosine learning rate schedule with cycles (Loshchilov and Hutter, 2017). Each period lasts for twice the number of updates than the previous cycle, and we lower the maximum and minimum learning rates by 25% compared to the previous

cycle. The initial minimum and maximum learning rates are 10^{-5} and 1 respectively (Baevski and Auli, 2019). We train the model with a batch size of about 74K tokens with a total of 286K steps (Baevski and Auli, 2019). When we convert a pretrained transformer to an RNN model by finetuning, we found that we could speed up training by reducing the warm-up steps, total update steps, maximum and minimum rates, and batch size to 8K steps, 142K steps, $5 \cdot 10^{-6}$, 0.5, and 25K tokens without loss in validation perplexity.

Randomly Initialized Training We generally follow the hyperparameters chosen in Baevski and Auli (2019); Fan et al. (2020). Specifically, we list the hyperparameters in Table B.1 for easy replication. All other hyperparameter options are left as default values in `fairseq`.

Finetuning Pretrained Transformer Seen in Table B.2 are the hyperparameters for finetuning a pretrained transformer to RNN models. The learning rates, the max number of updates, and the learning period length are all reduced.

architecture	transformer_lm_wiki103
criterion	adaptive_loss
tokens-per-sample	512
sample-break-mode	none
# max tokens	3072
dropout rate	0.2
layer dropout rate	0.2
decoder embed dim	1024
decoder ffn dim	4096
# decoder attn heads	8
optimizer	nag
lr-scheduler	cosine
lr-period-updates	270K
lr-shrink	0.75
t-mult	2
max-lr	1
min-lr	1e-9
lr	1e-4
clip-norm	0.1
warm-up lr	1e-7
# warmup updates	16K
# max updates	286K
# GPUs	8
update-freq	3

Table B.1: Language modeling hyperparameters when randomly initialized in the `fairseq` library.

architecture	transformer_lm_wiki103
criterion	adaptive_loss
tokens-per-sample	512
sample-break-mode	none
# max tokens	3072
dropout rate	0.2
layer dropout rate	0.2
decoder embed dim	1024
decoder ffn dim	4096
# decoder attn heads	8
optimizer	nag
lr-scheduler	cosine
lr-period-updates	135K
lr-shrink	0.75
t-mult	2
max-lr	0.5
min-lr	1e-9
lr	5e-5
clip-norm	0.1
warm-up lr	1e-7
# warmup updates	8K
# max updates	142K
# GPUs	8
update-freq	1

Table B.2: Finetuning language modeling hyperparameters in the `fairseq` library. The learning rates are smaller than randomly initialized training.

Machine Translation

We experiment with 3 translation benchmarks: WMT14 EN-DE (4.5M train pairs, Bojar et al., 2016), WMT14 EN-FR (36M, Bojar et al., 2014), and WMT17 ZH-EN (20M, Bojar et al., 2017). We follow the preprocessing and data splits by previous work (EN-DE: Vaswani et al., 2017; EN-FR: Gehring et al., 2017; EN-ZH: Hassan et al., 2018; Wu et al., 2019). These datasets are all encoded into subwords by BPE (Sennrich et al., 2016b). We run joint BPE on all language pairs except EN-ZH. We use the hyperparameters of the large sized transformer (Vaswani et al., 2017): 6 layers, 16 attention heads, 1024 model dimensions, and 4096 hidden dimensions for both the encoder and decoder. We apply dropout with 0.3, weight decay with 0.01 and label smoothing with $\varepsilon = 0.1$. Following Ott et al. (2018), we use an increased batch size of approximately 460K tokens by accumulating gradients without updating parameters.

Randomly Initialized Training We generally follow the hyperparameters chosen in Vaswani et al. (2017); Ott et al. (2018). Specifically, we list the hyperparameters in Table B.3 for easy repli-

cation. All other hyperparamter options are left as default values in `fairseq`. The parameters from the last five epochs were averaged to obtain the final model.

architecture	transformer_vaswani_en_de_big
criterion	label_smoothed_cross_entropy
label smoothing	0.1
# max tokens	3584
dropout rate	0.3
weight decay	0.0
encoder embed dim	1024
encoder ffn dim	4096
# encoder attn heads	16
# encoder layers	6
decoder embed dim	1024
decoder ffn dim	4096
# decoder attn heads	16
# decoder layers	6
max source positions	1024
max target positions	1024
Adam lrate	5e-4, 3e-4 (T2R)*
Adam β_1	0.9
Adam β_2	0.98
lr-scheduler	inverse square
warm-up lr	1e-7
# warmup updates	4000
# max updates	30K, 60K (EN-FR)
length penalty	0.6
beam size	5
# GPUs	8
update-freq	16

Table B.3: Machine translation hyperparameters when randomly initialized in the `fairseq` library. *: we reduced the learning rate for T2R to avoid training divergence.

Finetuning Pretrained Transformer Seen in Table B.4 are the hyperparameters for finetuning a pretrained transformer to RNN models. The learning rate and the max number of updates are reduced. The parameters from the last five epochs were again averaged to obtain the final model.

B.2 Attention Distribution

Peakiness of Attention Fig. B.1 plots the average entropy of the T2R models with and without pretraining. Entropy is averaged across validation samples, layers, and attention heads. Comparing Figs. 3.4 and B.1, we see that there is strong correlation between validation perplexity and

architecture	transformer_vaswani_en_de_big
criterion	label_smoothed_cross_entropy
label smoothing	0.1
# max tokens	3584
dropout rate	0.3
weight decay	0.0
encoder embed dim	1024
encoder ffn dim	4096
# encoder attn heads	16
# encoder layers	6
decoder embed dim	1024
decoder ffn dim	4096
# decoder attn heads	16
# decoder layers	6
max source positions	1024
max target positions	1024
Adam lrate	2e-4
Adam β_1	0.9
Adam β_2	0.98
lr-scheduler	inverse square
warm-up lr	1e-7
# warmup updates	4000
# max updates	20K, 40K (EN-FR)
length penalty	0.6
beam size	5
# GPUs	8
update-freq	16

Table B.4: Finetuning machine translation hyperparameters. The learning rate is smaller than randomly initialized training.

entropy. The entropy decreases (and thus the attention distribution gets peakier) when a large feature size is used or the transformer pretraining is applied. This observation hints at potential future improvement of linear transformer models by introducing an inductive bias towards peaky attention distributions.

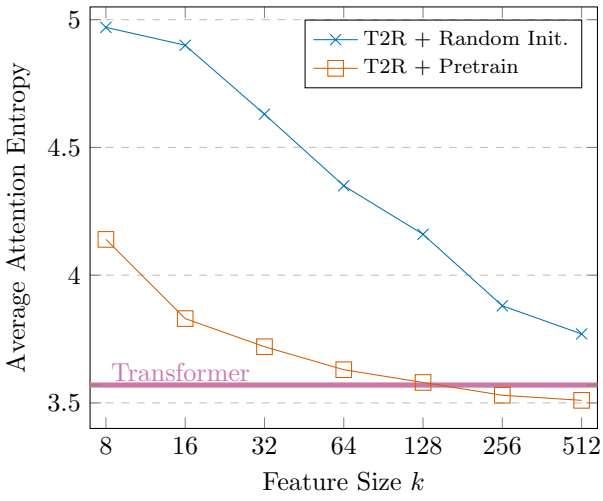


Figure B.1: Average entropy of the attention weights. They are computed on the Wikitext-103 validation data for predicting a word given the preceding 512 words.

Chapter C

Supplementary Materials for Twist Decoding: Diverse Generators Guide Each Other

C.1 Hyperparameters and Settings

We provide training and implementation details for easy replication of our experiments.

C.1.1 Domain Machine Translation

We generally follow the preprocessing and subword tokenization from Koehn and Knowles (2017); Hu et al. (2019). Table C.1 lists the hyperparameters and setting on `fairseq` that we use for all domain-specific translation models. All embeddings are shared (Press and Wolf, 2017; Inan et al., 2017). We choose the checkpoint that achieved the best loss on the validation data.

C.1.2 Left-to-Right and Right-to-Left

WMT20 ZH-EN Table C.2 lists the hyperprameters and setting on `fairseq` that we use for left-to-right and right-to-left models on the WMT20 ZH-EN dataset. We generally follow the preprocessing and tokenization from Kasai et al. (2022a). We use `newstest-2019` as the dev.

Hyperparameter	Value
label smoothing	0.1
# max tokens	8192
dropout rate	0.1
encoder embedding dim	512
encoder ffn dim	2048
# encoder attn heads	8
decoder embedding dim	512
decoder ffn dim	2048
# decoder attn heads	8
max source positions	1024
max target positions	1024
Adam lrate	5×10^{-4}
Adam β_1	0.9
Adam β_2	0.98
lr-scheduler	inverse square
warm-up lr	1×10^{-7}
# warmup updates	4000
# max updates	600K
# GPUs	8
length penalty	0.6

Table C.1: Domain translation `fairseq` hyperparameters and setting. We generally follow the base-sized configuration from Vaswani et al. (2017).

set and the official training data.[1] We apply Moses tokenization (Koehn et al., 2007) and BPE with 32K operations (Sennrich et al., 2016b) to English text. We tokenize Chinese text with the Jieba package,[2] following Hassan et al. (2018). Separately from English, BPE with 32K operations is then applied to Chinese. The decoder input and output embeddings are tied.

WMT20 EN-DE The same hyperparameters are chosen as in WMT20 ZH-EN (Table C.2). We again follow Kasai et al. (2022a) and preprocess both English and German text by the Moses tokenizer and *joint* BPE with 32K operations. All embeddings are shared.

C.1.3 SciTLDR

We use two BART-based pretrained models from Cachola et al. (2020): the abstract-only version of BART and the AIC version of CATTS$_{\text{XSUM}}$.[3] These two models are both BART-based models; CATTS$_{\text{XSUM}}$ is obtained by finetuning BART on the XSUM dataset (Narayan et al., 2018a) with

[1] http://www.statmt.org/wmt20/translation-task.html.
[2] https://github.com/fxsjy/jieba.
[3] They are both available at https://github.com/allenai/scitldr.

Hyperparameter	Value
label smoothing	0.1
# max tokens	4096
dropout rate	0.1
encoder embedding dim	1024
encoder ffn dim	4096
# encoder attn heads	16
decoder embedding dim	1024
decoder ffn dim	4096
# decoder attn heads	16
max source positions	1024
max target positions	1024
Adam lrate	5×10^{-4}
Adam β_1	0.9
Adam β_2	0.98
lr-scheduler	inverse square
warm-up lr	1×10^{-7}
# warmup updates	4000
# max updates	600K
# GPUs	8
length penalty	0.6

Table C.2: L2R and R2L translation `fairseq` hyperparameters and setting. We generally follow the large-sized configuration from Vaswani et al. (2017).

multitask scaffolding (Cachola et al., 2020).

C.1.4 λ Tuning

We tune λ_f and λ_g from $\{0.1, 0.3, 1.0, 3.0\}$, based on the dev. BLEU/ROUGE-L score on machine translation and paper summarization, respectively. Table C.3 reports the selected λ values in all scenarios.

C.2 Sensitivity Analysis on λ

Fig. C.1 presents the sensitivity analysis in the COMET score over many scenarios. Apart from a few exceptions, $\lambda_g > \lambda_f$ tends to yield good performance, suggesting the effectiveness of the initial exploration by g with relatively weaker guidance from f.

Dataset	f	g	Tuned λ λ_f	λ_g
Medicine	Domain	Generic	0.1	0.3
	Generic	Domain	0.1	3.0
Law	Domain	Generic	1.0	0.1
	Generic	Domain	0.1	3.0
Koran	Domain	Generic	1.0	3.0
	Generic	Domain	0.3	3.0
Subtitles	Domain	Generic	1.0	1.0
	Generic	Domain	1.0	1.0
WMT20 ZH-EN	L2R	R2L	1.0	3.0
	R2L	L2R	0.1	3.0
WMT20 EN-DE	L2R	R2L	0.3	0.3
	R2L	L2R	0.1	0.3
SciTLDR	Abst.	AIC	1.0	3.0
	AIC	Abst.	0.3	3.0

Table C.3: Selected λ_f and λ_g values.

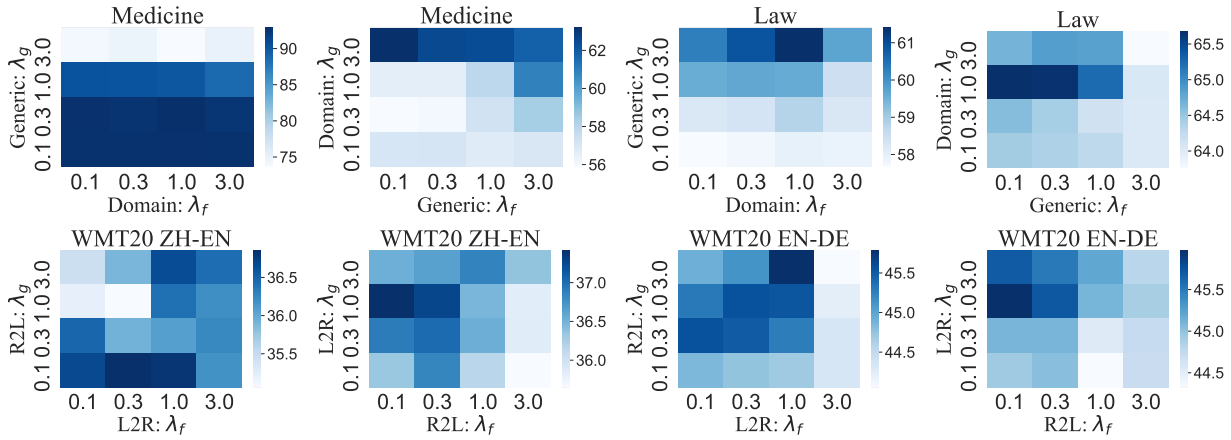


Figure C.1: Dev. set performance measured in the COMET score (Rei et al., 2020a,b) with varying λ_f and λ_g.

Chapter D

Supplementary Materials for Transparent Human Evaluation for Image Captioning

D.1 Fluency Rubrics

Table D.1 presents our fluency rubrics. They were developed by the first four authors (two of whom were native English speakers, and one was a graduate student in linguistics). Generally, if a fluency problem is expected to be easily corrected by a text postprocessing algorithm, the penalty is 0.1. More severe errors (e.g., broken sentence and ambiguity) are penalized more.

D.1.1 Automatic Metrics

Here we discuss details and configurations of the automatic metrics used in §5.3.2. CLIPScore and RefCLPScore use image features from CLIP (Radford et al., 2021), a crossmodal retrieval model trained on 400M image-caption pairs from the web. All the other five metrics only use reference captions.

BLEU BLEU (Papineni et al., 2002) is a precision-oriented metric and measures n-gram overlap between the generated and reference captions. We use the SacreBLEU implementation of BLEU-

Fluency Error Type	Penalty	Example
Obvious spelling error, one vs. two words	0.1	*cel phone, surf board*
Grammatical error that can be easily fixed	0.1	*a otter*
Casing issue	0.1	*tv, christmas*
Hyphenation	0.1	*horse drawn carriage*
Interpretable but unnatural wording	0.1	*double decked bus*
Non-trivial punctuation	0.2	*A bird standing in the wooded area with leaves all around.*
Misleading spelling error	0.5	*A good stands in the grass next to the water.* (good→goose)
Duplication	0.5	*A display case of donuts and doughnuts.*
Ambiguity	0.5	*A cat is on a table with a cloth on it.*
Awkward construction	0.1–0.5	*There is a freshly made pizza out of the oven.*
Broken sentence	0.5+	*A large concrete sign small buildings behind it.*

Table D.1: Fluency penalty rubrics.

4 and get sentence-level scores (Post, 2018).[1]

ROUGE ROUGE (Lin, 2004) measures the number of overlapping n-grams between the generated and reference captions. We use the rouge-score implementation of ROUGE-L.[2]

CIDEr CIDEr (Vedantam et al., 2015) measures the cosine similarity between the n-gram counts of the generated and reference captions with TF-IDF weighting. We use the implementation from the pycocoevalcap package.[3]

SPICE SPICE (Anderson et al., 2016) predicts scene graphs from the generated and reference captions using the Stanford scene graph parser (Schuster et al., 2015). It then measures the F_1 score between scene graphs from the generated and reference captions. WordNet Synsets are used to cluster synonyms (Miller, 1995). We again use the implementation from the pycocoevalcap package.

BERTScore BERTScore (Zhang et al., 2020b) aligns tokens between the generated and reference captions using contextual word representations from BERT (Devlin et al., 2019). We use the HuggingFace implementation (Wolf et al., 2020) and compute the F_1 score. As in Zhang et al. (2020b), we take the maximum score over all reference captions.

[1] https://github.com/mjpost/sacreBLEU/blob/v1.2.12/sacrebleu.py#L999.
[2] https://pypi.org/project/rouge-score/.
[3] https://github.com/salaniz/pycocoevalcap.

CLIPScore CLIPScore (Hessel et al., 2021) is the only *referenceless* metric out of the 7 metrics. It measures the cosine similarity between the generated caption and given image using the representations from CLIP. It is shown to correlate better with human judgments from prior work, compared to previous reference-based metrics (Hessel et al., 2021). We use the official implementation by the authors.[4]

RefCLIPScore RefCLIPScore augments CLIPScore with the maximum similarity between the generated and reference captions. We again use the official implementation.

D.1.2 Evaluated Captions

We evaluated the following four strong models from the literature as well as human-generated captions. They share similar pipeline structure but vary in model architecture, (pre)training data, model size, and (pre)training objective. Evaluating captions from them will enable us to better understand what has been improved and what is still left to future captioning models.

Up-Down The bottom-up and top-down attention model (Up-Down, Anderson et al., 2018) performs pipelined image captioning: *object detection* that finds objects and their corresponding image regions and *crossmodal generation* that predicts a caption based on the features from object detection. The bottom-up attention finds salient image regions during object detection, and the top-down one attends to these regions during crossmodal generation. Up-Down uses Faster R-CNN (Ren et al., 2015) and LSTMs (Hochreiter and Schmidhuber, 1997) for object detection and crossmodal generation respectively. Faster R-CNN is trained with the Visual Genome dataset (Krishna et al., 2016), and the crossmodal generation model is trained on the MSCOCO dataset. We generate captions for the test data with a model optimized with crossentropy.[5]

Unified-VLP Unified-VLP (Zhou et al., 2020b) also runs a pipeline of object detection and crossmodal generation. Faster R-CNN and the transformer architecture (Vaswani et al., 2017) are used for object detection and crossmodal generation respectively. Similar to Up-Down, the Faster

[4] https://github.com/jmhessel/pycocoevalcap.
[5] https://vision-explorer.allenai.org/image_captioning.

R-CNN object detection model is trained with the Visual Genome dataset. The transformer generation model, on the other hand, is initialized with base-sized BERT (Devlin et al., 2019) and pretrained on the Conceptual Captions dataset (3M images, Sharma et al., 2018) with the masked and left-to-right language modeling objectives for the captions. The crossmodal generation model is then finetuned on the MSCOCO dataset. We apply beam search of size 5 to the model with CIDEr optimization.

VinVL-base, VinVL-large VinVL with Oscar (Li et al., 2020b; Zhang et al., 2021) performs a similar pipeline of object detection, followed by crossmodal generation. The crossmodal model is initialized with BERT (Devlin et al., 2019) as in Unified-VLP but uses detected object tags to encourage alignments between image features and word representations. The object detection model with the ResNeXt-152 C4 architecture (Xie et al., 2017) is pretrained with ImageNet (Deng et al., 2009) and trained on 2.5M images from various datasets. The transformer-based crossmodal generator is initialized with BERT, pretrained with 5.7M images, and finetuned for MSCOCO captioning. We use VinVL-base and VinVL-large that are both finetuned with CIDEr optimization[6] and generate captions with beam search of size 5.

Human In addition to machine-generated captions from the four models, we assessed the quality of human-generated reference captions from MSCOCO. This will allow us to understand the performance gap between machines and humans, as well as the quality of crowdsourced captions. Human-generated captions were created using Amazon Mechanical Turk (Chen et al., 2015). Crowdworkers were only given the following instructions (Chen et al., 2015):

- Describe all the important parts of the scene.
- Do not start the sentences with "There is."
- Do not describe unimportant details.
- Do not describe things that might have happened in the future or past.
- Do not describe what a person might say.
- Do not give people proper names.
- The sentences should contain at least 8 words.

[6] https://github.com/microsoft/Oscar/blob/master/VinVL_MODEL_ZOO.md#Image-Captioning-on-COCO.

Every image has five human-generated captions, and we randomly selected one for each to evaluate. We found, however, a non-negligible number of noisy captions in the MSCCOCO dataset from annotation spammers. We often find subjective adjectives (e.g., *very nice/clean/cute*) or words that diverge from *inclusive language* in reference captions, probably because crowdworkers increased the number of words in captions effortlessly (see the last instruction item that says captions have to have 8+ words). To better estimate the performance of a human that invests reasonable effort into the captioning task, we resampled a caption for 13% of the test images, which would have been given a total score lower than 4.0.

D.1.3 Additional Machine vs. Human Examples

Table D.2 provides an additional example that contrasts machine- and human-generated captions. All machines generate generic captions and ignore the most important information that a traditional Thanksgiving dinner is being served on the table.

Image	Caption	P	R	Total
	**10-A**: Up-Down ***A table that has some food on it.***	5	2	3.5
	10-B: Unified-VLP ***A table with plates of food on a table.***	5	2	3.5
	10-C: VinVL-base ***A red table topped with plates of food and bowls of food.***	4	2	3
	10-D: VinVL-large ***A table with a turkey and other food on it.***	5	3	4
	10-E: Human ***A table set for a traditional Thanksgiving dinner.***	5	5	5

Table D.2: Additional example that contrasts machine- and human-generated captions. Similar to Table 5.5, machine-generated captions ignore the most salient information: Thanksgiving dinner. Note that this case is specific to North America; such salient information can vary across cultures or languages (van Miltenburg et al., 2017). None of these captions are penalized for fluency, conciseness, or inclusive language.

Chapter E

Supplementary Materials for Bidimensional Leaderboards: Generate and Evaluate Language Hand in Hand

E.1 Case Studies of Evaluation Practice

Fig. E.1 depicts breakdowns of evaluation metrics used in the papers on machine translation and summarization from NAACL and ACL 2021. We examined all papers whose title contains "machine translation" and "summarization." We see the clear gap between generation modeling and evaluation research; most researchers do not take advantage of recent metrics that correlate better with human judgments.

E.2 Participating Generators

Here we list the generators submitted in the initial Billboards.

E.2.1 WMT20 ZH-EN

We use all 16 submissions for the WMT20 ZH-EN task (Barrault et al., 2020)[1] as well as our own three transformer baselines that were implemented in `fairseq` (Ott et al., 2019). Our baselines

[1] https://www.statmt.org/wmt20/results.html.

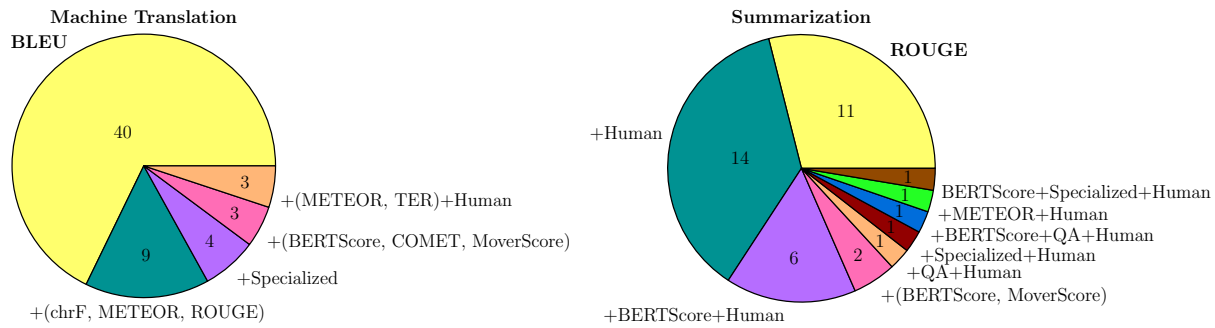


Figure E.1: Breakdowns of evaluation metrics used in the papers on machine translation and summarization from NAACL and ACL 2021. We examined all papers whose title contains "machine translation" and "summarization" and disregarded papers primarily on evaluation metrics. "QA" metrics use a QA system to evaluate summaries (e.g., Eyal et al., 2019). "Specialized" indicates specialized evaluation in a particular dimension, rather than the overall generation quality, such as document-level evaluations on contrastive sets (Voita et al., 2019).

allow researchers to compare their translation models without resource-intensive techniques such as backtranslation (Sennrich et al., 2016a), model ensembling, and deep encoders (Kasai et al., 2021a). Tables E.1 and E.2 list the hyperprameters. We generally follow the setting from Vaswani et al. (2017). We use `newstest-2019` as the dev. set and the official training data.[2] We apply Moses tokenization (Koehn et al., 2007) and BPE with 32K operations (Sennrich et al., 2016b) to English text. We tokenize Chinese text with the Jieba package,[3] following Hassan et al. (2018). Separately from English, BPE with 32K operations is then applied to Chinese. The decoder input and output embeddings are tied. Moses detokenization is applied to get the final outputs in the last step. We make the three models and preprocessed train/dev. data publicly available.[4] Table E.3 lists all generators and their automatic evaluation scores from the top-performing metric (ensemble in this case).

E.2.2 WMT20 EN-DE

Similar to WMT20 ZH-EN, we use all 14 submissions for the WMT20 EN-DE task along with our three transformer baselines. The same hyperparameters are chosen as in WMT20 ZH-EN (Tables E.1 and E.2). We preprocess both English and German text by the Moses tokenizer and *joint* BPE with 32K operations. All embeddings are shared. We apply the Moses detokenizer to

[2] http://www.statmt.org/wmt20/translation-task.html.
[3] https://github.com/fxsjy/jieba.
[4] https://github.com/jungokasai/billboard/tree/master/baselines.

Hyperparameter	Value
label smoothing	0.1
# max tokens	4096
dropout rate	0.1
encoder embedding dim	512
encoder ffn dim	2048
# encoder attn heads	8
decoder embedding dim	512
decoder ffn dim	2048
# decoder attn heads	8
max source positions	1024
max target positions	1024
Adam lrate	5×10^{-4}
Adam β_1	0.9
Adam β_2	0.98
lr-scheduler	inverse square
warm-up lr	1×10^{-7}
# warmup updates	4000
# max updates	600K
# GPUs	8
length penalty	0.6

Table E.1: Transformer-base `fairseq` hyperparameters and setting.

get the final outputs. Table E.4 shows the generators and their automatic evaluation scores from the top-performing metric (ensemble).

E.2.3 CNNDM Summarization

We submit all 26 models from Fabbri et al. (2021).[5] Table E.5 shows all models and their automatic evaluation scores from the top-performing metric (COMET).

E.2.4 MSCOCO Image Captioning

We submit the four strong models from the literature (Kasai et al., 2022d).[6] They share similar pipeline structure but vary in model architecture, (pre)training data, model size, and (pre)training objective. Table E.6 shows the models with their papers and automatic scores from the top-performing metric (**RefCLIP-S**).

[5] https://github.com/Yale-LILY/SummEval.

[6] https://github.com/jungokasai/THumB/tree/master/mscoco.

[7] Model with CIDEr optmization, https://github.com/microsoft/Oscar/blob/master/VinVL_MODEL_ZOO.md#Image-Captioning-on-COCO.

[8] Model with CIDEr optmization.

[9] Model with cross-entropy optimization, https://vision-explorer.allenai.org/image_captioning.

Hyperparameter	Value
label smoothing	0.1
# max tokens	4096
dropout rate	0.1
encoder embedding dim	1024
encoder ffn dim	4096
# encoder attn heads	16
decoder embedding dim	1024
decoder ffn dim	4096
# decoder attn heads	16
max source positions	1024
max target positions	1024
Adam lrate	5×10^{-4}
Adam β_1	0.9
Adam β_2	0.98
lr-scheduler	inverse square
warm-up lr	1×10^{-7}
# warmup updates	4000
# max updates	600K
# GPUs	8
length penalty	0.6

Table E.2: Transformer-large and transformer-large-ensemble `fairseq` hyperparameters and setting. Transformer-large-ensemble ensembles four transformer-large models with different random initializations.

Generator	Description	Score
Huoshan Translate	Wu et al. (2020a)	78.85
THUNLP	Not available	78.81
Huawei TSC	Wei et al. (2020)	78.79
DeepMind	Yu et al. (2020)	78.76
WeChat AI	Meng et al. (2020)	78.75
Tencent Translation	Wu et al. (2020b)	78.74
DiDi NLP	Chen et al. (2020)	78.66
OPPO	Shi et al. (2020)	78.59
Online-B	Not available	78.36
SJTU-NICT	Li et al. (2020c)	78.27
trans-large-ensemble	§E.2.1	77.35
trans-large	§E.2.1	76.98
Online-A	Not available	76.86
trans-base	§E.2.1	76.79
dong-nmt	Not available	76.74
Online-G	Not available	76.44
zlabs-nlp	Not available	75.79
Online-Z	Not available	75.05
WMT Biomed Baseline	Bawden et al. (2020)	73.89

Table E.3: WMT20 ZH-EN generators and reference papers. The score column indicates the score from the metric that currently correlates best with the human judgments (**ensemble**).

Generator	Description	Score
Tohoku-AIP-NTT	Kiyono et al. (2020)	90.50
Tencent Translate	Wu et al. (2020b)	90.43
OPPO	Shi et al. (2020)	90.42
eTranslation	Oravecz et al. (2020)	90.39
Online-B	Not available	90.38
Huoshan Translate	Wu et al. (2020a)	90.32
AFRL	Gwinnup and Anderson (2020)	90.16
Online-A	Not available	90.12
UEDIN	Germann (2020)	89.98
PROMT NMT	Molchanov (2020)	89.66
trans-large	§E.2.2	89.60
trans-large-ensemble	§E.2.2	89.59
trans-base	§E.2.2	89.35
Online-Z	Not available	89.26
Online-G	Not available	88.98
zlabs-nlp	Not available	88.65
WMT Biomed Baseline	Bawden et al. (2020)	88.23

Table E.4: WMT20 EN-DE generators and reference papers. The score column indicates the score from the metric that currently correlates best with the human judgments (**ensemble**).

E.3 Participating Metrics

Table E.7 discusses details and configurations of the automatic metrics that we implement in our initial BILLBOARDS.

E.4 Additional Ensemble Metric Ablations

Seen in Table E.8 are ablation studies for the ensemble metrics where one of the three selected metrics is removed at a time. Dropping one metric often has no impact on the correlation score, suggesting that these metrics are highly redundant and capture similar aspects of the output

[10] SACREBLEU implementation of sentence-level BLEU-4; https://github.com/mjpost/sacreBLEU/blob/v1.2.12/sacrebleu.py#L999.
[11] https://pypi.org/project/rouge-score/.
[12] https://github.com/mjpost/sacrebleu.
[13] https://www.nltk.org/_modules/nltk/translate/meteor_score.html.
[14] https://github.com/m-popovic/chrF.
[15] https://github.com/salaniz/pycocoevalcap.
[16] https://github.com/rwth-i6/CharacTER.
[17] https://github.com/ThomasScialom/summa-qa.
[18] https://huggingface.co/metrics/bleurt.
[19] https://github.com/Unbabel/COMET/.
[20] https://github.com/thompsonb/prism.
[21] https://github.com/salaniz/pycocoevalcap.

Generator	Description	Score
Lead-3	First 3 sentences	-0.011
T5	Raffel et al. (2020)	-0.030
BART	Lewis et al. (2020)	-0.032
Pegasus-dynamic-mix	Zhang et al. (2020a)	-0.044
RNES	Wu and Hu (2018)	-0.049
Unified-ext-abs	Hsu et al. (2018)	-0.056
Pegasus-huge-news	Zhang et al. (2020a)	-0.056
REFRESH	Narayan et al. (2018b)	-0.067
ROUGESal	Pasunuru and Bansal (2018)	-0.073
Human-H	Highlights	-0.075
NEUSUM	Zhou et al. (2018)	-0.083
BanditSum	Dong et al. (2018)	-0.083
LATENT	Zhang et al. (2018b)	-0.099
Closed-book-decoder	Jiang and Bansal (2018)	-0.112
Multi-task-Ent-QG	Guo et al. (2018)	-0.117
Pointer-Generator	See et al. (2017)	-0.144
UniLM	Dong et al. (2019)	-0.151
Bottom-Up	Gehrmann et al. (2018)	-0.160
JEC	Xu and Durrett (2019)	-0.167
Fast-abs-rl	Chen and Bansal (2018)	-0.189
NeuralTD	Böhm et al. (2019)	-0.215
Improve-abs	Kryściński et al. (2018)	-0.329
BertSum-abs	Liu and Lapata (2019)	-0.341
STRASS	Bouscarrat et al. (2019)	-0.405
GPT-2-zero-shot	Ziegler et al. (2019)	-0.441
SENECA	Sharma et al. (2019)	-0.735

Table E.5: CNNDM summarization generators and reference papers. They are from Fabbri et al. (2021), but we apply detokenization (Bird et al., 2009) and/or truecasing (Manning et al., 2014) to standardize the model outputs for better, reproducible evaluations. The score column indicates the score from the metric that currently correlates best with the human judgments (**COMET**).

Generator	Description	Score
VinVL-large[7]	Zhang et al. (2021)	83.78
VinVL-base[8]	Zhang et al. (2021)	83.45
Unified-VLP	Zhou et al. (2020b)	82.59
Up-Down[9]	Anderson et al. (2018)	80.63

Table E.6: MSCOCO image captioning generators and reference papers. The score column indicates the score from the metric that currently correlates best with the human judgments (**RefCLIP-S**).

quality. BILLBOARDS encourage researchers to explore ways to diversify automatic evaluations by updating the ensemble metric every time a new metric is submitted.

Metric	Description	Refs.	Src.	Cont.
BLEU[10]	Papineni et al. (2002)	✓	✗	✗
ROUGE-3[11]	Lin (2004)	✓	✗	✗
ROUGE-L	Lin (2004)	✓	✗	✗
METEOR	Banerjee and Lavie (2005)	✓	✗	✗
TER[12]	Snover et al. (2006)	✓	✗	✗
METEOR[13]	Banerjee and Lavie (2005)	✓	✗	✗
chrF[14]	Popović (2015)	✓	✗	✗
CIDEr[15]	Vedantam et al. (2015)	✓	✗	✗
SPICE	Anderson et al. (2016)	✓	✗	✗
CharacTER[16]	Wang et al. (2016)	✓	✗	✗
chrF++	Popović (2017)	✓	✗	✗
SummaQA[17]	Scialom et al. (2019)	✗	✓	✓
BERTScore	Zhang et al. (2020b)	✓	✗	✓
BLEURT[18]	Sellam et al. (2020)	✓	✗	✓
COMET[19]	Rei et al. (2020a)	✓	✓	✓
COMET-QE	Rei et al. (2020a)	✗	✓	✓
Prism-ref[20]	Thompson and Post (2020)	✓	✗	✓
Prism-src	Thompson and Post (2020)	✗	✓	✓
CLIP-S[21]	Hessel et al. (2021)	✗	✓	✓
RefCLIP-S	Hessel et al. (2021)	✓	✓	✓
RefOnlyC	Kasai et al. (2022d)	✓	✗	✓

Table E.7: Automatic metrics and their reference papers. The refs., src., and cont. columns indicate whether they use references, input source features, and pretrained contextual representations (e.g., BERT; Devlin et al., 2019), respectively.

E.5 Additional Mixed-Effects Analysis

Table E.9 presents fixed-effect coefficients that measure how much each automatic metric *overrates* machines over humans (§6.2.3). With some exceptions in CNNDM summarization, almost all automatic metrics *underrate* human generations (significantly positive coefficients). Table E.10 swaps the roles of human-generated text, but we still see similar patterns: almost all metrics overrate machines over humans, but the problem is mitigated in COMET-QE, a referenceless, quality estimation metric. This confirms that our findings hold independently of the design choice.

E.6 Crowdworker vs. Rubric-based Expert Evaluations

Seen in Table E.11 are examples where crowdworker evaluators (Barrault et al., 2020) and professional translators (Freitag et al., 2021) disagree: crowdworkers give lower scores to the human-

ZH-EN	–	COMET	COMET-QE	BLEURT
	0.61	0.61	0.57	0.61
EN-DE	–	COMET	COMET-QE	Prism-ref
	0.51	0.52	0.52	0.52
CNNDM	–	COMET	COMET-QE	BERTScore
	0.29	0.23	0.31	0.31
COCO	–	RefCLIP-S	RefOnlyC	CIDEr
	0.45	0.44	0.42	0.43

Table E.8: Correlations from ensemble ablation studies. One of the three selected metrics is removed at a time, and a new Lasso regression model is trained on the remaining metrics. The bigger the correlation drop is, the bigger the contribution is from the removed metric. COMET-QE is a referenceless metric.

generated translations than the machine-generated ones. The first case requires document-level context to properly evaluate. Document-level context and diversity in high-quality human translations can mislead crowdworkers.

ZH-EN	COMET-QE	Ensemble	COMET	BLEURT	BERTScore	CharacTER	MoverScore	METEOR
	$0.13_{\pm 0.01}$	$0.26_{\pm 0.01}$	$0.27_{\pm 0.02}$	$0.32_{\pm 0.02}$	$0.52_{\pm 0.02}$	$0.56_{\pm 0.02}$	$0.57_{\pm 0.02}$	$0.57_{\pm 0.02}$
	Prism-ref	chrF	TER	chrF++	ROUGE-3	BLEU	ROUGE-L	Prism-src
	$0.58_{\pm 0.02}$	$0.58_{\pm 0.02}$	$0.59_{\pm 0.02}$	$0.60_{\pm 0.02}$	$0.61_{\pm 0.02}$	$0.62_{\pm 0.02}$	$0.64_{\pm 0.02}$	$1.13_{\pm 0.02}$
EN-DE	COMET-QE	Ensemble	COMET	MoverScore	chrF	chrF++	BLEU	CharacTER
	$-0.17_{\pm 0.02}$	$0.03_{\pm 0.02}$	$0.08_{\pm 0.02}$	$0.22_{\pm 0.03}$	$0.29_{\pm 0.02}$	$0.32_{\pm 0.02}$	$0.33_{\pm 0.03}$	$0.33_{\pm 0.03}$
	BERTScore	Prism-ref	TER	Prism-src				
	$0.43_{\pm 0.02}$	$0.44_{\pm 0.02}$	$0.49_{\pm 0.03}$	$1.46_{\pm 0.03}$				
CNNDM	TER	COMET	Ensemble	BERTScore	MoverScore	COMET-QE	CharacTER	BLEURT
	$-0.58_{\pm 0.14}$	$-0.17_{\pm 0.12}$	$-0.16_{\pm 0.12}$	$-0.04_{\pm 0.12}$	$-0.03_{\pm 0.11}$	$0.02_{\pm 0.11}$	$0.14_{\pm 0.15}$	$0.25_{\pm 0.12}$
	SummaQA	ROUGE-L	BLEU	Prism-ref	chrF	chrF++	ROUGE-3	METEOR
	$0.27_{\pm 0.10}$	$0.33_{\pm 0.13}$	$0.37_{\pm 0.11}$	$0.38_{\pm 0.12}$	$0.43_{\pm 0.13}$	$0.45_{\pm 0.13}$	$0.49_{\pm 0.11}$	$0.53_{\pm 0.12}$
COCO	CLIP-S	RefCLIP-S	CharacTER	chrF	ROUGE-3	chrF++	RefOnlyC	Ensemble
	$-0.04_{\pm 0.05}$	$0.09_{\pm 0.06}$	$0.13_{\pm 0.07}$	$0.18_{\pm 0.07}$	$0.22_{\pm 0.07}$	$0.23_{\pm 0.07}$	$0.24_{\pm 0.06}$	$0.24_{\pm 0.06}$
	SPICE	METEOR	BLEU	CIDEr	ROUGE-L	BERTScore	TER	MoverScore
	$0.25_{\pm 0.07}$	$0.32_{\pm 0.07}$	$0.39_{\pm 0.07}$	$0.43_{\pm 0.06}$	$0.44_{\pm 0.07}$	$0.45_{\pm 0.06}$	$0.45_{\pm 0.07}$	$0.51_{\pm 0.05}$

Table E.9: Fixed-effect coefficients β_0 from the linear mixed-effects analysis that measures how much automatic metrics **overrate** machine text over human, as compared to human raters (§6.2.3). $\beta_0 = 0$ is neutral, and statistical significance is indicated by red (positive) or blue text (negative). The subscripts indicate 90% confidence intervals. COMET-QE, Prism-src, SummaQA and CLIP-S are referenceless metrics. In both WMT20 ZH-EN and WMT20 EN-DE, Human-B is evaluated as human-generated translations. Human-A (WMT20 ZH-EN) and Human-A and Human-P (WMT20 EN-DE) are used as the reference set for reference-based metrics.

ZH-EN	COMET-QE	Ensemble	COMET	BLEURT	TER	BERTScore	ROUGE-3	Prism-ref
	$0.03_{\pm 0.01}$	$0.07_{\pm 0.01}$	$0.08_{\pm 0.02}$	$0.09_{\pm 0.02}$	$0.23_{\pm 0.02}$	$0.24_{\pm 0.02}$	$0.24_{\pm 0.02}$	$0.25_{\pm 0.02}$
	CharacTER	ROUGE-L	chrF	MoverScore	METEOR	chrFpp	BLEU	Prism-src
	$0.25_{\pm 0.02}$	$0.26_{\pm 0.02}$	$0.27_{\pm 0.02}$	$0.27_{\pm 0.02}$	$0.29_{\pm 0.02}$	$0.29_{\pm 0.02}$	$0.30_{\pm 0.02}$	$0.79_{\pm 0.02}$
EN-DE	COMET-QE	Ensemble	COMET	MoverScore	Prism-ref	chrF	BERTScore	CharacTER
	$-0.09_{\pm 0.02}$	$-0.07_{\pm 0.02}$	$-0.06_{\pm 0.03}$	$0.02_{\pm 0.02}$	$0.18_{\pm 0.02}$	$0.20_{\pm 0.02}$	$0.21_{\pm 0.02}$	$0.22_{\pm 0.02}$
	chrF++	BLEU	TER	Prism-src				
	$0.22_{\pm 0.02}$	$0.23_{\pm 0.02}$	$0.32_{\pm 0.02}$	$1.38_{\pm 0.03}$				

Table E.10: Fixed-effect coefficients β_0 from the linear mixed-effects analysis that measures how much automatic metrics **overrate** machine text over human, as compared to human raters (§6.2.3). **The roles of human translations are swapped**: Human-A is evaluated, and Human-B (WMT20 ZH-EN) and Human-B and Human-P (WMT20 EN-DE) are used as the reference. We still see similar patterns to Table E.9: almost all automatic metrics overrate machines over humans, but the problem is less severe in the referenceless metric of COMET-QE.

	WMT20 ZH-EN	
Source	希望兴安省继续为白俄罗斯企业提供便利条件。	凭的是相机而动的时势驾驭。
Huoshan	It is hoped that **Xing'an Province** will continue to provide convenient conditions for Belarusian enterprises.	It is based on the current situation of the **camera**.
Human-A	He hoped that **Hung Yen Province** would continue to provide convenient conditions for Belarusian enterprises.	This relies on the ability to seize opportunities.
Human-B	He hoped that **this** could continue in the future.	It is based on the observation of various situations at different times.

Table E.11: Examples where crowdsource evaluators (Barrault et al., 2020) and professional translators (Freitag et al., 2021) disagree: crowdworkers give lower scores to the human-generated translations than the machine-generated ones. The first case requires document-level context to properly evaluate. 兴安省 is Hung Yen Province in Vietnam in this context, but there is entity ambiguity. (Xing'an Province that existed in the Republic of China.) The second one illustrates the diversity of human generations that misleads crowdworkers.