Mitigating Semantic and Distributional Discrepancies in Natural Language Processing

Chunting Zhou

CMU-LTI-22-006

Spring 2022

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15123

Thesis Committee:

Graham Neubig (Chair)	Carnegie Mellon University
Shinji Watanabe	Carnegie Mellon University
Zico Kolter	Carnegie Mellon University
Luke Zettlemoyer	University of Washington & Facebook AI Research

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: robustness, distribution shift, group distributionationally robust optimization, parameter-efficient transfer learning

Abstract

The advancement of neural network models has led to state-of-the-art performance in a wide range of NLP tasks, e.g. machine translation and question answering. Despite the remarkable performance gains, NLP systems deployed in the wild are brittle and fragile as it always experiences a shift in the data distribution: the training data distribution is different from the one at test time. For example, a multilingual machine translation model is expected to perform uniformly well across a set of language pairs while the training resources can be extremely imbalanced across different language pairs. Such distribution shift is also ubiquitous in the modern pretrain-then-fine-tune paradigm, where models pre-trained on a large text corpus are fine-tuned on various downstream tasks. Real-world applications demand robust adaptation methods such that a pre-trained model is robust to various types of distribution shift in a dynamically changing test environment.

To counter distribution shift, the goal of this thesis is to identify potential problems when models are evaluated under distribution shift, and to mitigate such discrepancies by developing distributionally robust methods and efficient transfer learning methods. This thesis consists of three parts. In the first part, we focus on the hallucination problem in conditional sequence generation where a model can generate fluent outputs but not faithful to the input text, particularly when tested on out-of-domain data. We identify and quantify the unfaithful tokens in the machine outputs and leverage the created tools to improve training in a low-resource setting by reducing hallucinated content in the noisy training data. The second part presents our distributionally robust methods for subpopulation shift where the training data is a mixture of different subpopulations, e.g. different languages, demographic groups, etc, and the test distribution is a subpopulation of it. Models trained on a dataset with imbalanced distribution of subpopulations can perform poorly on data from minority subpopulations. To mitigate this, we develop group-level distributionally robust methods that perform well over a set of potential test distributions. The last part of this thesis focuses on robust transfer learning of large-scale pre-trained language models. As the size of pre-trained language models (PLMs)is increasing every year, how to effectively adapt these models to downstream tasks is becoming increasingly important as models can catastrophically forget its previously acquired knowledge during transfer learning. Parameter-efficient fine-tuning provides an effective and robust way for this by only tuning a small number of additional parameters. To this end, we propose a unified framework to connect parameter-efficient transfer learning methods and instantiate a new state-of-the-art method. Furthermore, we developed a method based on parameter-efficient tuning to improve the performance of a large-scale zero-shot transfer learner.

Acknowledgments

About six years ago, when I first started my academic life in CMU, I only had little fragmented knowledge about the area of natural language processing and I had no idea how far I could go along this journey. Looking back now, it's been an incredible and invaluable experience for me to spend the past years in CMU where I have grown in every aspect of a researcher, witnessed the advancement of Artificial Intelligence and made acquaintance with many brilliant minds. Ph.D. is a long journey that I could not have completed without the help and support of many people and I want to thank them all here.

First and foremost, I would like to thank my excellent advisor Graham Neubig. His passion and commitment towards high-quality research, and his enthusiasm and willingness to learn new knowledge always inspires me to push beyond my best effort. He taught me how to write, make presentations, execute insightful analysis and explain things clearly to people who lack background knowledge. I am grateful for everything he taught me and the time we learnt together throughout my exploration about various topics. More importantly, Graham is an extremely kind, considerate and supportive mentor. I am very grateful for his support when I switched the research direction from generative modeling to robustness in the fourth year of my PhD, which made this dissertation of what it is today.

I also would like to thank the rest of my thesis committee members, Zico Kolter, Shinji Watanabe and Luke Zettlemoyer, for their insightful feedback. Luke was also my internship mentor at Facebook AI Research in the summer of 2020. He not only has an insightful, high-level view about the field but also knows all the details of the problem. I am also very grateful for his important support after my internship.

Over the past six years, I was fortunate to collaborate and interact with many excellent researchers. I thank my fellow collaborators who made this dissertation possible: Junxian He, Paul Michel, Daniel Levy, Marjan Ghazvininejad, Jiatao Gu, Luke Zettlemoyer, Xian Li, Mona Diab, Paco Guzman and Taylor Berg-Kirkpatrick. In particular, I want to thank Junxian He, who is a close collaborator and a good friend of mine. We often exchanged ideas about interesting and promising research directions and I enjoyed all the wonderful discussions and brainstorm meetings with him. I also appreciate all the great discussions on distributionally robust optimization I had with Paul and Daniel, and I learned a lot from them. I want to thank my collaborators in other papers and projects which didn't make it into this document: Pengcheng Yin, Junjie Hu, Aditi Chaudhary, Xiang Kong, Di Wang, Kyunghyun Cho, Eduard Hovy and Jaime Carbonell. Pengcheng is my friend since my MPhil at The University of Hong Kong. I was always motivated to improve myself further by his perseverance in doing impactful research.

During my PhD, I had two wonderful internships at Facebook AI Research. I thank my

mentors Jiatao Gu, Marjan Ghazvininejad and Luke Zettlemoyer when I worked at these places. The work in Chapter 5 on detecting hallucinated content is a collaboration with Marjan and Luke during my second internship. I have learned a lot from Jiatao since my MPhil time at The University of Hong Kong through various discussions. He has a real passion for research and a great problem-solving ability, which opened my mind when thinking about solutions to research problems.

I am thankful to members of the whole NeuLab group: Shuyan Zhou, Frank Xu, Zecong Hu, Hao Zhu, Zhengbao Jiang, Mengzhou Xia, Pengfei Liu, Cindy Wang, Shruti Rijhwani, Aditi Chaudhary, Patrick Fernandes, Danish, Hiroaki Hayashi, Lucio Dery, Uri Alon, John Wieting, Kayo Yin, Atieno Perez Ogayo, Antonis Anastasopoulos, Emmy Liu, Ziyi Dou, Junjie Hu, Pengcheng Yin and Junxian He, who gave me support at various times. The time I spent with them becomes one of my best memories in graduate school. I thank Pengfei for sharing his vision for future research and his advice on mentorship. I wish I had these conversations earlier with Pengfei. I will remember those relaxing gatherings and joyful moments I had with Shuyan, Frank, Pengcheng, Junxian and many others, who helped me out from the stressful PhD life.

I also would like to thank other friends in LTI who I am lucky to befriend with, make discussions with or chat with (forgive me for not being able to list all of them): Jingzhou Liu, Di Wang, Hector Liu, Jiateng Xie, Jiarui Xu, Yulun Du, Qizhe Xie, Guokun Lai, Zichao Yang, Chenyan Xiong, Zhuyun Dai, Haohan Wang, Zhiting Hu, Zihang Dai, Diyi Yang, Zhou Yu, Tian Tian, Keyang Xu, Shikun Zhang, Siddarth Dalmia, Shruti Palaskar, Abhilasha Ravichander, Chirag Nagpal, and all the others. Also many thanks to my dear friends outside LTI for their companionship, support during those good days or bad days: Xiaojing Zhu, Rui Zhong, Futeng Wan, Yujie Zhang, Xianyi Cheng, Ziyi Chen, Yi Chen, Xiaoguang Han and Xiaomeng Li.

I am thankful to my mentor Jinhui Yuan during my very first intership at MSRA in my college, who enlightened and led me into the research of deep learning. I also want to thank my advisor Francis C.M. Lau at The University of Hong Kong, who exchanged thoughts and ideas with me for a couple of times during my PhD. He has a high expectation on me of being a good researcher that benefits the world and the society. I hope I will not disappoint him for now and in the future.

I also would like to thank the staff at LTI, especially Stacey Young and Mary Jo Bensasi, who always patiently answered my questions regarding graduation requirements, part-time CPT application, etc. I am thankful to their hard work for all the LTI students.

Special thanks to my parents: Hongju Wang and Chuankai Zhou. Their unconditional love and care made me who I am today. Although they are living 12 hours ahead of me and I

haven't seen them in person for almost 3 years due to the pandemic, their love and support for me never changes. I am also thankful to other family members who always remind me of taking care of myself.

Lastly, I would like to thank Max, who is my partner, my close collaborator, my best friend and a role model for my life and career for his integrity, intelligence, dedication and boldness. I thank all his love, support and encouragement especially during the rock bottom of my PhD life when I had health issues, when I lost my direction. Without him, I would not go as far as where I am today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Language has a fundamental role in facilitating comprehension, communication and creation. As a unique and effective tool of communication accessible to human beings, language relates to social structure and contexts such as gender, age, geography, occupation, religion, identity and cultural backgrounds. These different aspects define the dynamic nature of language and how it exhibits a variety of properties under different contexts. Thus, when applying a natural language processing (NLP) system to the real-world text inputs, it requires that the intelligent agent must be able to cope with a changing test environment, which can be significantly different from the one seen at training time (Lazaridou et al., 2021; Dhingra et al., 2022).

Modern NLP models powered by neural networks have made remarkable advances in a number of applications, such as machine translation (Fan et al., 2021), open domain question answering (Izacard and Grave, 2020) and code generation (Chen et al., 2021b). Training these neural network models requires a large training corpus consisting of heterogeneous data points with a substantial variation in topics, domains, and even languages. And it is often the case that data with diverse characteristics and content are not equally represented in the training set, e.g. the majority users of an online forum Reddit are male (Statista, 2022). However, we expect a model to perform fairly on different demographics in the real world. Recently, the NLP community has adopted a pre-train then fine-tune paradigm to perform downstream tasks (Devlin et al., 2019a; Lewis et al., 2020a). Under this paradigm, a large neural network is pre-trained as a language model on a massive corpus and then is fine-tuned on downstream tasks with task-specific annotated examples. With the ever-increasing size of pre-trained languages models (PLMs), fine-tuning all parameters on a relatively small training set is not only prone to over-fitting but also results in catastrophic forgetting of previously learned knowledge at pre-training stage. This is disadvantageous when the pre-trained model is expected to be reused for different tasks and adapt to new data distributions.

The prevalence of distribution shift in the wild hinders the deployment of real-time machine learning models(Holtzman et al., 2019; Marcus and Davis, 2020; Metz and Satariano, 2020), as models can fail in different ways when exposed to a distribution shift from the training environment:

- They perform poorly on *under-represented* subpopulations: when models are learned by minimizing the average training loss on all the training data (known as ERM defined in §2), the majority classes contribute a large part to the training objective. Models can recklessly rely on spurious correlations between the input and the output embedded in the majority classes or underestimate the tail subpopulation. As a result, using these models can lead to negative consequences on underrepresented groups: a multilingual model often performs worse on low-resource languages (Arivazhagan et al., 2019); a hiring system can adversely prefer male candidates over female candidates with similar profiles (Xu et al., 2019); toxicity detection systems predict tweets containing features associated with African-American English as more offensive than tweets without these features (Sap et al., 2019; Davidson et al., 2019).
- They can undergo *catastrophic forgetting* of previously acquired knowledge when models are adapted to new domains or new tasks (Kirkpatrick et al., 2017; Veniat et al., 2021). This poses great challenges especially under the pre-train-then-fine-tune paradigm where a large model (e.g. with billions of parameters) pre-trained on a large corpus has stored a wealth of factual knowledge (Dai et al., 2021) and textual patterns (Geva et al., 2021). The vanilla fine-tuning method tunes all the parameters of the pre-trained models when adapting to a downstream task (Devlin et al., 2019b), which inevitably leads to loss of information previously learned. We hope that the knowledge or patterns acquired during pre-training are preserved after the adaptation stage, as the same model can be reused for other tasks. In addition, models not overfitting to a specific dataset tend to generalize better, especially under the low-resource settings.
- They are vulnerable to *variations of the inputs* carrying synonymous meanings. This phenomenon is widely studied in the research line of adversarial examples where crafted noise such as word or character-level perturbations (Ebrahimi et al., 2017; Michel et al., 2019; Wang et al., 2021a) are injected into the inputs such that they can alter models' predictions while being imperceptible to humans. More recently, prompting-based methods have become an effective way of leveraging pre-trained language models for downstream tasks especially in the zero-shot or few-shot learning setting. These methods reformulate a variety of NLP tasks to mimic the original pre-trained LM objective with the help of a textual prompt (Liu et al., 2021b; Brown et al., 2020). It has been found that the pre-trained models are sensitive to different wordings of prompts (Jiang et al., 2020), which results in inconsistent predictions given synonymous prompted inputs. The variations of real-world text inputs are abundant and frequent, thus we need to develop methods that are robust to these synonymous variations.

To remedy these issues, one important step towards robust machine learning is to identify and assess potential risks in out-of-domain environments and take into account the demand of dynamic evaluation at training time. In this thesis, we try to improve the robustness of NLP models from three aspects:

- *Diagnosis and mitigation of unfaithful content in conditional neural sequence generation.* Neural sequence generation models have been the driving force of many NLP applications, such as machine translation (Vaswani et al., 2017), question answering (Izacard and Grave, 2020), abstract text summarization (Rothe et al., 2020), and code generation (Chen et al., 2021b). Generally speaking, these tasks generate the target output by conditioning on an input text or a user query, which put constraints on what is expected in the target. The outputs should faithfully reflect the content in the source input such that there are no fabricated facts or irrelevant information which mislead the users. Unfortunately, neural sequence generation systems often hallucinate content that is not faithful to the source (Maynez et al., 2020). In Part I, we focus on detection and mitigation of hallucinated content in conditional sequence generation systems.
- *Mitigating subpopulation shift with improved learning objectives.* As discussed above, the training distribution often comprises data from a variety of types, e.g. text from different topics, written by different demographic groups, or in different languages. Under this setting, there are two *challenges* of developing a model that performs well on all subpopulations at test time: (1) the real data is highly imbalanced across data varieties and ERM training can result in a model that performs poorly on minority groups or learns spurious correlations presented in the majority class (Sagawa et al., 2020a); (2) the varying intrinsic levels of learning difficulty and the intra-type heterogeneity can prevent positive and effective transfer between groups (Pham et al., 2021). To this end, in Part II we propose group distributionally robust optimization methods (Ben-Tal et al., 2013a; Duchi et al., 2016; Sagawa et al., 2020a) that encourage uniform good performance across all the groups.
- *Preventing catastrophic forgetting with parameter-efficient transfer learning.* A key challenge in the pre-train-then-fine-tune paradigm is how to efficiently adapt a pre-trained model to any downstream tasks through transfer learning. There are several desiderata for this goal: for any downstream tasks, fine-tuning should be fast and parameter-efficient; the knowledge stored in the pre-trained models can be reused across different tasks which can also facilitate the generalization ability of PLMs. Efficient transfer learning enables flexible adaptation and is widely applicable to a variety of scenarios, e.g. continual learning of real-world knowledge. To realize these goals, in Part III we propose a unified framework for parameter-efficient transfer learning that connects different methods and allows us to instantiate a new state-of-the-art method for such learning efficiencies. We further explore the use of parameter-efficient fine-tuning in a zero-shot continual learning setting to improve the consistency of model predictions under different input views.

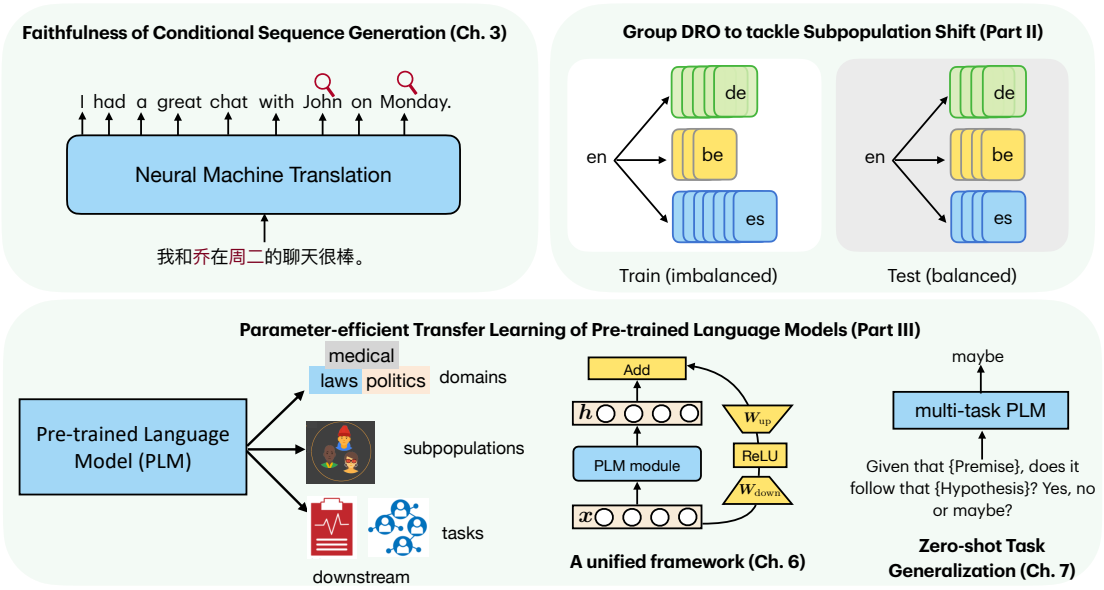


Figure 1.1: An overview of the thesis.

1.2 Thesis Overview

The detailed outline of this thesis is presented below:

- Chapter 2 introduces scenarios where models are fragile and fail to make correct predictions. We also survey on the approaches that aim to improve the robustness of NLP systems.
- **Part I: On the Faithfulness of Conditional Sequence Generation**
  - In Chapter 3, we propose a method to detect hallucinated content that is unfaithful to the input for conditional sequence generation. As there is no readily available labeled data for this task, we create synthetic labeled data with automatically inserted hallucinations using a pretrained denoising autoencoder model. We also use this hallucination detection model to create token-level hallucination labels for the noisy parallel training data and proposed a fine-grained loss to improve the performance on machine translation with low-quality training data. This work was published at ACL-IJCNLP Findings 2021 (Zhou et al., 2021c).
- **Part II: Group Distributionally Robust Optimization**
  - In Chapter 4, we study group distributionally robust optimization (group DRO), which is a popular robust optimization method to achieve high worst-group accuracy. We find that group DRO can fail when the pre-defined groups are not the "perfect" ones, i.e. groups do not directly account for various spurious correlations. To address this, we propose a group-conditional DRO algorithm that minimizes the worst loss over a more flexible set of distributions that are defined on the joint distribution of groups and their instances. The proposed

method outperforms group DRO by a large margin when confronted with imperfect partitions, and it also performs well as regards to perfect partitions. This work was published at ICML 2021 (Zhou et al., 2021b).

- In Chapter 5, we extend group DRO to a more challenging real-world problem, multilingual machine translation, where groups correspond to different languages. In multilingual training, the data across languages can be highly imbalanced and there is also a varying level of difficulty in learning each language. When model capacity is limited, this results in trade-offs or decreased performance on some languages, particularly low-resource languages. We propose a new learning objective based on group DRO for multilingual training which is both efficient (incurs negligible computational overhead compared to ERM) and effective (strong empirical results on multilingual machine translation). This work was published at EMNLP 2021 (Zhou et al., 2021a).

- **Part III: Efficient Transfer Learning Approaches of Pre-trained Language Models**

  - In Chapter 6, we propose a unified framework connecting many recently proposed parameter-efficient transfer learning methods that only tune a small number of additional parameters when fine-tuning a pre-trained language model. These methods naturally overcome catastrophic forgetting at fine-tuning time and have potential for performing efficient continual learning. Our framework allows us to study different methods along several design dimensions we have defined, identify the critical design choices, and transfer design elements across approaches. Under this framework, we propose a new state-of-the-art parameter-efficient tuning approach that matches full fine-tuning by tuning 6.7% of relative number of parameters for complex and high-resource NLP tasks. This work was published at ICLR 2022 (Zhou et al., 2022).

  - In Chapter 7, we propose an unsupervised learning method to improve the zero-shot task generalization performance of the pre-trained language models. Our method is based on a recently proposed state-of-the-art zero-shot learner that is pre-trained in a supervised and massively multi-task fashion. At training time, all of the NLP tasks are represented in a unified way with natural language prompts. We explore unannotated examples to improve the consistency of model predictions using different prompts and further improve the zero-shot performance. We also adopt a parameter-efficient fine-tuning method for model training. Empirically, we outperform the previous SOTA by a large margin without using any labeled data, in the mean time, our method also introduces a novel learning paradigm–test-time tuning–that continues learning and improving the model at test time. This work is in submission.

Chapter 2

Background and Literature Review

Robustness is increasingly and widely studied across diverse dimensions and modalities. In this chapter, we first discuss different scenarios where NLP models can fail by categorizing different types of distribution shift in §2.1, which also motivates our work in this thesis. Next in §2.2, we survey existing mitigation strategies in the literature.

2.1 Characterization of Robustness in NLP

Machine learning is generally concerned with learning to make predictions given sample data. Consider predicting a target random variable $y \in \mathcal{Y}$ from input variable $x \in \mathcal{X}$, where $\mathcal{X}$ represents the input space (e.g. all valid German sentences in de-en translation) and $\mathcal{Y}$ represents the space of all possible outputs (e.g. all valid English sentences in de-en translation), ideally a machine learning model is capable of learning correlations between x and y (i.e. the conditional distribution $p(y|x)$) that can generalize to any test distribution. Many machine learning models adopt empirical risk minimization (ERM, Equation 2.1) that minimizes the expected risk:

$$\mathcal{L}_{\text{ERM}} = \mathbb{E}_{(x,y)\sim P_{\text{data}}}[\ell(x, y; \theta)] \tag{2.1}$$

via minimizing the empirical risk:

$$\hat{\mathcal{L}}_{\text{ERM}} = \frac{1}{N}\sum_{i=1}^{N} \ell(x_i, y_i; \theta) \tag{2.2}$$

In ERM, models are trained and evaluated on randomly shuffled and split training-test sets based on the i.i.d. assumption, where the training set is composed of a finite number of samples drawn from the same distribution $(x, y) \sim P_{\text{train}}$. However, this training distribution might be divergent from the test distribution P_{test}. This divergence in distribution shift may prevent the model from learning relationships between x and y that generalize to the test distribution, resulting in models that achieve high performance on data similar to the training set but fail on test distributions that drift away from the training

one. For example, a machine translation system trained on news data may be evaluated on a broader set of domains and may perform poorly on, for example, twitter data. In this section, while we will not attempt to formulate a full taxonomy of distribution shift, we will categorize commonly studied robustness issues in the NLP literature. In particular, the challenges introduced in §2.1.1 are closely related to the problems this thesis aims to tackle.

2.1.1 Distribution Shift

Distribution shift is ubiquitous in the real-world setting. This thesis primarily focuses on distribution shift that occurs naturally in practice. One categorization of distribution shift (Koh et al., 2020) is based on if the test domain data has been seen in the training data, where each domain d corresponds to a distribution P_d over data points (x, y) that are similar in some way. It considers two broad categories of distribution shift: domain generalization and subpopulation shift.

Domain Generalization This type of distribution shift is also known as out-of-domain generalization, in which the training and test distributions comprise data from related but disjoint sets of domains, i.e. $D_{\text{train}} \cap D_{\text{test}} = \varnothing$, where D_{train} and D_{test} are the set of domains of training and test data respectively. Because it is often infeasible to collect data from all the domains of interest, this issue arises naturally in many applications. For example, a legal document summarization model may be trained on data from business law, criminal law and health care law, while it is applied to a more diverse set of laws at test time, e.g. constitutional law. Domain generalization also frequently arises in domains beyond NLP, for example, a pneumonia detection model trained using chest X-ray data collected from several hospital could be later applied more broadly to hospitals outside the training set (Zech et al., 2018). In Chapter 7, we tackle zero-shot task generalization in the context of prompt-based pre-trained models where all the tasks are reformulated as text-to-text generation tasks, which is essentially a domain generalization problem.

Subpopulation Shift In this type of distribution shift, the training and test domains overlap, but their relative proportions differ, i.e. the test distributions are subpopulations of the training distribution $D_{\text{test}} \subseteq D_{\text{train}}$. Specifically, the training distribution can be formulated as a mixture of K different domains $P_{\text{train}} = \alpha_1 q_1 + \alpha_2 q_2 + \cdots + \alpha_K q_K, \sum_i \alpha_i = 1$, and the mixture weights of the test distribution are different from the training one. Subpopulation shift also naturally arises in many cases, because the uneven spread of data across domains prevents us from easily collecting data from every domain we are interested in. Even if under these constraints, we wish a machine learning model to perform well even on the worst-case subpopulations. For example, a toxicity detection model that tells whether an online comment is toxic or not should perform equally well on all demographic subpopulations, e.g. African American, White American, Asian, etc. However, it is well known that standard models perform poorly

on under-represented demographics (Hashimoto et al., 2018a; Buolamwini and Gebru, 2018; Koenecke et al., 2020). Our works in Part II focuses on alleviating subpopulation shift in various scenarios, including reducing the reliance on spurious correlations (§2.1.2) and improving multilingual machine translation where the training data is imbalanced across language pairs.

Another categorization of distribution shift distinguishes various shift types by mathematically describing how the data generation process changes between two distributions. Under this umbrella, Three commonly studied types of distribution shift are covariate shift (Shimodaira, 2000), label shift (Quiñonero-Candela et al., 2008) and concept shift (Quiñonero-Candela et al., 2008). Suppose the data generation process can be factorized into a joint distribution $p(x, y) = p(x)p(y|x)$ over x and y, *covariate shift* refers to the change in the distribution of the input variable, i.e. $p(x)$ and the conditional distribution $p(y|x)$ stays the same; *concept shift* on the other hand refers to the change in the conditional distribution $p(y|x)$ while the marginal distribution $p(x)$ is fixed. It is often assumed that the conditional $p(y|x)$ is invariant across in supervised classification problems (Arjovsky et al., 2019), which is the setting we study in Chapter 4. However, in text generation tasks (e.g. the translation tasks in Chapter 5), we often see a combination of covariate shift and covariate shift as the language variations may be divergent across domains or languages, which is more realistic and common in a variety of NLP tasks Finally, motivated by medical diagnosis where diseases (output y) cause symptoms (input x), we can factorize the data distribution as $p(x, y) = p(y)p(x|y)$, and under which *label shift* refers to the change in the distribution of the target variable $p(y)$ while the conditional $p(x|y)$ does not change.

2.1.2 Robustness Failures in NLP

Although there are many instantiations of different types of distribution shift, we provide an overview of some widely (but incomplete) investigated robustness scenarios where NLP system fail to make accurate predictions in the face of distribution shift that is artificially produced or naturally arises.

Adversarial Attacks While distribution shift naturally arises in the real world, adversarial attacks create adversarial examples by inserting artificially crafted perturbations or noise into the input texts x to deceive the model such that it makes wrong predictions (Goodfellow et al., 2014; Ebrahimi et al., 2017). This is motivated by the observation that models are sensitive to the variations of inputs such that their predictions can be altered when the input is transformed with changes imperceptible to humans. Some popular techniques for creating adversarial examples in NLP design perturbation functions by optimizing a pre-defined adversarial loss, which is based on the principles of label-preserving, label-changing and semantics-preserving. These methods include token replacement with gradient-based methods (Michel et al., 2019), token or character swapping (Ebrahimi et al., 2017; Alzantot et al., 2018; Ren et al., 2019), paraphrasing (Gan and Ng, 2019; Iyyer et al., 2018), and adding adversarial distractors (Jia and Liang, 2017). Morris et al. (2020) question the validity of adversarial examples in terms of semantics-preserving,

as they found perturbations often do not preserve semantics and they call for fair and accurate evaluations with different adversarial perturbation methods.

Dataset biases (Goyal et al., 2017; McCoy et al., 2019; Gururangan et al., 2018) are a typical cause of model failures, where spurious associations between the input and the output variables are presented in the dataset due to annotation artifacts or biases introduced during data collection phase. For example, in natural language inference (NLI) tasks, where a model is asked to label if a hypothesis contradicts, entails or is neutral to a premise, McCoy et al. (2019); Geva et al. (2019) found that current NLI models tend to rely on the spurious features in the hypothesis to predict labels, e.g. using the strong association between the high word overlap between premise and hypothesis and the label "entailment". Dataset biases may come from annotation artifacts (Gururangan et al., 2018) that annotators choose specific words in their inputs to realize a certain label, for example, one can add a cause ("because") to the premise and create a contradicting sentence by negating the premise. Standard models relying on these spurious correlations can achieve high average accuracy on data similar to the training set but fail on test data where the correlation does not hold. Our work in Chapter 4 tries to prevent the model from relying on spurious correlations using a DRO approach. Another finding in question answering by Lewis et al. (2021a) show that there is a significant overlap between the test set answers and the training set one in three popular open-domain benchmark datasets.

Neural sequence generation produces an output sentence from an encoder-decoder or a decoder-only model using some decoding algorithm, and has become the backbone for many NLP applications, such as machine translation (Vaswani et al., 2017), text summarization (Rothe et al., 2020), and free-form question answering (Fan et al., 2019). However, they often generate fluent text that fails in a variety of aspects, which impedes their safe and responsible deployment in the wild. Some commonly discussed issues include lacking global logical consistency (Marcus and Davis, 2020), toxic outputs with societal biases regarding gender, race and religion (Brown et al., 2020), hallucinated outputs that are unfaithful to the source inputs (Maynez et al., 2020), etc. These issues can often be attributed to domain generalization where models are evaluated out-of-domain (Wang and Sennrich, 2020), subpopulation shift where models are evaluation on low-resource domains (Chapter 3) or dataset biases (Maynez et al., 2020; Liu et al., 2020a). Recent works have made some remarkable progress in responsible and robust sequence generation (Ouyang et al., 2022; Wang et al., 2022; Sap et al., 2021), but there is still a long way to go. In Chapter 3, we examine the hallucination problem in neural sequence generation by proposing methods for detecting hallucinated content in the model output and using it to reduce hallucination.

Transfer learning essentially is dominating the NLP field, as the pretrain-then-fine-tune paradigm, where pre-trained language models are tuned on a downstream task, has been adopted in most NLP appli-

cations (Devlin et al., 2019a). The domain of the downstream task is generally different from the domain of the pre-training corpus, which involves the adaptation of pre-trained weights to the target task using task-specific objectives. This is also closely related to subpopulation shift and domain generalization. While pre-trained models have achieved exceptional progress in various tasks, researchers scale the size of pre-trained language models every year (Liu et al., 2019; Brown et al., 2020; Chowdhery et al., 2022), which incurs more computational cost at the adaptation stage and degenerated performance when the training set is small (Phang et al., 2018). Moreover, fine-tuning all the parameters of a pre-trained model can result in catastrophic forgetting of previously acquired knowledge, which is not an efficient method of adaptation as each task requires a new copy of model parameters and it is not extensible for continual learning. Recently, parameter-efficient fine-tuning and prompting methods provide new opportunities for effective transfer learning (Li and Liang, 2021a; Zhou et al., 2022), few-shot learning (Brown et al., 2020; Gao et al., 2021) and zero-shot learning (Sanh et al., 2021)s, which naturally alleviates catastrophic forgetting and perform well in low-resource settings. Our works in Part III are dedicated to efficient and robust transfer learning with pre-trained language models using parameter-efficient tuning.

2.2 Methods for Improving Robustness

There are a number of lines of work that aim to improve robustness in NLP systems. While we do not attempt to provide a complete survey of all existing methods, we survey methods spanning data intervention/augmentation, transfer learning with pre-trained model, reweighting and causality.

2.2.1 Data-driven approaches

Data augmentation methods increase the diversity of training data without directly collecting more data with human annotators. It has been shown effective in improving performance in low-resource setting, few-shot learning and mitigating biases and improving robustness in NLP models (Feng et al., 2021). Data augmentation techniques typically include (1) heuristic/rule-based techniques, such as word/character dropout (Wei and Zou, 2019) , Cutoff (Shen et al., 2020), and leveraging predicate-argument structures (Moosavi et al., 2020), (2) example interpolation techniques that interpolate the text or hidden representation of existing examples to create synthetic examples, such as MixText (Chen et al., 2020), Seq2MixUp (Guo et al., 2020) and Hiddencut (Chen et al., 2021a), and (3) model-based techniques that generates synthetic examples from a trained sequence generation model, such as back translation (Sennrich et al., 2016a), self training (He et al., 2020) and paraphrasing (Sun et al., 2021). Our work in Part I proposes a truncated loss to remove losses of potentially hallucinated tokens, falling into this category.

2.2.2 Adaptation approaches with pre-trained models

As discussed in §2.1.2, transfer learning, also known as adaptation, is essentially extracting knowledge from a source task/domain and applying it to a different target task/domain. The ability to perform

transfer learning is more and more a requirement in various NLP tasks, in particular the AI community is now at a stage where we aim to enable a single AI system to generalize across thousands or millions of tasks and to understand different types of data (Chowdhery et al., 2022). One taxonomy (Ruder et al., 2019) classifies transfer learning into (1) *transductive transfer learning* where the source and target tasks are the same and a model trained on the source domain/task is adapted to a target domain or language (He et al., 2019; Neubig and Hu, 2018; Dou et al., 2019), and (2) *inductive transfer learning* where the source task/domain the model is trained on is different from the target task/domain. Transfer learning of pre-trained models to downstream tasks belongs to the latter category as the pre-training tasks (i.e. language modeling tasks) are usually different from the downstream tasks (Devlin et al., 2019b; Liu et al., 2019; Brown et al., 2020).

With the advancement of pre-training, it has become the de facto workhorse of almost all the NLP tasks. However, as pointed out in 2.1.2, transfer learning of large pre-trained language models may result in catastrophic forgetting of stored knowledge, over-fitting to small training set, inconsistent predictions across different runs or inputs (Jiang et al., 2020) and large compute overhead of saving model copies for many tasks. To alleviate these issues, previous work has recently proposed parameter-efficient fine-tuning methods (Houlsby et al., 2019; Lester et al., 2021; Li and Liang, 2021a; Hu et al., 2021) that freeze the pre-trained model weights and only fine-tune a small number of additional parameters. These methods have shown similar performance to full fine-tuning while being able to keep the pre-trained weights intact. Parameter-efficient transfer learning methods have potential for more effective continual learning in the real world. In Part III, we explore this potential by connecting existing methods, instantiating a new state-of-the-art method and examing their utility in the zero-shot task generalization setting.

2.2.3 Generalized Reweighting

To counter distribution shift especially subpopulation shift, one broad class of *generalized reweighting* (GRW) (Zhai et al., 2022) algorithms iteratively assign each example a weight during training and iteratively minimize the weighted average risk:

$$\hat{\mathcal{L}}_{\text{GRW}} = \sum_{i=1}^{N} q_i^{(t)} \ell(x_i, y_i; \theta) \tag{2.3}$$

Depending on whether the weights $q_i^{(t)}$ are updated every iteration, these methods can be categorized into (1) Static weighting GRW, which assigns each example with a fixed weight throughout training. This category includes the classic method *importance weighting* that commonly reweights the classes proportionally to the inverse of their frequencies (Wang et al., 2017; Arivazhagan et al., 2019; Conneau et al., 2020a). (2) Dynamic weighting GRW, which adjusts the weights dynamically during training. Dynamic GRW is gaining more and more attention recently as it provides a more flexible weighting strategy based on the training dynamics. One class of dynamic GRW heuristically leverages the training

dynamics to perform curriculum learning that reweight examples based on the prediction confidence (Lin et al., 2017), upweight examples that are not well classified (Liu et al., 2021a), or focus more on "easy" examples based on the training loss (Jiang et al., 2015). A second class of dynamic GRW learns a scorer function to optimize data usage (Zhang et al., 2020b; Lahoti et al., 2020; Wang et al., 2020d,c) using meta learning or adversarial training to upweight training examples that have a larger similarity to examples in a validation set of interest. Last, distributionally robust optimization (DRO) refers to a large family of methods that are based a principled mathematical framework are that minimizes the worst expected loss over an adversarial distribution q:

$$\mathcal{L}_{\text{DRO}} = \sup_{q \in \mathcal{Q}} \mathbb{E}_{(x,y)\sim q}[\ell(x, y; \theta)] \tag{2.4}$$

This worst-case objective upper bounds the test risk for all $q_{\text{test}} \in \mathcal{Q}$, which potentially guards the model against a wide range of test distributions. The uncertainty set $\mathcal{Q}$ is usually defined as a neighborhood around an empirical training distribution P_{train}, which encodes potential test distributions that we wish the model to perform well on. The notion of distance is often defined by some divergence, e.g. KL-divergence ball (Michel et al., 2022). Instances of DRO algorithms include CVaR-, χ^2-DRO (Hashimoto et al., 2018a) that considers the α-fraction subpopulations or the χ^2-divergence ball of the training distribution, the large-scale method fastDRO (Levy et al., 2020) and group DRO (Oren et al., 2019; Sagawa et al., 2020a) that optimizes the worst expected loss over a set of groups in the training data. Group DRO leverages the data structures to construct the uncertainty set, and can prevent an overly pessimistic objective. Our works in Part II propose optimization methods based on group DRO by identifying some of its underlying problems and applying our improved version to the real world NLP applications.

2.2.4 Causal Methods

Causal inference (Feder et al., 2021) has also been utilized to examine and improve robustness of NLP systems especially for preventing the model from learning spurious correlations in the data (§2.1.2). Some evaluation methods (Ribeiro et al., 2020; Gardner et al., 2020) have been developed to ensure that predictors are not "right for the wrong reasons", which generally take two forms: *invariance test*s that assess whether predictions change by perturbations that are causally unrelated to the label, and *sensitivity tests* which apply perturbations that should be the minimal change necessary to flip the label. Besides, to learn robust predictors that pass tests of sensitivity and invariance, some causal inference motivated methods have been proposed by incorporating domain knowledge of causal structure of data into the learning objective. Particularly, data augmentation methods that generate counterfactual examples via manual pos-editing (Kaushik et al., 2020; Gardner et al., 2020), heuristic keyword replacement (Shekhar et al., 2017; Garg et al., 2019) and automatic text rewriting (Zmigrod et al., 2019; Wu et al., 2021) have been shown effective in improving model's performance against dataset bias or artifacts.

2.2.5 An Overview of the Thesis on the Connections with Distribution Shift

In §2.1, we have reviewed various failure cases of NLP systems in the face of distribution shift with pointers to their connections with later chapters. In this section, we will present a holistic picture of how the works in this thesis address distribution shift under various situations. We tackle two main types of distribution shift—subpopulation shift and domain generalization (new task generalization)—under the settings of either fine-tuning a pre-trained model or learning a supervised model from scratch. In subpopulation shift, the training data usually comprises imbalanced data across different domains or groups while at test time we expect the model to perform equally well across these domains or groups. We tackle two major problems associated with subpopulation shift: inferior performance of underrepresented groups (or worst-case group) and hallucinating new content in the face of domain shift as being manifested in conditional sequence generation. We take two approaches to handle distribution shift in a conventional learning setting: (1) In Chapter 3, we first identify erroneous content in the model outputs and then filter the training data by removing harmful content that may lead to hallucination. (2) In Part II, we developed group distributionally robust optimization methods to alleviate subpopulation shift for classification and generation (machine translation) problems respectively, where the training distribution is adjusted dynamically based on the model losses across groups.

Transfer learning of pre-trained language models to downstream tasks has become the *de facto* method for various NLP tasks. This transfer learning process essentially undergoes the distribution shift from the pre-training data distribution (generally a large text training corpus) to the downstream data distribution. Loosely speaking, this can be viewed as a more extreme extension of domain generalization where the data distributions across tasks can be more divergent compared to those across domains. However, there is usually a fine-tuning stage for using the pre-trained model on downstream tasks. In Part III, we explore parameter-efficient transfer learning approaches which not only preserve the original knowledge stored in PLMs but also enable effective generalization in the low data regime. Concretely, we unify previous parameter-efficient tuning methods and propose a better version based on this unified framework. Then, we explore the use of parameter-efficient tuning method in a zero-shot task generalization setting.

Part I

On the Faithfulness of Conditional Sequence Generation

Chapter 3

Detecting Hallucinated Content in Conditional Neural Sequence Generation

Neural sequence models for tasks such as data-to-text generation (Puduppully et al., 2019), machine translation (MT; Vaswani et al. (2017); Wu et al. (2016)) and text summarization (Rothe et al., 2020) can often generate fluent text that is sometimes *preferred* to human-written content (Läubli et al., 2018; Brown et al., 2020). However, they also often generate fluent texts that lack global logical consistency (Marcus and Davis, 2020), are dull and repetitive (Welleck et al., 2019), or contain hallucinated content that is not entailed by the input (Maynez et al., 2020; Martindale et al., 2019). As we have discussed in 1, these variety of fluent but wrong outputs are particularly problematic, as it will not be possible for users to tell they are being presented incorrect content. In this chapter, we focus on tackling the latter problem, aiming to automatically identify and quantify content in the output that is not faithful to the input text.

3.1 Introduction

The risk of generating unfaithful content impedes the safe deployment of neural sequence generation models. The first step to building models that do not suffer from these failures is the assessment and identification of such hallucinated outputs. Prior work has shown that standard metrics used for text evaluation, such as BLEU scores (Papineni et al., 2002a; Post, 2018), ROUGE (Lin and Hovy, 2004) and BERTScore (Zhang et al., 2019), do not correlate well with the faithfulness of model outputs (Maynez et al., 2020; Wang and Sennrich, 2020; Tian et al., 2019). They also require reference output text, limiting their applicability in a deployed system at run-time. Very recent efforts have started to develop automatic metrics to measure the faithfulness of output sequences using external semantic models, e.g. the question-generation and question-answering systems (Wang et al., 2020a; Durmus et al., 2020) or textual entailment inference models (Maynez et al., 2020), to score faithfulness tailored for abstractive text summarization. However, these scores do not directly identify hallucinated tokens and only correlate

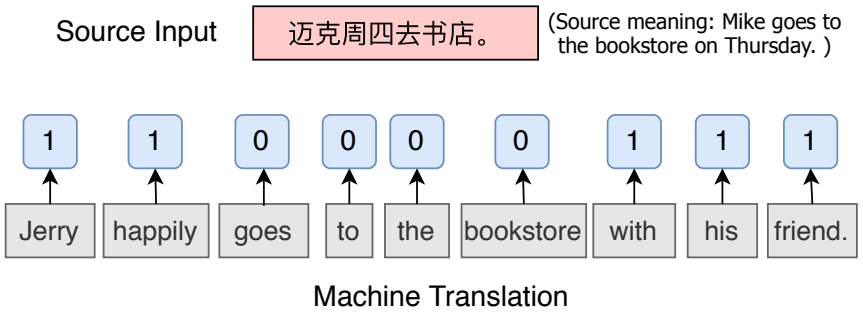


Figure 3.1: An example of token-level hallucination detection from MT. The grey box is an example of MT output and the labels above indicate if each word is faithful (0) to the input or hallucinated (1).

weakly with human judgements.

We propose a new task for faithfulness assessment - hallucination detection at the token level, which aims to predict if each token in the machine output is hallucinated or faithful to the source input. This task does not use the reference output to assess faithfulness, which offers us the ability to also apply it at run-time. Similar to the spirit of our proposed task, word-level quality estimation (Specia et al., 2018; Fonseca et al., 2019) in the MT community predicts if tokens are correctly translated based on human post-editing. However, these methods generally do not distinguish errors in terms of fluency and adequacy (Specia et al., 2011), with the exception of a subset of the WMT 2020 shared task on quality estimation (Specia et al., 2020), where different types and levels of severity of word-level errors are defined. Our proposed task specifically focuses on hallucination errors, and we define these errors in a simpler way with only binary labels, which we argue makes them simpler to use and more conducive to labeling at large scale. The proposed hallucination detection method (described below) is also applicable to the word-level quality estimation task as demonstrated in §3.5.4.

We measure hallucination for two conditional sequence generation tasks – abstractive summarization and MT. For the former, we produce a benchmark dataset from recently released annotations (Maynez et al., 2020). For MT, we carefully design human assessment guidelines and create high-quality annotations, which will be released to aid future research. To learn token-level hallucination prediction for general conditional sequence generations tasks, we propose a novel method that creates synthetic "hallucinated" data and finetunes a pretrained language model (Liu et al., 2019; Conneau et al., 2020a) on it. Without any human annotated supervised training data, we achieve an average F1 of around 0.6 across all the benchmark datasets, setting initial performance levels for this new task.

Predicting hallucination labels at the token level provides a tool for diagnosing and interpreting model outputs, which allows us to flag potential risks when the model is applied to previously unseen inputs. Additionally, we show how to use these token-level hallucination labels in two case studies to improve self-training (Scudder, 1965) and learning from noisy mined bitext (Koehn et al., 2019)

in low-resource MT. In both cases, there can be noise in the target text, either produced by the self-training teacher or errors in the mining process. However, most outputs are only partially erroneous (see examples in Appendix .5.3) and the rest of the output is still useful for training, as we show by introducing different token-level loss truncation schemes that use our proposed hallucination detection methods. Our best methods outperform strong baselines by a large margin, and reduce the number of hallucinations. Our codes and data available at https://github.com/violet-zct/fairseq-detect-hallucination.

3.2 Task: Token-level Hallucination Prediction

For source sequence S and generated output sequence G, following Maynez et al. (2020) we define any span $g_i, \cdots, g_{i+j}$ $(j >= 0)$ in G as being "hallucinated" if it is not supported by the source input S.[1] More specifically, we consider two types of hallucination, which are not mutually exclusive:

Extrinsic hallucinations: a span $g_i, \cdots, g_{i+j}$ in G consists of additional content without clear grounding in the input. In Fig. 3.1, the word *"happily"* in the machine translation belongs to this case, as there is no word in the input sentence that clearly corresponds to "happy".

Intrinsic hallucinations: a span of word(s) in G contains incorrect information due to synthesizing content using information present in S. In Fig. 3.1, *"Jerry"* in the MT is a hallucinated word and should be replaced by *"Mike"*. Note that multi-word phrases can also be marked intrinsic hallucinations, such as "this is a book" being hallucinated from "this is not a book", where "this is" is a minimal span corresponding to the hallucination.

The above definitions are for illustrative purposes; we do not explicitly label whether a hallucination is intrinsic or extrinsic, only whether one exists at all. Given these spans, we aim to identify all the span(s) satisfying the above conditions in machine generation G.[2]

Human Assessment of Hallucinations To facilitate the assessment of hallucinations in MT, we conduct human annotations on outputs of MT models in the patent and COVID-19 domain. Three bilingual annotators were presented the source sentence, the reference sentence and the MT output, and they were asked to label each sentence with one of the three types of labels: incomprehensible, faithful, and contains hallucinations. If the translation contains hallucinations, we asked the annotators to tag all the tokens that were not faithful to the source. The final benchmark datasets were created by taking majority labels among three annotators. We present more details regarding annotation guidelines and pipelines in Appendix .1.

[1] Content that is paraphrased or can otherwise be inferred by the source document is not considered hallucinated.

[2] We do not annotate under-generations e.g. the source input is only partially translated or summarized.

We compute the Fleiss's Kappa (Fleiss, 1971) (FK) scores of our annotations for MT and the processed annotations from (Maynez et al., 2020) on abstractive summarization (Tab. 1 in Appendix .1). We achieved moderate agreement (FK≈0.56) on the token-level hallucination annotations and substantial agreement (FK≈0.67) on the sentence-level annotations, while Maynez et al. (2020) achieved substantial or almost perfect agreement (FK≈0.8) on the XSUM dataset. For MT, we conjecture that it is relatively hard to achieve consistent agreement among annotators for several reasons. First, although we have made detailed annotation guidelines following the definition of hallucination in § 3.2, it could still be difficult for annotators to distinguish between ungrammatical translations and hallucinations. Second, it was sometimes difficult for annotators to understand the specialized text in the patent domain.

3.3 Token-level Hallucination Detection

We propose a general-purpose method for token-level hallucination detection for conditional sequence generation tasks. Given the source input S, we first formulate the task of token-level hallucination detection as a sequence labeling problem where a binary label is predicted at each position G_t of the machine generation G. One straightforward way of learning this task is to train a model with supervised data in the form of $((S, G), L_G)$ where L_G are the labels at every position of G that indicate if each word is a hallucinated one or not. However, because such labeled training data is not readily available, we propose an approach to automatically create synthetic training data.

3.3.1 Synthetic Data Creation

We use bi-text from the training data to create synthetic examples by automatically inserting new, hallucinated target-side tokens. More specifically, we take target sequence T and create a hallucinated version of it denoted T' with associated hallucination labels for each token in T'. Then we can train a supervised model on this synthetic labeled data set of $((S, T'), L_{T'})$. The key challenge is that T' should be a fluent sentence that does not differ too much from T.

Generation of hallucinated sentences To control this synthetic hallucination process, we build on a pre-trained denoising autoencoder, which maps a corrupted sentence back to the original text it was derived from, learning to reconstruct missing words that have been arbitrarily masked out. Specifically, we use the BART model (Lewis et al., 2020a), without providing it any access to the source sentence, thereby encouraging it to insert new content as needed to ensure fluency. As shown in Fig. 3.2, we first apply a noising function that removes words from the original target sentence T[3] and then use a pretrained BART to generate T' conditioned on the noised T with beam search.

[3]We also applied other noising functions, please see §3.5.1

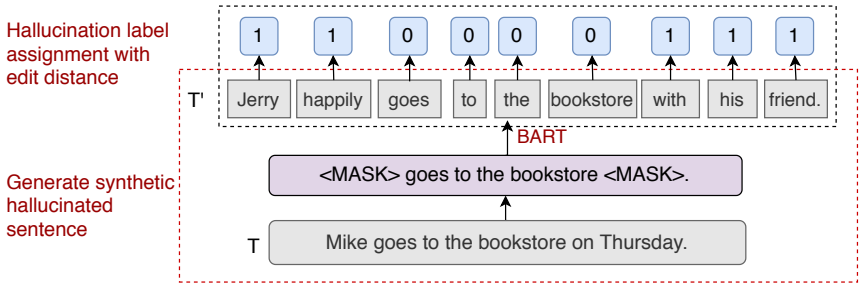


Figure 3.2: Generation of synthetic data with hallucination labels. A hallucinated version of T is generated by feeding the noised sentence to the encoder-decoder model BART. Hallucination labels are assigned to each token by computing the edit distance between T' and T. Labels of 1 refer to hallucinated words.

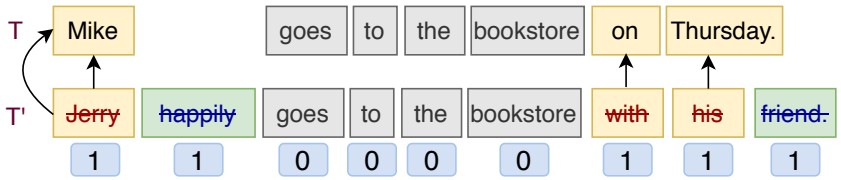


Figure 3.3: An example of label assignment.

Label assignments After obtaining the hallucinated sentence T' with BART, we need to assign appropriate labels to each token in T' to mark which words are hallucinated. We compute the edit distance between T' and T, and back-trace the deletion and substitution operations with dynamic programming. All the positions in T' involving these two operations are labeled as hallucinations and everything else is considered faithful to T. Fig. 3.3 shows an example of label assignment with edit distance, where words in red are replaced and words in blue are deleted to convert T' to T. Assigning labels with edit-distance can not always guarantee correct labels, but we find that this simple approach provides sufficiently high quality training data for effective hallucination detection in practice.

3.3.2 Finetuning on Synthetic Data

Hallucination prediction loss We follow the common practice in natural language understanding (NLU) tasks and finetune a pretrained language model (LM) on our synthetic data. We finetune a cross-lingual LM (Conneau et al., 2020a) for MT and a monolingual LM (Liu et al., 2019) for summarization. In both cases, we concatenate the input, true target and hallucinated target denoted (S, T, T') as a single input sequence to the model. Then we minimize the standard classification loss $\mathcal{L}_{pred}$ over the pseudo hallucination labels $L_{T'}$ on top of the final hidden vectors of each token in T' as shown in Fig. 3.4.

Although using only the source text and hallucinated target (S, T') as the input should be sufficient to learn to predict hallucinations, we can also easily measure the extent to which including the true target T in the input could help the model. At test time, when evaluating the faithfulness of the machine

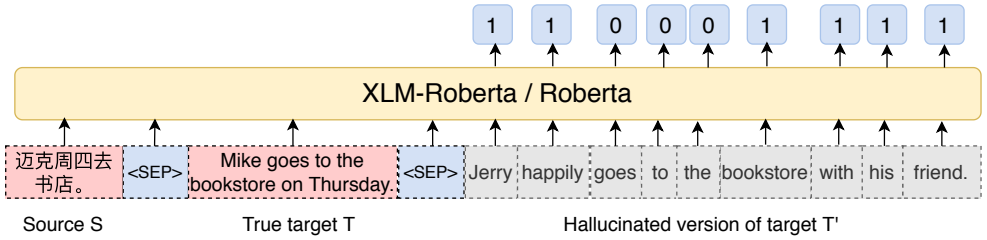


Figure 3.4: Finetuning XLM-Roberta (for cross-lingual generation task, e.g. MT) or Roberta (for monolingual generation task, e.g. text summarization) on the synthetic training data.

outputs G, we do not use the true target T and perhaps surprisingly find our model can generalize well without references, even when they were present during training.

To prevent the model from overly relying on the true target T and learning spurious correlations (e.g. the edit distance), we explored two techniques: (1) *dropout* – randomly drop out tokens in T to force the dependence on the source input; (2) *paraphrase* – recall that at synthetic data generation time, we generate T' from BART conditioned on the noised T. Instead, we can apply noise functions to the paraphrased sentence of T. We create paraphrased targets via knowledge distillation (Kim and Rush, 2016a) where we use the output from pretrained Seq2Seq model conditioned on the source sentence in the bi-text corpus as the paraphrased target. Let D denote the paraphrased sentence of T and D' denote the generation from BART conditioned on the noised D. Then we create pseudo labels of D' denoted $L_{D'}$ by computing the edit-distance between the D' and D and use $((S, T, D'), L_{D'})$ as the training data for finetuning. Since the pseudo labels are created based on D, it can prevent the model from learning the edit-distance between T and D' easily. We provide ablation studies in Appendix .4.

Masked LM loss We also add the masked language model loss (MLM) $\mathcal{L}_{mlm}$ following (Devlin et al., 2019a). To learn this loss, we create a different batch from the above by concatenating only the source S and target T as the input, since the hallucinated target T' could provide erroneous information for predicting masked words in T. We find that such multi-task learning objective helps learn better representations of the input and further improves performance on predicting hallucination labels. The final loss is $\mathcal{L} = \mathcal{L}_{pred} + \alpha \cdot \mathcal{L}_{mlm}$ where α is a hyperparameter.

3.4 Evaluation Tasks and Data

We examine hallucination in abstractive text summarization and machine translation (MT) tasks, using the models and datasets described below.

Methods	MT		Summarization			
	TranS2S	**MBART**	**PtGen**	**TConvS2S**	**TranS2S**	**BERTS2S**
Alignment	29.47	9.93	38.92	37.94	34.47	35.81
Overlap-based	9.14	3.24	57.22	54.25	53.79	55.13
Synonym-based	–	–	59.54	63.73	58.66	53.07
Ours (w/o reference)	**65.75**	**41.92**	63.66	65.94	61.70	55.45
Ours (w/o reference + synonym)	–	–	**64.72**	**69.37**	**63.88**	**56.49**
Ours (w/ reference)	66.08	46.81	63.89	66.28	62.24	55.88

Table 3.1: F1 (x100) of hallucination labels on MT (seeSection 3.4.2) and abstractive summarization (XSUM). The first block are baseline methods and the second block are our results. Bold indicates best results not using references.

Methods	MT		Summarization			
	TranS2S	**MBART**	**PtGen**	**TConvS2S**	**TranS2S**	**BERTS2S**
True hal. tokens (%)	18.12	11.10	46.09	52.89	46.74	37.51
Pred hal. tokens (%)	18.56	7.99	57.22	57.68	55.78	48.84

Table 3.2: Annotated (True) and predicted (Pred) percentage of hallucinated tokens on benchmark test sets.

3.4.1 Abstractive Text Summarization

Maynez et al. (2020) studied hallucination problems in extreme summarization on the XSUM dataset which comprises 226,711 British Broadcasting Corporation (BBC) articles paired with their single-sentence summaries. They randomly sampled 500 articles from the XSUM test set and evaluated summaries from four abstractive summarization systems: **PtGen** (See et al., 2017), **TConvS2S** (Narayan et al., 2018), **TranS2S** (Vaswani et al., 2017) and **BERTS2S** (Rothe et al., 2020). Maynez et al. (2020) asked human annotators to label the spans in the machine generated summaries if they were unfaithful to the article. We post-processed their human annotations by majority voting and created test datasets for each of the summarization systems.

3.4.2 MT

Previous work (Wang and Sennrich, 2020; Müller et al., 2019; Koehn and Knowles, 2017) has shown that translation models are particularly prone to hallucination when tested out of domain. We similarly focus on this regime and additionally consider the low resource case where a modest amount of out of domain data is available at training time.

Data We use a multi-domain Chinese-English (Zh-En) translation dataset (Wang et al., 2020e) which consists of four balanced domains: *law, news, patent* and *subtitles*. We create a new training data $\mathcal{D}_{train}$ with *law* (1.46M sentences), *news* (1.54M), *subtitles* (1.77M) train data and randomly sample 870 parallel sentences from the *patent* training data. We train two NMT models (described below) on this dataset and test on 150 examples from the patent test data. In addition, we also test the NMT models on the COVID-19 domain, sampling 100 examples from the dataset of Anastasopoulos et al. (2020). We denote this 250-sentence dataset as $\mathcal{D}_{eval}$ and ask human annotators to evaluate the level of hallucinations thereof.

Models Our data is generated from two models on which we will measure hallucination (see Appendix .2 for more details): (1) **TranS2S** (Vaswani et al., 2017) is the standard Transformer Seq2Seq model with 6 encoder layers and 6 decoder layers. (2) **MBART** (Liu et al., 2020c) is a Seq2Seq denoising auto-encoder pretrained on large-scale monolingual corpora in many languages. We finetune the 12 layer model on $\mathcal{D}_{train}$.

3.5 Experiments

3.5.1 Experimental setup

Synthetic Data Generation We use a pretrained 12 layer BART (Lewis et al., 2020a) model in the fairseq toolkit (Ott et al., 2019) for synthetic labeled data generation. We uniformly sample the percentage of tokens p_m to mask from $[0, h_m]$ for each sentence. We also uniformly sample the probability of replacing a token with a random token from $[0, h_r]$ denoted p_r. p_m and p_r are two important factors that affect the noise level when generating the synthetic data. For MT, we set h_m and h_r to 0.6 and 0.3 respectively. For abstractive summarization, we use 0.4 and 0.2. We use beam search for decoding from BART with beam size of 4 and length penalty of 3. For MT, we first create paraphrased target sentences D' through knowledge distillation (Kim and Rush, 2016a) by using the outputs from the same trained TranS2S model on the source inputs.

Hallucination Predictor For MT, we finetune XLM-R (Conneau et al., 2020a) on the synthetic dataset with batch size of 128, and we annotated 50 examples (different from those in $\mathcal{D}_{eval}$) from the patent test data as the validation dataset. For summarization, we finetune RoBERTa (Liu et al., 2019) with batch size of 96 and early stop training with 10K update steps. In addition, we dropout tokens from the reference T in the input with a rate of 0.5 and 0.3 respectively for summarization and MT to learn $\mathcal{L}_{pred}$. We set α to be 0.6 for MT and 0.5 for summarization based on the scales of $\mathcal{L}_{pred}$ and $\mathcal{L}_{mlm}$. For both tasks, we set the mask probability used for $\mathcal{L}_{mlm}$ to be 0.5, and the initial learning rate to be $2e-5$ with polynomial decay. We describe other hyperparameters, including training of MT models, in the Appendix .2 and .3.

3.5.2 Evaluation of hallucination prediction

In Tab. 3.1, we present the F1 of token-level hallucination labels across six benchmark datasets for MT and abstractive summarization (full results of precision, recall and F1 are presented in Tabs. 3 and 5 in the appendix). We compare with three baseline methods that we proposed for this new task: **(1)** The **alignment-based** method uses a word alignment model for hallucination assessment. We employ SimAlign (Sabet et al., 2020), an unsupervised aligner, that extracts word alignments from similarity matrices induced from pretrained word embeddings. SimAlign is essentially used for crosslingual tasks, and we adapt it to summarization by using embeddings from the pretrained BERT-large (Devlin et al., 2019a). We predict a target token as being hallucinated if it is not aligned to the source tokens. **(2)** The **overlap-based** method is a heuristic one that predicts a target token as being hallucinated if does not appear in the source. Since it's not feasible to perform string matching between two languages for MT, we use a bilingual lexicon induction method (Zhou et al., 2019) to first translate each English word into a Chinese word and then check its existence in the source text. **(3)** We go further by exploiting **synonyms** to assess hallucination in the summarization task where we use WordNet (Miller, 1998) to find synonyms of nouns, verbs, adjectives and adverbs of the target summary and the source article; we predict a target as being hallucinated if its synonym can not be found in the set of the source synonyms.

From Tab. 3.1, we note: **(1)** The proposed method achieves decent performance on this task and ranks the best among all baseline methods. However the task is still far from being solved is worthy of study in the future. **(2)** We can see that even though our model learns hallucination prediction with reference T during training (Sec. 3.3.2), by applying token dropout to T, our model generalizes well without feeding the reference at test time. As a contrast, we report the results of predicting with reference at test time and observe that the model can achieve a significantly higher recall but worse precision (Tab. 5 in appendix). **(3)** The two non-neural baselines we proposed work surprisingly well on the summarization datasets, especially the synonym-based system. We guess this is because the information of the summaries should come from the source article and a majority of hallucinated words are nouns (§3.5.3) which can be easily detected by string matching or synonym matching. Our neural system performs better than these baseline methods but not significantly, and we hypothesize that this is because the RoBERTa model we finetune on only allows a maximum input length of 512, which results in an average cutoff of 158 subwords from the source article and hence loss of source information. By taking the union of the predictions from the synonym-based and our models, we can further obtain improvements on the summarization datasets. We believe the advances in long sequence modeling (Beltagy et al., 2020; Kitaev et al., 2020) could help here, and are important to study in future work. **(4)** At the same time, the baseline methods can not obtain reasonable performance for MT since crosslingual semantic matching is more challenging and our model shows significant improvements.

In Tab. 3.2, we show the percentage of annotated and model predicted hallucinated tokens across the

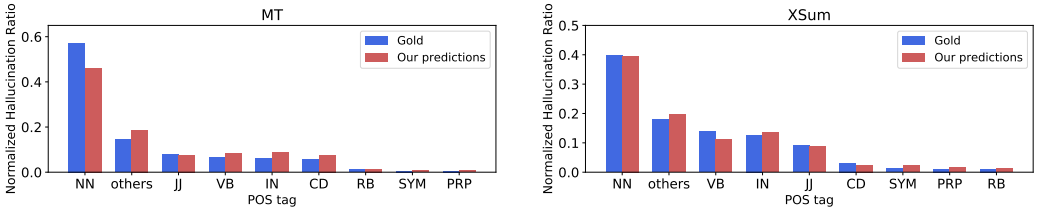


Figure 3.5: Relationship of POS tags and percentage of hallucinations for MT (left) and summarization (right).

six benchmark sets. We can see that model predictions correlate well with human assessment and have a Pearson correlation coefficient of 0.986.

3.5.3 Analysis

Analysis on Pretrained Models for Conditional Sequence Generation Recent work (Maynez et al., 2020) has shown that pretrained models are better at generating faithful summaries as evaluated by humans. In Tab. 3.2, summaries generated from BERTS2S contain significantly fewer hallucinations than other model outputs. We also confirmed this trend in MT that translations from MBART contain less hallucinated content than that from TranS2S.

Analysis on Hallucinated Words and their Part-of-Speech Tags In Fig. 3.5, we present the percentage of hallucinated tokens categorized by their part-of-speech tags predicted by a POS tagger of Toutanova et al. (2003). First, we see that for both MT and summarization datasets, nouns are the most hallucinated words. In abstractive summarization, verbs also account for a certain number of hallucinations. Second, our model predicted hallucinated words match well with gold annotations on the distributions of POS tags. We also compare the percentage of hallucinations within each POS tag in Appendix .5.2. In addition, we provide more ablation studies in Appendix .4.

3.5.4 Evaluation on Word-level Quality Estimation

As noted in §3.1, our model is also readily applicable to word-level quality estimation (QE) for MT (Fonseca et al., 2019; Specia et al., 2020), which aims to detect word-level errors in MT output. In the WMT shared task of word-level QE, each token of the target sentence is labeled as OK/BAD based on the post-edited target sentences. We evaluate our model on the WMT18 en-de word-level QE shared task (Specia et al., 2018) in both the unsupervised and supervised setting. There are 13,442 labeled parallel sentences where the tagged target sentences are from an NMT model. In our supervised setting, we finetune the XLM-R model on these parallel sentences with the objective: $\mathcal{L}_{pred} + 0.5 * \mathcal{L}_{mlm}$. In the unsupervised setting, we first create the synthetic data (§3.3.1) using the post-edited target sentences from the labeled parallel set (13,442) and an additional 50K target sentences from the provided unlabeled parallel set.

Models	BAD-F1	OK-F1	F1-MULT
OpenKiwi	-	-	44.77
1^{st} place in WMT18	48.00	91.00	44.00
3^{rd} place in WMT18	36.00	85.00	30.00
Ours (unsupervised)	37.09	**92.54**	34.32
Ours (supervised)	**50.78**	91.91	**46.68**

Table 3.3: F1 scores (x100) on the test set of WMT18 word-level QE. OpenKiwi (Kepler et al., 2019) is the state-of-the-art result on this task. 1^{st} and 3^{rd} place are results from the shared task (Specia et al., 2018).

Then we finetune XLM-R on the created synthetic labeled data. For both settings, we set the weights of the cross-entropy loss for the bad-token labels to be 2.0 because the labels are imbalanced with fewer bad-token labels.

Results We present results in Tab. 3.3, where F1-Mult is the multiplication of F1-scores for the OK and BAD labels. Note that all the baseline models are in the supervised setup and the best baseline OpenKiwi (Kepler et al., 2019) is a strong ensembled system using predictions from multiple models. In contrast, our supervised model only leverages the parallel labeled data without using other resources. Among all the supervised settings, our model outperforms the best system by 2 points in F1-Mult. To make it clear how our unsupervised model performs, we also show the best performed systems in the shared task of WMT18. We observe that our unsupervised setting achieves descent performance and even outperforms the 3^{rd}-ranked system. These results demonstrate that both the full version and the finetuning part of our method provide strong results for word-level QE.

3.6 Case Study I: Improving Self Training in Machine Translation

Predicting hallucination labels at token-level not only allows us to flag potential risks in generation models, but also opens up the possibility of providing fine-grained signals which can be used to define new learning objectives. In this section and the following one, we demonstrate how to leverage the hallucination labels to reduce adverse effects of noisy training instances. Specifically, we show that the fine-grained hallucination signals allow for improved semi-supervised learning (§3.6) and training with noisy parallel data (§3.7).

3.6.1 Rectified Self-Training for Neural MT

Self training (Scudder, 1965) is an important semi-supervised approach that utilizes unlabeled source data to improve system performance. In a conditional sequence generation task, a teacher model is first

Methods	BLEU (↑)	BLEURT (↑)	Hal words (%, ↓)
baseline	16.14	-0.166	13.69
ST	19.31	-0.059	10.00
ST + paraphrase noise	19.05	-0.051	13.48
ST + random noise	19.97	-0.041	12.55
ST + seq loss truncation	19.91	-0.048	8.26
ST + random noise + seq loss truncation	19.37	-0.057	10.06
ST + token loss truncation	20.32	0.00244	**6.37**
ST + decoder HS masking	20.57	-0.0001	6.38
ST + random noise + token loss truncation	**21.02**	**0.043**	7.34
ST + random noise + decoder HS masking	20.64	0.0308	8.70

Table 3.4: BLEU(↑), BLEURT(↑) and hallucinated tokens (↓) on the CWMT2017 test set. We compare with the noised self-training method (He et al., 2020) in the second block and sequence-level loss truncation method (Kang and Hashimoto, 2020) in the third block.

trained with bitext $\mathcal{D}_l = \{\mathbf{s}_i, \mathbf{t}_i\}_{i=1}^{N}$ and used to make predictions on each sequence in a unlabeled dataset $\mathcal{D}_u = \{\mathbf{s}_j\}_{j=N+1}^{N+M}$ to create *pseudo parallel data* $\mathcal{D}_p = \{\mathbf{s}_j, \mathbf{t'}_j\}_{j=N+1}^{N+M}$. The model is then trained on $\mathcal{D}_l \cup \mathcal{D}_p$. He et al. (2020) finds that with self-training the student model can benefit from such pseudo-parallel data. However, such results require a relatively high-quality teacher, and performance suffers in low-resource setting where no such teacher is available.

We propose to use our token-level hallucination predictions to define a fine-grained loss during training in MT, by penalizing errors less on tokens that more likely to be hallucinated. This is in contrast to previous data filtering methods for MT, which remove entire sentence pairs (Junczys-Dowmunt, 2018; Kang and Hashimoto, 2020).

First, we predict the token-level hallucination labels on the target side of the pseudo parallel data $\mathcal{D}_p$. Then we propose two simple methods of using these labels in self-training: (1) We discard the losses of tokens that are predicted as hallucinations and compute the loss on the remaining tokens for each target sequence (**token loss truncation**). (2) Instead of adjusting losses, we mask the decoder hidden states of those hallucinated positions after the target-to-source cross attention in each decoder layer (**decoder HS masking**).[4]

[4]We also tried removing hallucinated target words before training. This underperformed, likely because it produces too many ungrammatical target sentences.

3.6.2 Experimental Setup and Results

Experimental Setup To train a teacher model (baseline in Tab. 3.4), we use the same training data described in §3.4.2 using *patent* (870) as the low-resource domain. We evaluate on the full patent test set (1,500) from CWMT2017 (Wang et al., 2020e). For the unlabeled data, we use the withheld Chinese patent training data (2.9M). Next, we employ our hallucination detection system to predict hallucination labels on the pseudo target sentences given the source sentences.

Baselines We compare with the state-of-the-art self-training (ST) method of He et al. (2020), which injects two types of noise into the input sentences: (1) paraphrase noise created by round-trip translations, and (2) random noise from dropping, masking and shuffling input tokens. We also compare with the recently proposed loss truncation method (Kang and Hashimoto, 2020) that adaptively removes entire examples with high log loss, which was shown to reduce hallucinations.

Results and Analysis We present the tokenized BLEU score (Papineni et al., 2002a), BLEURT score (Sellam et al., 2020) and the percentage of hallucinated tokens predicted by our system in Tab. 3.4. We can see that ST improves over the baseline by around 3 BLEU and our best result further improves ST by 1.7 BLEU. Compared with strong baseline methods, our method not only achieves the best translation quality measured by BLEU and BLEURT but also the largest hallucination reduction. We also observe that: (1) Our method with ST alone can outperform other baseline methods, when combined with perturbed ST (noise), and using fine-grained control over the target tokens can further improve the results. (2) ST with paraphrase noise (by round-trip translation) does not perform as well as the random noise, which further confirms that the noisy outputs from a teacher model may hurt the student model. (3) The sequence-level loss truncation approach can improve over the vanilla ST and reduce the level of hallucinations as measured by our system. However, the performance drops when combined with the noised ST.

3.7 Case Study II: Improving Corpus Filtering for Low-Resource MT

High-quality parallel data is critical for training effective neural MT systems, but acquiring it can be expensive and time-consuming. Many systems instead use mined and filtered parallel data to train NMT models (Junczys-Dowmunt, 2018; Zhang et al., 2020a; Koehn et al., 2019). Nonetheless, the selected parallel data can still be noisy, containing misaligned segments. In this section, we demonstrate that token-level hallucination labels can allow us to make better use of noisy data to and improve the overall translation quality. We apply the token loss truncation method proposed in §3.6 to the filtered parallel data and evaluate it on the WMT2019 low-resource parallel corpus filtering shared task.

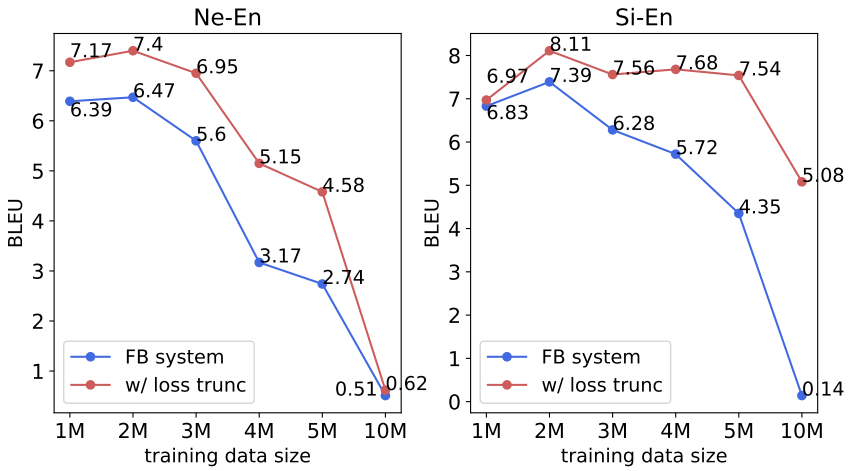


Figure 3.6: The BLEU scores of the best submission (FB system) in the WMT19 shared task on parallel noisy corpus filtering and our method (w/ loss trunc) on the Ne-En and Si-En flores test sets.

Experimental Setup The WMT19 shared task focuses on two low-resource languages – Nepali and Sinhala. It released a very noisy 40.6 million-word (English token count) Nepali-English and a 59.6 million-word Sinhala-English corpus crawled from the web. Participants were asked to score each sentence pair in the noisy parallel set. Scores were used to subsample sentence pairs amounting to 1 million and 5 million English words, which were used to train an MT system that was evaluated on the test set using SacreBLEU (Post, 2018). In addition, the shared task also provides additional clean parallel data for Nepali-English (564K), Sinhala-English (646K) and Hindi-English (1.6M), but they can not be used for training the final NMT system.

First, we train a token-level hallucination prediction system with the combined parallel data from all the three language pairs (as Hindi is related to Nepali). Second, we use the scores (Chaudhary et al., 2019) that achieve the best overall performance for both language pairs among all the submissions to select the top-scored 1M, 2M, 3M, 4M, 5M, and 10M data (in English tokens) and predict the token-level hallucination labels on the target side. We follow the same setup and use the script provided by the shared task to train the NMT model with the selected subsets. During training, we discard losses of tokens that are predicted as hallucinations and only compute the losses for the remaining tokens. We use the validation and test data from the flores dataset (Guzmán et al., 2019) during training and evaluation.

Results and Analysis In Fig. 3.6, we present the BLEU of the best submission (FB system) and our method on the Ne-En and Si-En test sets of the flores dataset. First, with token-level loss truncation, our model achieves the new best results on the flores test set in this shared task for both Ne-En (7.4) and Si-En (8.11). Second, for both language pairs our method further improves the state-of-the-art system when varying the training data sizes. Notably, in the extreme case of 10M training data, which is very noisy, the baseline can not obtain decent BLEU scores for Si-En while our method still achieves reason-

able performance (0.14 vs. 5.18). However, for Ne-En data sizes after 2M causes performance of both the baseline and our method to drop significantly, possibly because the dataset contains many pairs of misaligned sentences (the source is not Nepali and the target is not English).

3.8 Conclusion

This work proposed a new task of token-level hallucination detection, created human-annotated benchmark datasets, proposed a method for unsupervised learning of hallucination detectors, and showed that the models can be used to define fine grained losses that improve MT training. We demonstrate the remark performance of the proposed hallucination detection method in several downstream tasks, including word-level quality estimation and noisy neural machine translation. In the future, we hope to create a large-scale pretrained hallucination detector for any dataset or model, and also would extend our method to data-to-text generation scenarios. We are also interested in investigating how to leverage our detection methods to mitigate hallucination problems in conditional sequence generation.

Part II

Group Distributionally Robust Optimization

Chapter 4

Examining and Combating Spurious Features under Distribution Shift

A central goal of machine learning is to learn robust representations that capture the causal relationship between inputs features and output labels. However, as discussed in 1, the prevalent distribution shift in the real world can significantly degrade the accuracy of standard machine learning models, resulting in poor performance on test distributions that differ from the training one. In Part II, we aim to mitigate this problem by proposing group distributionally robust optimization (group DRO) methods that improve worst group performance at training time. First, in §4, we define and analyze robust and spurious representations using the information-theoretic concept of *minimal sufficient statistics*. Further, we demonstrate that group DRO can fail when groups do not directly account for various spurious correlations that occur in the data and we propose to minimize the worst-case losses over a more flexible set of distributions to alleviate this. Next, in §5, we propose efficient group DRO algorithms for multilingual machine translation to promote uniform performance across high-resource and low-resource languages, which is a large-scale and real-world problem in NLP.

4.1 Introduction

Many machine learning models that minimize the average training loss via empirical risk minimization (ERM) are trained and evaluated on randomly shuffled and split training and test sets. However, such in-distribution learning setups can hide critical issues: models that achieve high accuracy on average often underperform when the test distribution drifts away from the training one (Hashimoto et al., 2018a; Koenecke et al., 2020; Koh et al., 2020). Such models are often "right for the wrong reasons" due to reliance on *spurious correlations* (or "*dataset biases*") (Torralba and Efros, 2011; Goyal et al., 2017; McCoy et al., 2019; Gururangan et al., 2018), heuristics that hold for most training examples but are not inherent to the task of interest, such as strong associations between the presence of green pastures background

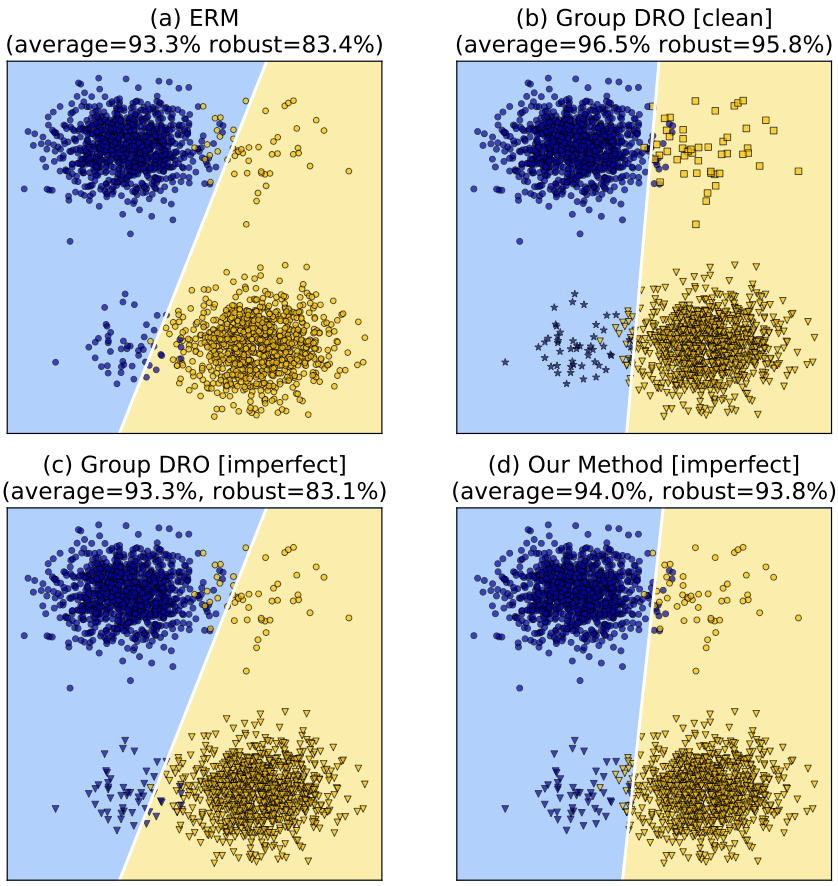


Figure 4.1: Consider data points x in $\mathbb{R}^2$ with two classes y. The vertical axis of x is a spurious feature that highly correlates with y, and the horizontal axis is the robust feature. There are two subclasses in each class, where the top-right and lower-left are two minority subclasses. The robust accuracy is test worst-case accuracy over the four subclasses. We train a linear classifier with different methods. For models trained with the clean partitions, each subclass is a group. For the imperfect partitions, dots with the same shape is a group (best viewed in color).

with the label "cows" in image classification. Naturally, models that use such features will fail when tested on data where the correlation does not hold.

Recent work has investigated how models trained with ERM learn spurious features that do not generalize, from the points of view of causality Arjovsky et al. (2019), understanding model overparameterization (Sagawa et al., 2020b) and information theory (Lovering et al., 2021). However, these works have not characterized the idea of spurious features mathematically. In this chapter, we characterize spurious features from an information-theoretic perspective. We consider prediction of target random variable $Y \in \mathcal{Y}$ from input variable $X \in \mathcal{X}$ and characterize spurious features learned under changes to the input distribution $p(X)$ (i.e. covariate shift).

A central goal of machine learning is to learn true causal relationships between X and Y in a manner

robust to spurious factors concerning the variables. We assume that there exists an *"ideal"* data distribution p_{ideal} (short for $p_{\text{ideal}}(X, Y)$ below) which contains data from all possible experimental conditions concerning the confounders that cause spurious correlations, both observable and hypothetical (Lewis, 2013; Arjovsky et al., 2019; Bellot and van der Schaar, 2020). For example, consider the problem of classifying images of cows and camels (Beery et al., 2018). Under the ideal conditions, we assume that pictures of cows and camels on any background can be collected, including cows in deserts and camels in green pastures. Therefore, under p_{ideal} the background of the image X is no longer a spurious factor of the label Y. However, such an "ideal" distribution p_{ideal} is not accessible in practice (Bahng et al., 2020; Koh et al., 2020; McCoy et al., 2019), and our training distribution p_{train} (often, in practice, an associated empirical distribution) does not match p_{ideal}. ERM-based learning algorithms indiscriminately fit all correlations found in p_{train}, including spurious correlations based on confounders (Tenenbaum, 2018; Lopez-Paz, 2016).

To investigate the spurious features learned under the distribution shift from p_{ideal} to p_{train}, we first characterize those features of X which most efficiently capture all possible information needed to predict Y. We define these *robust features* using the notion of *minimal sufficient statistic (MSS)* (Dynkin, 2000; Cvitkovic and Koliander, 2019) under p_{ideal}. We then examine *whether the features learned under p_{train} contain spurious features compared to the MSS learned under p_{ideal}*. Through our analysis, we find that even only with covariate shift, the features learned on p_{train} can contain spurious features or miss robust features of p_{ideal}.

Models that fit spurious correlations in p_{train} can be vulnerable to groups (subpopulations of $p_{\text{ideal}}/p_{\text{test}}$) where the correlation does not hold. A common approach to avoid learning a model that suffers high worst group errors is group distributionally robust optimization (group DRO), a training procedure that efficiently minimizes the worst expected loss over a set of groups in the training data (Oren et al., 2019; Sagawa et al., 2020a). The partition of groups can be defined in several ways, such as by presence of manually identified potentially spurious features (Sagawa et al., 2020a), data domains (Koh et al., 2020), or topics of text (Oren et al., 2019). In a typical setup, the groups of interest in the test set align with those used to partition the training data. Under such setups, group DRO usually outperforms ERM with respect to the worst-group accuracy. We contend that this is because it promotes learning robust features that perform uniformly well across all groups. However, in many tasks, we can not collect clean group membership of training examples due to expensive annotation cost or privacy concerns, e.g. demographic identities of users or other sensitive information.

Inspired by our analysis of spurious features, we demonstrate that group DRO can fail under "imperfect" partitions of training data that are not consistent with the test set, especially when reducing spurious correlation in one group could exacerbate the spurious correlations in another (§4.4.2), as shown in Fig. 4.1. This is because group DRO treats each training group as a unit, preventing it from adjusting learning weights differently for subgroups within each group. Recent work has pro-

posed to use sophisticated unsupervised clustering algorithm to search for meaningful subclasses (Sohoni et al., 2020) and execute group DRO on the found subclasses. To learn robust models under noisy protected groups, Wang et al. (2020b) designs robust approaches that is based on an estimate of a noise model between the clean and noisy groups. Instead of relying on good partitions of groups or a not readily available noise model, we propose group-conditional DRO (GC-DRO) that defines the uncertainty set over the joint distribution of groups and their instances (i.e. $q(G)q(X, Y|G)$). Every training example is reweighted by both its group weight and the instance-level weight, which offers a more flexible uncertainty set compared to group DRO. Through extensive experiments on three tasks — facial attribute classification, natural language inference, and toxicity detection, we show that GC-DRO significantly outperforms both ERM and group DRO in various partitions of training data and demonstrate the robustness of GC-DRO against various group partitions. Our code is available at `https://github.com/violet-zct/group-conditional-DRO`.

4.2 Preliminaries on Robust Representations

To study spurious features, we need to formally define which features or properties of the data describe spurious correlations, and which features are robust features relevant to the task at hand. In supervised learning we are interested in finding a good representation $T(X)$ of the input X[1] that is useful to predict a target label Y. What characterizes the optimal representations of X w.r.t. Y is much debated, but a common assertion is that $T(X)$ should be a *minimal sufficient statistic* (MSS) of X for Y (Adragni and Cook, 2009; Schwartz-Ziv and Tishby, 2017; Achille and Soatto, 2018; Cvitkovic and Koliander, 2019), which is:

(i) $T(X)$ should be *sufficient* for Y, i.e. $\forall x \in \mathcal{X}, t \in \mathcal{T}, y \in \mathcal{Y}, p(x|t, y) = p(x|t)$, which is equivalent to $p(y|t, x) = p(y|t)$. This means given the value of $T(X)$, the distribution of X does not depend on the value of Y.

(ii) Given that $T(X)$ is sufficient, it should be *minimal* w.r.t. X, i.e. for any sufficient statistic S, there exists a deterministic function f such that $T = f(S)$ almost everywhere w.r.t. X. This means for any measurable, non-invertible function g, $g(T)$ is no longer sufficient for Y.

In other words, the minimal sufficient statistics most efficiently capture all information useful for predicting Y. The notion of MSS has been connected to Shannon's information theory (Kullback and Leibler, 1951; Cover, 1999) and extended to any joint distribution $P(X, Y)$ of X and Y in the information bottleneck (IB) framework (Tishby et al., 2000; Shamir et al., 2010; Kolchinsky et al., 2019), which provides a principled way to characterize the extraction of relevant information from X for predicting Y. Loosely speaking, learning a MSS T is equivalent to maximizing $I(T(X); Y)$ (sufficiency) and minimizing $I(X; T(X))$ (minimality).

[1] We assume that $T(X)$ is a deterministic mapping of X given neural network parameters.

Robust Features. Suppose $\mathcal{A}$ contains all possible combinations of spurious variables, both observable and hypothetical, and we consider datasets $\mathcal{D}_{(a,y)} = \{x_i\}_{i=1}^{N_{a,y}}$ collected under each condition of $(a \in \mathcal{A}, y \in \mathcal{Y})$, where each $\mathcal{D}_{(a,y)}$ contains examples that are *i.i.d.* according to some probability distribution $p(X|y,a)$. We define p_{ideal} as the mixture distribution of $p(X|y,a)$ with uniform weights over $(a,y) \in \mathcal{A}\times\mathcal{Y}$. Thus, MSS learned on p_{ideal} provide a good candidate for *robust features* $T(X)$ (sometimes denoted $T_{ideal}(X)$ for clarity), which most efficiently capture the information from X necessary for predicting Y on a distribution that is free of spurious factors.

Spurious Features. In contrast, we define representations $T'(X)$ that contain spurious features. Specifically, the entropy of $T'(X)$ conditioned on $T(X)$ under p_{ideal} is positive.

$$H_{ideal}(T'(X)|T(X)) > 0 \tag{4.1}$$

Because these learned features are not deterministic given $T(X)$ then they contain *additional* information that is not useful for predicting Y.[2] For example, in image classification, knowing that the image contains a horse, we cannot predict the background with certainty (a horse could be on a race track or a beach). Another example in natural language inference (NLI) task is that model learned on a biased data set often associates negation with the label "contradiction", which is another spurious feature under our definition. Because given the meaning of a sentence (robust features), whether it contains negation or not is not deterministic, e.g. "Don't worry." and "Be calm." are synonymous but only one contains negation. A classifier that uses these spurious features can suffer from the risk of learning the *spurious correlations* between $T'(X)$ and the labels Y.

4.3 Spurious Features under Covariate Shift

The training data is often marred by various abnormalities, such as selection biases (Buolamwini and Gebru, 2018) and confounding factors (Gururangan et al., 2018). We ask *if the MSS learned under p_{train} are robust features under p_{ideal}*. Note that we do not study how to learn MSS via ERM in this chapter, on the other hand, considering that MSS provides a good candidate for robust representations, we want to study if the MSS learned under p_{train} contains spurious features with respect to the MSS learned under p_{ideal}, which are universal robust features against various spurious factors.

We consider the distribution shift in $p(X)$,[3] also known as covariate shift (David et al., 2010), and we show that the entropy of MSS learned under p_{train} conditioned on the robust features is zero in Theorem 1 with proofs in §.6.

Theorem 1. *Suppose that there is only covariate shift in p_{train}, i.e. $\exists x \in \mathcal{X}_{train}$ s.t. $p_{train}(x) \neq p_{ideal}(x)$ but $p_{train}(Y|X=x) = p_{ideal}(Y|X=x)$, $\forall x \in \mathcal{X}_{train}$. Let $T_{train}(X)$ be the MSS representation learned*

[2] Note that it is not just the case of $T'(X)$ containing redundant features, in which case $H(T'(X)|T(X)) = 0$.

[3] It is often assumed that $p(Y|X)$ is invariant in supervised learning problems (Arjovsky et al., 2019).

under p_{train}, then we have:

$$H_{train}(T_{train}(x)|T_{ideal}(x)) = 0. \tag{4.2}$$

Theorem 1 tells us that $T_{train}(X)$ is deterministic given $T_{ideal}(X)$ under p_{train} (shown in blue to distinguish from Eq. 4.1). However, this does not imply $H_{ideal}(T_{train}(X)|T_{ideal}(X)) = 0$ under p_{ideal}. Thus, we *cannot* conclude that $T_{train}(X)$ contains no spurious features. We further discuss the implications with two cases based on the relationship between the support of input $\mathcal{X}_{train}$ and that of $\mathcal{X}_{ideal}$: (1) $\mathcal{X}_{train} = \mathcal{X}_{ideal}$ and (2) $\mathcal{X}_{train} \subset \mathcal{X}_{ideal}$. When the input support of p_{train} is equal to that of p_{ideal}, we have the following corollary:

Corollary 1. *Suppose $\mathcal{X}_{train} = \mathcal{X}_{ideal}$ in Theorem 1, then $T_{train}(X)$ is also the MSS under p_{ideal}.*

Corollary 1 corroborates the findings in Wen et al. (2014) that the (unweighted) solution learned by ERM is also the robust solution when only covariate shift exists and $\mathcal{X}_{train} = \mathcal{X}_{ideal}$. In practice, however, this assumption does not hold (because we only have datasets with limited support) and thus the representation $T_{train}(X)$ learned by ERM is not necessarily equivalent to $T_{ideal}(X)$. By Theorem 1, $T_{train}(X)$ is deterministic given $T_{ideal}(X)$ under p_{train}, which implies that the information contained in $T_{train}(X)$ is equal to or less than that contained in $T_{ideal}(X)$. In the former case, $T_{train}(X)$ can be equivalent in representation to $T_{ideal}(X)$ but can also contain spurious features that co-occur with the robust features in the training data. In the latter case, $T_{train}(X)$ can miss robust features in $T_{ideal}(X)$. We demonstrate these two cases with synthetic experiments in Appendix .9 due to space limit.

Discussion. We have discussed the cases of learning spurious features when the model learns MSS under p_{train}. However, we normally adopt maximum likelihood estimation (MLE) as an instantiation of ERM for classification problems. We provide the connection of MLE with learning MSS via the information bottleneck method (Tishby et al., 2000; Shamir et al., 2010) in the Appendix .7, where under certain assumptions, we can view MLE as an objective that approximately learns MSS.

4.4 Does Group DRO Learn Robust Features?

The discussions in §4.3 suggest that under covariate shift, directly learning from the empirical data distribution could result in learning the spurious correlations satisfied by the majority of the training data. When the spurious factors are known, we can apply group distributionally robust optimization (group DRO), which reweights the losses of different groups associated with spurious factors to alleviate covariate shift and learn robust features that generalize to both minority and majority groups. In this section, we first review group DRO and discuss under which cases it can fail.

4.4.1 Group Distributionally Robust Optimization

Group DRO is an instance of distributionally robust optimization (Ben-Tal et al., 2013a; Duchi et al., 2016) that minimizes the worst expected loss over a set of potential test distributions $\mathcal{Q}$ (the uncertainty set):

$$\mathcal{L}_{\text{DRO}}(\theta) = \sup_{q \in \mathcal{Q}} \mathbb{E}_{(x,y)\sim q}\big[\ell(x, y; \theta)\big] \tag{4.3}$$

This worst-case objective upper bounds the test risk for all $q_{\text{test}} \in \mathcal{Q}$, which is useful for learning under train-test distribution shift. However, its success crucially depends on choosing an adequate uncertainty set that encodes the possible test distributions of interest. Choosing a general family of distribution as the uncertainty set, such as a divergence ball around the training distribution (Ben-Tal et al., 2013a; Hu and Hong, 2013; Gao and Kleywegt, 2016), encompasses a wide set of distribution shifts, but can also lead to a conservative objective emphasizing implausible worst-case distributions (Duchi et al., 2019; Oren et al., 2019).

To construct a viable uncertainty set, one can optimize models over all meaningful subpopulations or groups g depending on the available source information regarding the data, such as domains, demographics, topics, etc. Group DRO (Hu et al., 2018; Oren et al., 2019) leverages such structural information and constructs the uncertainty set as any mixture of these groups. Following Oren et al. (2019), we adopt the conditional value at risk (CVaR) which is a type of distributionally robust risk to achieve low losses on all α-fraction subpopulations (Rockafellar et al., 2000) of the training distribution (i.e. $\{p : \alpha p(x) \leq p_{\text{train}}(x), \forall x\}$). As we assume that each data point comes from some group $p(x, y|g)$ and p_{train} is a mixture of m groups $p_{\text{train}}(g)$, we can extend the definition of CVaR to groups and construct the uncertainty set $\mathcal{Q}$ as all group distributions that are α-*covered* by $p_{\text{train}}(g)$ (or *topic CVaR* (Oren et al., 2019)):

$$\mathcal{Q} = \left\{ q : q(g) \leq \frac{p_{\text{train}}(g)}{\alpha} \;\; \forall g \right\} \tag{4.4}$$

This upper bounds the group distribution within the uncertainty set by its corresponding training distribution. The group DRO objective then minimizes the expected loss under the worst-case group distribution:

$$\mathcal{L}_{\text{GDRO}} = \sup_{q \in \mathcal{Q}} \mathbb{E}_{g\sim q} \mathbb{E}_{(x,y)\sim p(x,y|g)} \left[\ell(x, y; \theta)\right] \tag{4.5}$$

Intuitively, this objective encourages uniform losses across different groups, which allows us to learn a model that is robust to group shifts. We adopt the efficient online greedy algorithm developed in Oren et al. (2019) to update the model parameters θ and the worst-case distribution q in an interleaved manner. The greedy algorithm roughly amounts to upweighting the sample losses by $\frac{1}{\alpha}$ which belong to the α-fraction of groups that have the worst losses. We present the detailed algorithm in Appendix .8.

	G_1		G_2	
	$S=0$	$S=1$	$S=0$	$S=1$
$P(Y=0\|S)$	0.5	0	1	0.5
$P(Y=1\|S)$	0.5	1	0	0.5

Table 4.1: An example of imperfect partition.

4.4.2 Group DRO Can Fail with Imperfect Partitions

As discussed earlier, we aim to learn a model that is robust to spurious factors. For example, in toxicity detection, a robust model should perform equally well on data from different demographic groups. Group DRO mitigates covariate shift by minimizing the worst-case loss under the uncertainty set $\mathcal{Q}$, consisting of mixtures of sub-group distributions. Intuitively, given that optimizing p_{ideal} allows for learning of robust, non-spurious features, defining a $\mathcal{Q}$ that covers p_{ideal} is highly advantageous from a learning perspective.

If we know all the spurious attributes of the training data $\mathcal{A}$, we can adopt the setup in Sagawa et al. (2020a) that divides the data into $|\mathcal{A}| \times |\mathcal{Y}|$ groups, where each example belongs to one of the groups $g = (a, y)$. We define such grouping strategy as "clean partitions" in which each group is uniquely associated with one value of (a, y).[4] If $\mathcal{A}$ contains all the spurious factors of interest, it can be seen that there exists some mixture of groups $\sum_{g=1}^{m} q(g) p_{\text{train}}(\cdot \mid g)$ that can recover p_{ideal}, where $q \in \Delta_m$ and Δ_m is the $(m-1)$-dimensional probability simplex. Thus, p_{ideal} is contained in $\mathcal{Q}$. Such clean partitions provide a plausible environment for group DRO to learn well in the presence of covariate shift that causes spurious correlations in the training data.

In contrast, we define "imperfect partitions" where each group contains samples from multiple values of (a, y) such that there does not exist a $q \in \Delta_m$ that recovers p_{ideal}, in other words, $\mathcal{Q}$ does not include p_{ideal}. In this case, group DRO can not eliminate covariate shift effectively.

To illustrate, consider a binary random variable $S \in \{0, 1\}$ following a uniform distribution, and the target label $Y \in \{0, 1\}$ also follows a uniform distribution and is independent of S. Due to covariate shift, there are spurious correlations between $S = 0, Y = 0$ and between $S = 1, Y = 1$ in the training data. We partition the training data into two groups with an equal number of samples and the conditional distribution of $P(Y|S)$ is shown in Tab. 4.1. To prevent the model from learning the spurious correlations between $S = 1$ and $Y = 1$, one can upweight losses of its "negative" samples for which the spurious correlation does not hold, i.e. samples of $(S = 1, Y = 0)$ in G_2; however, group DRO upweights the group as a whole, which inevitably also upweights the $(S = 0, Y = 0)$ and causes the model to latch on the spurious attribute $S = 0$ to predict $Y = 0$. Therefore, there does not exist a mixture distribution of

[4] Our discussions also apply to multiple spurious attributes for which the clean partition corresponds to $|\mathcal{Y}| \times \prod_i |\mathcal{A}_i|$ groups.

Algorithm 1: Online greedy algorithm for GC-DRO.

Input: α; β; m: #groups; n_i: #samples of group i

Initialize historical average group losses $\hat{L}^{(0)}$, historical estimate of group probabilities $\hat{p}^{tr(0)}$, historical average instance losses $\hat{L}_g^{(0)}$ and $q^{(0)}(x, y|g) = \mathbf{1}^T$ for $g \in \{1, \cdots, m\}$

for $t = 1, \cdots, T$ **do**

Sample a mini-batch $(\mathbf{x}, \mathbf{y}, \mathbf{g})$ from P_{train}

Perform online greedy updates for $q^{(t)}(Alg.3)$

▷ *Update model parameters* θ

$d_i = \frac{n_i q^{(t)}(\mathbf{g}_i) q^{(t)}(x,y|\mathbf{g}_i)}{\hat{p}^{train(t)}(\mathbf{g}_i)} \nabla \ell(\mathbf{x}_i, \mathbf{y}_i; \theta^{(t-1)})$

$\theta^{(t)} = \theta^{(t-1)} - \frac{\eta}{|B|} \sum_{i=1}^{|B|} d_i$

if *reached inner update criterion* **then**

▷ *Update* $q^{(t)}(x, y|g)$

for $g = 1, \cdots, m$ **do**

Sort instances in group g in the decreasing order of $\ell(x, y; \theta^t)$; denote the sorted index $\boldsymbol{\pi}^g$

$\text{cutoff} = \left\lceil \frac{(N-n_i)n_i\beta}{N-n_i} \right\rceil$

$q^{(t)}((x, y)_{\boldsymbol{\pi}^g(j)}|g) = \frac{1}{\beta}, \forall 1 \leq j \leq \text{cutoff}$

$q^{(t)}((x, y)_{\boldsymbol{\pi}^g(j)}|g) = \frac{n_i}{N}, \forall j > \text{cutoff}$

end

end

end

these two groups, under which $S \perp Y$ (p_{ideal}). Such underlying conflicts prevent the group DRO from formulating a worst-case distribution that can eliminate covariate shift, resulting in a passive reliance on certain spurious correlations.

Imperfect partitions of training data are common in practice, as it can be expensive or infeasible to acquire the labels of spurious attributes for each training instance. For example, we may only have rough partitions based on the data sources or the outputs from (unsupervised) clustering algorithms. Our analysis shows that under these practical settings, the group DRO algorithm can not effectively alleviate covariate shift due to the rigid treatment of group losses.

4.5 Proposed Method: Group-conditional DRO

Since group DRO can be problematic with imperfect partitions, we propose a more flexible uncertainty set over the joint distribution of (x, y, g), i.e. $q(g)q(x, y|g)$, using fine-grained weights over instances within each group instead of treating the entire group as a whole. We extend the α-covered distribution

to both the group-level ($q(g)$) and conditional instance-level ($q(x, y|g)$) distributions to define the uncertainty set $\mathcal{Q}$. At training time, a sample is weighted by both its group weight induced from $q(g)$ as well as the instance-level weight induced from $q(x, y|g)$. Specifically, the new uncertainty set is

$$\mathcal{Q}^{\alpha,\beta} = \Bigg\{ q(g)q(x,y|g) : q(g) \leq \frac{p_{\text{train}}(g)}{\alpha}, \\ \frac{1}{N} \leq q(x,y|g) \leq \frac{p_{\text{train}}(x,y|g)}{\beta}, \forall x, y, g \Bigg\}, \tag{4.6}$$

where N is the number of training examples and $\alpha, \beta \in (0, 1]$. Denote n_i the number of samples in group i, then $p^{train}(x, y|g = i) = \frac{1}{n_i}$. The second constraint of Eq. 4.6 can be rewritten as $\frac{1}{N} \leq q(x, y|g) \leq \frac{1}{\beta n_i}$. Compared with the β-covered distribution, we add a lower bound $q(x, y|g) \geq \frac{1}{N}$ to compensate for imbalanced group sizes. With a plain β-covered distribution for $q(x, y|g)$, the DRO objective roughly upweights a β-fraction of instance losses of each group. However, we only want to emphasize a small subset of examples that perform badly in the majority groups. Thus, we add this lower bound to $q(x, y|g)$ in Eq. 4.6 to directly "punish" larger groups. To see this, the percentage of examples that are upweighted by $\frac{1}{\beta}$ in group i is roughly $\frac{N-n_i}{N-n_i\beta}\beta$, which is monotonically decreasing function w.r.t. n_i. Therefore, the larger the group size n_i is, the smaller fraction of instances in group i are upweighted.

Online Optimization Algorithm. Similarly to the online greedy algorithm for group DRO (Oren et al., 2019) (details in Appendix .8), we interleave the updates between model parameters θ and the worst-case distribution $q(g)q(x, y|g)$. The greedy algorithm involves sorting losses of all the variables when updating the worst-case distribution defined by the α-covered distribution. However, frequently updating $q(x, y|g)$ over large-scale training data (e.g millions of samples) can be costly and unstable. Therefore, we only update $q(g)$ at every iteration, while performing updates on $q(x, y|g)$ lazily once every epoch or when the robust accuracy on the validation set drops (inner update criterion). We present the pseudo code for the training process in Alg. 1.

Discussions. Another potential approach to circumventing the purely group-level loss is constructing an instance-level uncertainty set (Ben-Tal et al., 2013a; Husain, 2020; Michel et al., 2021), however, the resulting $\mathcal{Q}$ can be too pessimistic (Hu et al., 2018; Duchi et al., 2019) or difficult to optimize (Michel et al., 2021). Instead, we leverage the structural information of data partitions and expand the flexibility of uncertainty set by incorporating the conditional probabilities of instances. Furthermore, this allows us to execute the min-max optimization in an efficient manner.

	male	female
dark	65,487 / 1,387	22,880 / 48,749
blonde	0 / 1,387	22,880 / 0

Table 4.2: The imperfect partitions for the CelebA dataset (G_1/G_2).

	no neg	neg 1	neg 2
contradiction	57,605 / 0 / 0	0 / 1,406 / 0	0 / 0 / 9,897
entailment	67,335 / 0 / 0	0 / 0 / 215	0 / 1,318 / 0
neutral	66,401 / 0 / 0	0 / 0 / 251	0 / 1,747 / 0

Table 4.3: The imperfect partitions for the MNLI dataset ($G_1/G_2/G_3$).

4.6 Experiments

In this section, we evaluate the proposed group-conditional DRO on one image classification task and two language tasks — natural language inference and toxicity detection. To demonstrate the effectiveness of our method under various partitions of data, we first introduce the clean (group number $m = |\mathcal{A}| \times |\mathcal{Y}|$) and imperfect data partitions of each task. As we discussed at the end of §4.4.2, there are various cases where the partitions of training data are imperfect such that each group is *not* purely associated with examples from one pair of (a, y). In this section, we inspect several cases reflecting diverse properties of partitions to evaluate our method. First, on the image and NLI tasks, we manually design adversarial partitions of data such that there are explicit conflicts between groups and purely reweighing over groups cannot eliminate covariate shift (§4.6.1). Second, we use the attributes provided by a supervised classifier to create the imperfect partitions of the toxicity data set (§4.6.1). Third, we also perform unsupervised clustering on the toxicity data set to obtain imperfect partitions in §4.6.4.

	White-aligned	AAE	Hispanic	Others
abusive	11,281	7,392	6,707	1,770
spam	8,147	1,041	541	4,301
normal	41,756	2,562	2,638	6,895
hateful	2,696	1,420	509	340

Table 4.4: Statistics of each group in the clean partition of the hate speech dataset. Data of each dialect attribute (column) corresponds one group in the imperfect partition.

Datasets	Methods	Clean Partition		Imperfect Partition	
		Robust Acc	Average Acc	Robust Acc	Average Acc
Celeb-A	ERM	40.14 ± 0.99	**95.92 ± 0.05**	40.14 ± 0.99	**95.92 ± 0.05**
	resampling	86.81 ± 1.26	92.72 ± 0.28	44.17 ± 1.15	95.58 ± 0.03
	group DRO (EG)	86.11 ± 1.96	92.33 ± 0.65	42.92 ± 0.91	95.82 ± 0.07
	group DRO (greedy)	88.19 ± 2.31	92.65 ± 0.20	45.97 ± 1.73	95.81 ± 0.09
	GC-DRO	**88.75 ± 0.82**	92.92 ± 0.16	**82.85 ± 1.54**	89.32 ± 2.21
MNLI	ERM	70.84 ± 2.47	**86.18 ± 0.18**	70.84 ± 2.47	**86.18 ± 0.18**
	resampling	67.02 ± 2.43	85.72 ± 0.37	67.26 ± 1.63	85.22 ± 0.58
	group DRO (EG)	**77.88 ± 1.36**	85.16 ± 0.44	69.66 ± 1.98	84.96 ± 0.56
	group DRO (greedy)	75.14 ± 3.96	85.82 ± 0.24	70.34 ± 2.19	86.02 ± 0.25
	GC-DRO	77.82 ± 1.45	85.04 ± 0.67	**75.32 ± 0.93**	84.82 ± 0.74
FDCL18	ERM	34.30 ± 1.83	**79.70 ± 1.05**	34.30 ± 1.83	79.70 ± 1.05
	resampling	55.44 ± 4.69	72.04 ± 1.99	26.10 ± 4.11	**80.66 ± 0.52**
	group DRO (EG)	55.98 ± 1.67	70.06 ± 3.06	35.20 ± 2.24	79.58 ± 0.95
	group DRO (greedy)	56.83 ± 2.94	70.52 ± 1.99	36.24 ± 3.80	79.40 ± 1.12
	GC-DRO	**57.28 ± 2.71**	70.26 ± 0.94	**48.42 ± 6.72**	72.02 ± 2.96

Table 4.5: Robust and average test accuracy and standard deviation on the three tasks.

4.6.1 Data and Tasks

Object Recognition. We use the **CelebA** dataset (Liu et al., 2015) which has 162,770 training examples of celebrity faces. We classify the hair color from $\mathcal{Y} = \{\text{blond, dark}\}$ following the set up in Sagawa et al. (2020a). In this task, labels are spuriously correlated with the demographic information — gender of the input $\mathcal{A} = \{\text{female, male}\}$, which together with $\mathcal{Y}$ results in 4 clean groups. The statistics of groups in the imperfect partition are presented in Tab. 4.2 (separated by "/"), each of which consists of data from multiple values of (a, y). Concretely, we create an imperfect partition of **2** groups with two explicit spurious correlations: i) in group G_1 (dark, male) are spuriously correlated since we put all their counterparts (blonde, male) in group G_2; ii) similarly, (dark, female) in G_2 are spuriously correlated.

Natural Language Inference (NLI). NLI is the task of determining whether a hypothesis is true (entailment), false (contradiction) or undetermined (neutral) given a premise. We use the **MultiNLI** dataset (Williams et al., 2018a) and follow the train/dev/test split in Sagawa et al. (2020a), which results in 206,175 training examples. Gururangan et al. (2018) have shown that there is spurious correlation between the label of contradiction and the presence of negation words (*nobody, nothing, no, never*) due to annotation artifacts. We further split the negation words into two groups: set 1 (*nobody, nothing*) and

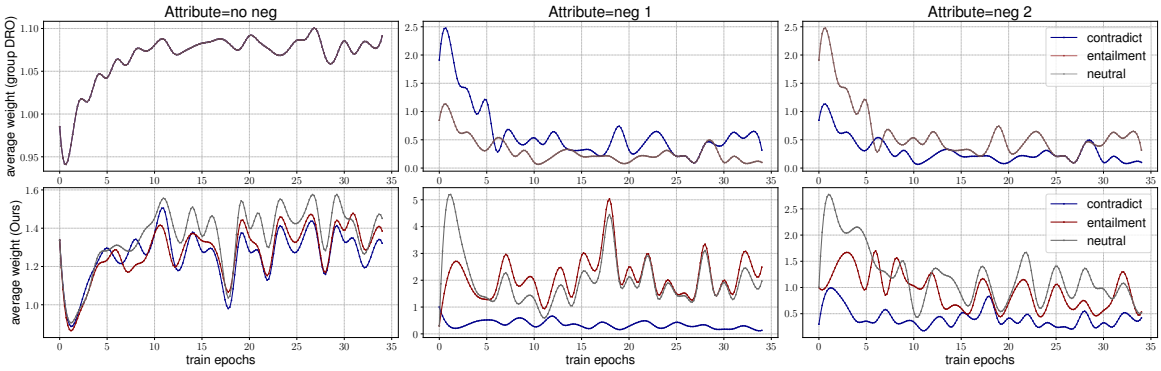


Figure 4.2: Under the imperfect partition of the MNLI dataset, the aggregated average training weights of instance losses in each group divided by attributes and labels (top: group DRO; bottom: GC-DRO).

set 2 (*no, never*) to have more variety in the attributes, i.e. $\mathcal{A} = \{\text{no negation, negation 1, negation 2}\}$, which together with labels forms 9 groups in the clean partition. We create **3** groups in the imperfect partition as shown in Tab. 4.3, where G_1 only contains examples from a =no negation, while G_2 and G_3 contain data from both a =negation 1 and a =negation 2. This causes a dilemma when upweighting either of the groups.

Toxicity Detection. This task aims to identity various forms of toxic languages (e.g. abusive speech, hate speech), an application with practical and important real-world consequences. Sap et al. (2019) have shown that there is a strong correlation between certain surface markers of English spoken by minority groups and the labels of toxicity. And such biases can be acquired and propagated by models trained on these corpora. We perform experiments on the **FDCL18** (Fortuna and Nunes, 2018) dataset, a corpus of 100k tweets annotated with four labels: $\mathcal{Y} = \{\text{hateful, spam, abusive and normal}\}$. Since the dataset does not contain the dialect information, we follow Sap et al. (2019) and use annotations predicted by the dialect classifier in Blodgett et al. (2016) to label each example as one of four English varieties: $\mathcal{A} = \{\text{White-aligned, African American (AAE), Hispanic, and Other}\}$. As noted in Sap et al. (2019), these automatically obtained labels correlate highly with self-reported race and provide an accurate proxy for the dialect labels. These dialect attributes and toxicity labels together divide the dataset into 16 groups in the clean partition. To construct the imperfect partitions, we investigate a natural setting where data is divided by the dialect attributes, therefore we have **4** groups in the imperfect partition. The test set of FDCL18 contains some groups that are severely under-represented. In order to make the robust accuracy reliable yet still representative of the under-represented groups, we follow Michel et al. (2021) and combine groups that contain less than 100 samples into a single group to compute robust test accuracies.

4.6.2 Experimental Setup

We fine-tune pretrained models for object recognition and NLP tasks that achieve high average test accuracies, specifically ResNet18 (He et al., 2016) on CelebA and RoBERTa (Liu et al., 2019) on the MultiNLI and FDCL18 datasets. We select hyperparameters by the robust validation accuracy. For the clean partitions, we set $\alpha = 0.2, \beta = 0.5$ for all the three tasks. For the imperfect partitions, we set a relatively lower value of β to highlight badly performed instances within groups. Specifically, for NLP tasks we set $\alpha = 0.5$, $\beta = 0.2$ and 0.25 for NLI and toxicity detection respectively, and for the image task, we set $\alpha = 0.2, \beta = 0.1$. For more training details, see Appendix .10. We measure both the *average* accuracy over all the test data as well as the *robust* accuracy (worst accuracy across all groups). Even though different partitions (clean/imperfect) are used at training time, we always evaluate the model's robust accuracy across groups of the clean partitions of the test data.

We compare with **ERM**, which minimizes the average training loss on the empirical training distribution, formally

$$\mathcal{L}_{\text{ERM}}(\theta) = \mathbb{E}_{(x,y)\sim P_{train}}[\ell(x, y; \theta)] \tag{4.7}$$

We also compare with two variants of group DRO with different objective and optimization procedures: a **greedy** (Oren et al., 2019) algorithm for CVaR-group DRO and a **exponentiated-gradient (EG)** (Sagawa et al., 2020a) procedure with full simplex . Note that while previous work (Sagawa et al., 2020a) has found the greedy algorithm is unstable and underperforms EG, we did not observe this issue with our implementation where we took a slight different approach to compute the worst expected loss and we detail this difference in Appendix .10.2. In addition, we compare with the **resampling** method, which optimizes on minibatches sampled from uniform group frequencies, which is often used for imbalanced datasets.

4.6.3 Main Results

We present the robust and average test accuracies of all three tasks under different partitions in Tab. 4.5. Models are selected based on the worst-performing accuracy of group (of the clean partition) in the validation set. All the results are averaged over 5 runs with different random seeds. Except for ERM, all the models leverage the group information at training time. **First**, as expected, ERM models attain high average test accuracies across all the datasets but perform poorly on the worst-case group. **Second**, we observe that under the clean partition, group DRO models always significantly outperform ERM on the worst-group test accuracy with modest drop in the average test accuracy. And we also note that group DRO optimized with the greedy algorithm performs on par with that optimized by the EG based algorithm. By contrast, the resampling method can not consistently perform well on the worst test groups on all datasets. Furthermore, our method performs similarly to or slightly better than group DRO on all three datasets under the clean partition. **Third**, under the imperfect partition, neither group

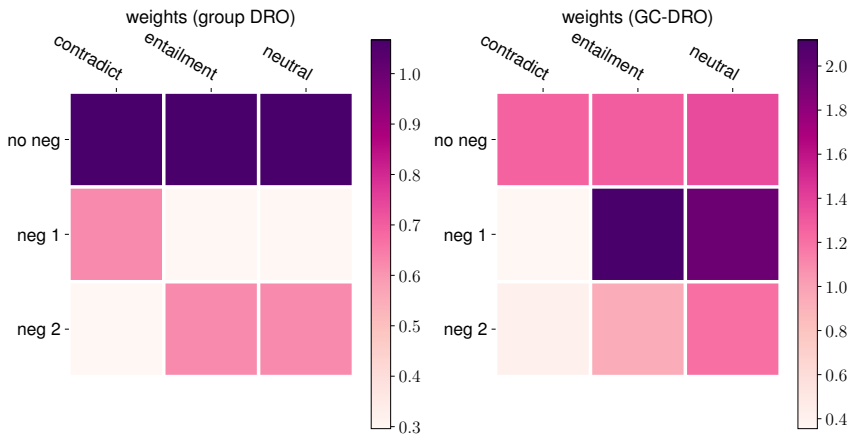


Figure 4.3: The heatmap of summarized learning weights for different groups.

DRO nor resampling can perform well in the worst test groups and achieves similar performance to that of ERM models. On the other hand, our method performs significantly better in terms of the robust accuracy on all three datasets, with 5-37 points in improvement over group DRO models. Although the results are worse compared to those under the clean partition, we demonstrate that our method is much more agnostic to the underlying data partitions.

4.6.4 Analysis

Why does GC-DRO perform well on robust accuracy? In this section, we investigate why group-conditional DRO works well under imperfect partitions. To do this, we first compute the actual weight ($\frac{n_i q^{(t)}(g_i) q^{(t)}(x,y|g_i)}{\hat{p}^{train(t)}(g_i)}$ in Alg. 1) applied to each instance (x_i, y_i) at every step t for group DRO and our method respectively. The groups in imperfect partitions contain instances from different values of (a, y) and to study the effects of learning weights on the test groups, we aggregate the weights of instances on each group (a, y) (i.e. the clean partition) by taking average over all steps in each epoch. In Fig. 4.2, we plot the dynamic aggregated weights over the training course learned with the imperfect partition (3 groups, see Tab. 4.3) of the MNLI dataset. We observe that GC-DRO can assign higher weights to instances that belong to groups of (a =negation, $y \neq$ contradiction), which helps prevent the model from learning the spurious correlations between a = negation and y =contradiction. On the contrary, group DRO can not accomplish this goal because it can not handle these subgroups inside groups in a fine-grained way.

To make this trend more clear, we summarize the weights across all the epochs for each group of (a, y) and present the heat map in Fig. 4.3. We can see that group DRO focuses on learning from the large group that does not contain negation words but pays less attention to those minority groups. On the contrary, our method encourages the model to learn from minority groups that can help combat spurious features.

	Robust Acc	Average Acc
ERM	34.30 $\pm$ 1.83	79.7 $\pm$ 1.05
resampling	34.20 $\pm$ 2.36	79.4 $\pm$ 1.24
group DRO (EG)	32.84 $\pm$ 2.72	**80.5** $\pm$ **0.59**
group DRO (greedy)	34.48 $\pm$ 4.69	79.62 $\pm$ 0.59
GC-DRO (ours)	**45.06** $\pm$ **6.77**	70.7 $\pm$ 4.81

Table 4.6: Average and robust test accuracies of FDCL18 under the partitions via unsupervised clustering.

On groups produced by unsupervised clustering. We study a more realistic setting where no group information is available and we use an unsupervised clustering algorithm to produce the partitions. Specifically, we first embed the training sentences of the FDCL18 dataset with Sentence-BERT (Reimers et al., 2019), a well-performing semantic sentence embedder, then we use K-means to obtain 8 clusters. In Tab. 4.6, we show the robust and average accuracy on the test set of the toxicity detection task. Our method once again significantly outperforms other baseline methods on the robust test accuracy, which demonstrates the robustness of GC-DRO under different partitions.

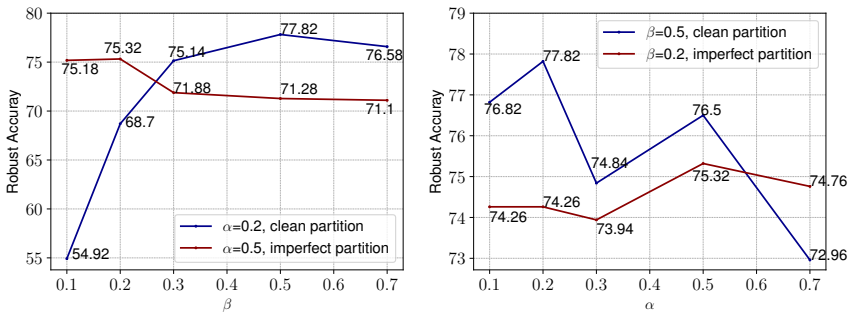


Figure 4.4: Ablation studies on α and β on the MNLI datasets.

Ablation studies on α and β. We perform ablation studies on the two important hyperparameters α and β used in our method. In Fig 4.4, we fix one value and vary the other and plot the robust test accuracies over 5 random runs (the variance of average test accuracies is very small) on the NLI task. We observe that GC-DRO is less sensitive to different combinations of α and β under the imperfect partitions. However, for the clean partitions, a larger β and a smaller α tends to yield better performance, as GC-DRO behaves more close to the plain group DRO.

4.7 Conclusion

Through a mathematical characterization of features used in prediction, we have demonstrated that under covariate shift ERM models can pick up spurious features or miss robust features. The GC-DRO

algorithm resulting from this analysis allows for a more flexible uncertainty set that performs consistently well in the worst test group under different partitions. This new understanding of features opens up new avenues in both redesigning our distributionally robust algorithms, and further characterizing possible spurious factors that may influence model robustness, for example through unsupervised learning.

Chapter 5

Distributionally Robust Multilingual Machine Translation

While the previous chapter revisits group DRO to improve the worst-case group performance when confronted with biased training data that contains spurious correlations, in this chapter we propose a group DRO algorithm for a more natural and realistic NLP problem — multilingual neural machine translation (MNMT), which learns to translate multiple language pairs with a single model. In MNMT, the heavy data imbalance between languages is common and hinders the model from performing uniformly across language pairs. In this chapter, we propose a new learning objective for MNMT based on *group DRO*, which automatically adjusts the training distribution by minimizing the worst-case expected loss over the set of language pairs.

5.1 Introduction

Multilingual methods that process multiple languages with one single model have gained favor across a variety of NLP tasks (Firat et al., 2016; Ha et al., 2016; Johnson et al., 2017; Devlin et al., 2019c; Aharoni et al., 2019; Conneau et al., 2020b) because (1) training and deployment of one multilingual model is more computationally efficient compared to maintaining one model for each language (Arivazhagan et al., 2019), (2) training multilingually can improve accuracy, particularly for low-resource languages (LRLs) (Zoph et al., 2016; Neubig and Hu, 2018; Pires et al., 2019).

However, in multilingual training, the amount and type of training data available varies greatly across languages. Because most models are trained using empirical risk minimization (ERM), which minimizes the average training loss on the training set, high-resource languages (HRLs) with large amounts of data contribute to the majority of the training objective. When model capacity is limited, this results in trade-offs or decreased performance on some languages, particularly LRLs (Arivazhagan et al., 2019; Wang et al., 2020f, 2021c). To better control this trade-off, a common practice is to balance the training

distribution by heuristic oversampling of LRLs (Johnson et al., 2017; Neubig and Hu, 2018; Arivazhagan et al., 2019).

Although simple data balancing can improve the performance on LRLs significantly, it is far from optimal. First, the sampling hyperparameters need to be adjusted for different datasets. Second, the use of simple heuristics does not consider the inherent level of difficulty in learning each language, the similarity or distance of languages in the multilingual dataset, and other factors that affect cross-lingual transfer. Because of this, previous work has indicated the importance of learning strategies that are explicitly tailored for each multilingual learning scenario (Wang et al., 2020d).

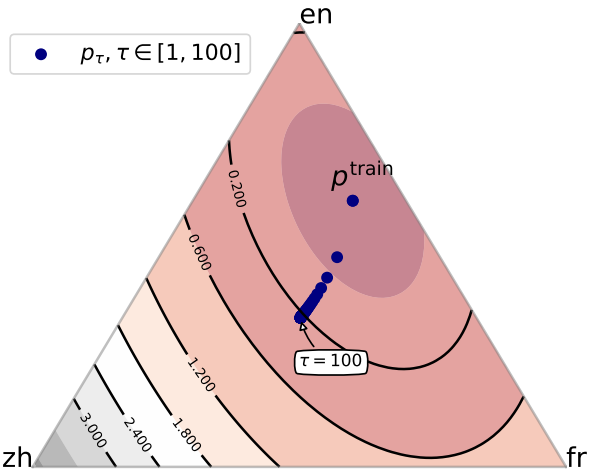


Figure 5.1: Illustration of different training distributions where the training distribution of the three languages `fr`, `zh` and `en` is $(0.1, 0.3, 0.6)$. Contours represent different radii of the χ^2-ball around p^{train}. The blue points are the tempered distributions described in §5.2.1.

In this chapter, we propose a new learning procedure for multilingual translation that automatically adjusts the training distribution of different languages using distributionally robust optimization (DRO) (Ben-Tal et al., 2013a; Duchi et al., 2016). In constrast to ERM, DRO casts learning as a game between the learner and an adversary, where the learner picks a model while the adversary picks the hardest data distribution for that model within an *uncertainty set* $\mathcal{Q}$ of potential distributions we wish to perform well on (which typically contains the training distribution P_0).

We first demonstrate how to apply DRO to multilingual training by letting the adversary choose the relative weights of individual language pairs in the training objective. However, we empirically find that naively applying existing results to multilingual learning yields inferior results to ERM, mostly because (1) standard DRO objectives tend to be overly conservative and only take into account language pairs with very large losses and (2) existing optimization algorithms for DRO essentially reweigh the gradients of examples in a mini-batch, which implicitly changes the scale of the learning rates. This hurts modern NLP models like Transformers (Vaswani et al., 2017) that are highly sensitive to learning rate schedules.

To remedy this, we propose both a novel training objective and a corresponding optimization algorithm amenable to the multilingual setting. Our objective is a variation on Group DRO of Sagawa et al. with a less conservative uncertainty set, parameterized by the χ^2-divergence. To efficiently solve the min-max game, we propose an *iterated best response* scheme that re-samples the training data at each

epoch according to the worst weighting for the current model parameters, and then runs ERM training on the re-sampled dataset. Our method—which we refer to as χ-IBR —incurs negligible additional computational cost compared to ERM.

While this method applies to essentially any multilingual task, we specifically demonstrate its benefit on three sets of language pairs from two multilingual machine translation datasets. We experimentally test these choices by comparing several objectives and optimization algorithms, and results show that our method consistently outperforms existing DRO procedures and various strong baselines.

5.2 Preliminaries

Notation. Throughout this chapter, n denotes the training set size and d the number of parameters of the model. For $m \in \mathbb{N}$, Δ^m denotes the m-dimensional simplex, i.e. $\Delta^m \coloneqq \{q \in \mathbb{R}^m, q_i \geq 0 \text{ and } \sum_i q_i = 1\}$. The data lies in $\mathcal{X} \times \mathcal{Y}$ where $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ consists of a source and target sentence pair with $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{L_{\boldsymbol{x}}})$ and $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{L_{\boldsymbol{y}}})$. $\ell : (\mathcal{X} \times \mathcal{Y}) \times \mathbb{R}^d \to \mathbb{R}$ refers to the loss function. We consider maximum-likelihood estimation, i.e. for a target sentence $\boldsymbol{y}$ with $L_{\boldsymbol{y}}$ tokens, we define

$$\ell(\boldsymbol{x}, \boldsymbol{y}; \theta) = -\frac{1}{L_{\boldsymbol{y}}} \sum_{i=1}^{L_{\boldsymbol{y}}} \log p(\boldsymbol{y}_i | \boldsymbol{x}, \boldsymbol{y}_{<i}; \theta)$$

5.2.1 Multilingual Machine Translation

In contrast to bilingual machine translation, which translates from a single source language S to a target language T, multilingual neural MT (MNMT) learns a single model to translate between N language pairs $\{(S_1, T_1), \ldots (S_N, T_N)\}$. The training data D_{train} is the concatenation of the N parallel datasets, i.e. $D_{\text{train}} = [D_1; D_2, \cdots; D_N]$. We can then define the probability over each language pair $p^{\mathsf{train}} \in \Delta^N$ as $p_i^{\mathsf{train}} = \frac{|D_i|}{\sum_j |D_j|}$. We now describe two common training objectives for MNMT.

Empirical Risk Minimization (ERM). The simplest and most common approach for MNMT is to minimize the empirical loss over data points, which we will refer to as ERM. More precisely, we define the average loss on a parallel dataset D_i as

$$\mathcal{L}(\theta; D_i) \coloneqq \frac{1}{|D_i|} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in D_i} \ell(\boldsymbol{x}, \boldsymbol{y}; \theta).$$

ERM for multilingual models then corresponds to simply minimizing the loss over D, i.e. over all the aggregated parallel sentence pairs. This yields

$$\widehat{\theta}_n^{\mathsf{ERM}} \in \arg\min_{\theta} \mathcal{L}(\theta; D) = \sum_{i \leq N} p_i^{\mathsf{train}} \mathcal{L}(\theta; D_i).$$

Classical results in learning theory guarantee that, under mild assumptions, as n goes to infinity, $\widehat{\theta}_n^{\mathsf{ERM}}$ will show good performance on test sets with the same distribution as D. However, this does not guarantee that our model will perform adequately on individual parallel datasets. To remedy this issue, several

works propose varying the sampling distribution—or equivalently the weighting of the parallel datasets in ERM—in order to encourage more uniform performance across language pairs.

Weighted Risk Minimization and Sampling Strategies. The amount of training data can vary significantly across language pairs. As a result, in ERM training—i.e. optimizing for the average loss across sentence pairs—HRLs contribute most of the objective, resulting in poor performance on LRLs. Balancing the objective—or equivalently, the usage of training data—between HRLs and LRLs is important to maintain good performance across all languages (Devlin et al., 2019c; Arivazhagan et al., 2019). A commonly adopted approach in multilingual training is *temperature*-based sampling (Arivazhagan et al., 2019; Conneau et al., 2020b) where the probability of sampling data from D_i is proportional to its data size exponentiated by a temperature term τ, i.e. $p_{\tau,i} = \frac{|D_i|^{1/\tau}}{\sum_j |D_j|^{1/\tau}}$ (referred to as ERM with τ in §5.4). This is equivalent to optimizing the re-weighted objective

$$\mathcal{L}_\tau(\theta; D) = \sum_{i \leq N} p_{\tau,i} \mathcal{L}(\theta; D_i).$$

As a result, $\tau = 1$ corresponds to ERM where most of the contribution comes from the HRLs and $\tau = \infty$ corresponds to sampling language pairs uniformly at random, i.e. with data from LRLs being over-sampled. This approach comes with three major drawbacks (1) τ is an extra hyper-parameter that requires tuning for each MNMT instance to balance the performance across both HRLs and LRLs, (2) this heuristic sampling method does not consider the training dynamics of each language and how the optimal sampling distribution might evolve during the training process and (3) the parameterization of p_τ is not only very constrained (essentially one degree of freedom), it is also only a function of the quantity of training data, which is too rigid to achieve the desired performance.

To resolve some of the above issues, the recently proposed MultiDDS (Wang et al., 2020d) uses a gradient-based meta-learning approach to learn the sampling distribution over language pairs to maximize gradient similarity with a multilingual development set. However, due to the necessity to calculate and store extra gradients, their approach comes at an increased computational and memory cost. In contrast, χ-IBR enjoys the same computational complexity as ERM, and as we show in experiments it also largely outperforms MultiDDS.

5.2.2 Distributionally Robust Optimization

In contrast to ERM and related sampling strategies which optimize for a fixed training distribution, DRO aims to find a model θ that performs well on an entire collection of potential test distributions $\mathcal{Q}$ (the *uncertainty set*). Formally, we wish to

$$\underset{\theta}{\text{minimize}} \sup_{Q \in \mathcal{Q}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}) \sim Q}[\ell(\boldsymbol{x}, \boldsymbol{y}; \theta)]. \tag{5.1}$$

Originating from operations research Delage and Ye (2010); Ben-Tal et al. (2013a,b); Bertsimas et al. (2018), DRO has proven a promising way to tackle robustness in a variety of machine learning and NLP problems Hashimoto et al. (2018b); Oren et al. (2019); Levy et al. (2020).

We present here a recent variant, Group DRO, developed by Sagawa et al. (2020a). As introduced in §4.4.1, group DRO incorporates additional information about the data distribution to define more meaningful uncertainty sets. Abstractly, this method assumes a collection of distributions over subpopulations $\{P_g\}_{g\in\mathcal{G}}$ such that the training distribution is a mixture of these subpopulations. Importantly, it assumes that this *group structure* is observed. The Group DRO objective then minimizes the worst-case loss over these groups, which is equivalent to equation 5.1 with $\mathcal{Q} = \{\sum_{g\in\mathcal{G}} q_g P_g : q \in \Delta^{|\mathcal{G}|}\}$, in other words, all possible mixtures of the distributions over subpopulations. In MNMT, the N language pairs at our disposal naturally correspond to groups; thus the Group DRO objective can be defined as

$$\mathcal{L}^{\text{GDRO}}(\theta; D) = \max_{i\in[N]} \mathcal{L}(\theta; D_i). \tag{5.2}$$

In other words, Group DRO wishes to find a model θ that performs well for the *worst* language pair. Oren et al. (2019) propose a related but less conservative objective, CVaR-Group DRO at level $\alpha \in [0, 1]$ which, considers instead the average of the $\lceil \alpha N \rceil$ largest group losses.

5.3 Methods for Distributionally Robust Multilingual Learning

As we previewed in §5.2, Group DRO is a natural objective for the multilingual setting. However, in experiments we found that naively applying existing DRO objectives fails to achieve performance on par with strong baselines, often improving results on language pairs with high losses but sacrificing too much performance overall. Our main contribution is showing how to successfully apply DRO to the MNMT setting, and to the best of our knowledge, our work is the first to do so. To that end, our methodological contributions are two-fold: (i) we first describe the shortcomings of the Group DRO objective equation 5.2, then propose a related training criterion that addresses these issues, (ii) we describe an optimization algorithm to solve the min-max optimization problem that is amenable to the MNMT setting.

5.3.1 χ^2-Group DRO

Shortcomings of Group DRO. A weakness of the objective equation 5.2 is that apart from the language pair with largest loss, the objective does not take into account the value of the loss on the other language pairs. To illustrate this, consider this example with $N = 3$ language pairs and suppose that there exists two parameters θ_1 and θ_2 with the following loss:

$$\mathcal{L}(\theta_1; D_1) = 0.1,\ \mathcal{L}(\theta_1; D_2) = 0.1,\ \mathcal{L}(\theta_1; D_3) = 1.1$$
$$\mathcal{L}(\theta_2; D_1) = 1.0,\ \mathcal{L}(\theta_2; D_2) = 1.0,\ \mathcal{L}(\theta_2; D_3) = 1.0$$

We have that $\mathcal{L}^{\text{GDRO}}(\theta_1; D) = 1.1$ but $\mathcal{L}^{\text{GDRO}}(\theta_2; D) = 1.0$. Consequently, the Group DRO objective will prefer θ_2 to θ_1 while clearly one would pick θ_1 over θ_2 in most practical cases. The aforementioned CVaR-Group DRO also exhibits this behavior and ignores the values of the language pairs not in the largest α-fraction.

To address this issue, let us rewrite the objective

$$\mathcal{L}^{\text{GDRO}}(\theta; D) = \max_{q \in \Delta^N} \sum_{i \leq N} q_i \mathcal{L}(\theta; D_i),$$

where the equality holds because the optimal weighting just puts all the mass on the language with largest loss. A natural way to make the objective less conservative is to instead take the maximum over a *subset* of the simplex $\mathcal{U} \subset \Delta^N$. This leads to the following objective

$$\mathcal{L}^{\mathcal{U}}(\theta; D) = \sup_{q \in \mathcal{U}} \sum_{i \leq N} q_i \mathcal{L}(\theta; D_i). \tag{5.3}$$

Different choices of $\mathcal{U}$ will yield different objectives with different robustness properties. Note that this is a general formulation as $\mathcal{U} = \{p^{\text{train}}\}$ reduces to the ERM objective, while $\mathcal{U}_\alpha = \{q : \|q/p^{\text{train}}\|_\infty \leq 1/\alpha\}$ corresponds to the CVaR-Group DRO of Oren et al. (2019). We would like to choose $\mathcal{U}$ such that optimizing this objective results in models with better performance on language pairs with large losses (typically LRLs) without significant degradation of average performance.

To this end, we turn to a common and flexible choice for $\mathcal{U}$: f-divergence balls Csiszár (1967) of radius $\rho > 0$ around p^{train}, namely

$$\mathcal{U}_\rho^f := \{q : D_f(q, p^{\text{train}}) \leq \rho\}, \tag{5.4}$$

where $D_f(q, p) := \sum_{i \leq N} p_i f(q_i/p_i)$. In particular, we propose using the χ^2-divergence which corresponds to $f(t) = \frac{1}{2}(t-1)^2$ and define $\chi^2(q, p) = \frac{1}{2}\sum_i p_i (q_i/p_i - 1)^2$ with its corresponding uncertainty set $\mathcal{U}_\rho^{\chi^2}$. The χ^2-divergence has a long history Ben-Tal et al. (2013b) and previous work shows that minimizing the robust loss with the χ^2-uncertainty set enjoys favorable statistical properties such as optimally trading-off bias and variance Duchi and Namkoong (2019) or guaranteeing robustness and fairness Hashimoto et al. (2018b); Duchi and Namkoong (2020). We refer to the objective $\mathcal{L}^{\mathcal{U}_\rho^{\chi^2}}$ as the χ^2-Group DRO. Going back to the toy example, setting $\rho = 0.1$, yields that $\mathcal{L}^{\mathcal{U}_\rho^{\chi^2}}(\theta_1; D) = 0.64$ while $\mathcal{L}^{\mathcal{U}_\rho^{\chi^2}}(\theta_2; D) = 1.0$. With the χ^2-uncertainty set, the objective rightly prefers θ_1 to θ_2 and takes into account all the losses and not only the largest. We further confirm these intuitions and show in §5.4 and §5.5 that this is a suitable choice of uncertainty set for MNMT.

5.3.2 Optimization algorithm

Desiderata of the optimization algorithm. Minimizing the objective equation 5.3 effectively corresponds to a min-max optimization problem. Even in the relatively simple convex-concave setting, these

are generally harder to solve than convex minimization problems. Recall that we want to

$$\underset{\theta}{\text{minimize}} \sup_{q:\chi^2(q,p^{\text{train}})\leq\rho} \sum_{i\leq N} q_i \mathcal{L}(\theta; D_i). \tag{5.5}$$

Due to the architectures we consider in MNMT, we wish for an algorithm that effectively changes the sampling distribution over mini-batches of data instead of importance-weighting the gradients. Indeed, standard MT architectures such as the Transformer (Vaswani et al., 2017) are extremely sensitive to learning rate schedules and we empirically observe that importance-weighting the gradients result in poor performance.

The canonical way to solve min-max problem is via primal-dual methods (PD) Nemirovski et al. (2009) (see background in Appendix .12), where at each step t, one keeps two vectors (θ_t, q_t) and alternates between a gradient descent step on θ_t and a gradient ascent step on q_t. One can perform these updates efficiently as they only require *unbiased stochastic gradient estimate* of the loss w.r.t. θ_t and q_t. To obtain unbiased gradient estimate of the loss w.r.t. θ_t, one either has to, at each step, sample a mini-batch of examples from $\mathsf{Multinomial}(q_t)$ and return the gradient of the loss or sample a mini-batch from $\mathsf{Multinomial}(p^{\text{train}})$ and importance-weight the gradient.

As previously mentioned, the latter is not suitable for Transformer-type architectures. The former option is not ideal as it is more convenient for an algorithm to decide the sequence of mini-batches every epoch rather than every optimizer step as this integrates much more smoothly with data loaders in deep learning frameworks, especially when doing distributed training. As a result, we posit that primal-dual algorithms are not an adequate choice for optimizing DRO-type objectives in our setting. We further discuss this point in §5.5.

To circumvent this issue, we consider a different optimization algorithm which we refer to as *iterated best response* (IBR), where, instead of doing a single gradient descent and ascent step, we iterate between (approximately) solving the maximization (resp. minimization) on q (resp. θ), while keeping θ_t (resp. q_t) fixed. This is similar in spirit to algorithms in the game theory literature where individuals play the optimal strategy (best response) assuming everyone else's strategies remain constant. Under some mild assumptions, this procedure converges to the equilibrium of the game Roughgarden (2016). Formally, we alternate between

$$\theta^{t+1} \leftarrow \arg\min_{\theta} \sum_i q_i^t \mathcal{L}(\theta; D_i) \tag{5.6}$$

$$q^{t+1} \leftarrow \arg\max_{q:\chi^2(q,p^{\text{train}})\leq\rho} \sum_i q_i \mathcal{L}(\theta^{t+1}; D_i). \tag{5.7}$$

Practical implementation. As we show in Appendix .11, given the values of the loss, the q-update (equation 5.7) is computed to accuracy ϵ in $O(N \log(1/\epsilon))$ steps. Indeed, by taking dual Boyd and Vandenberghe (2004) of equation 5.7, we transform the N-dimensional problem into a one-dimensional

Algorithm 2: Iterated Best Response

Input: N parallel datasets $D_1, \dots, D_N$, radius of the uncertainty set ρ, number of epochs T, learning rate schedule $\{\eta_{t,j}\}_{t \leq T, j \leq n}$, baseline loss $\{b_i\}_{i \leq N}$, EMA parameter $\lambda \in [0, 1]$.

Set $p_i^{\mathsf{train}} \leftarrow |D_i| / (\sum_j |D_j|)$.

Set $q^0 \leftarrow p^{\mathsf{train}}$ and $\widehat{\mathcal{L}}_i \leftarrow 0$ for $i \in [N]$.

Initialize θ^0.

for $t = 0, \dots, T-1$ **do**

▷ *Construction of* $D(q^t)$

Set $n(q^t)_i \leftarrow \lceil N \cdot q_i^t \rceil$.

Sample $n(q^t)_i$ data points from D_i and add them to $D(q^t)$ for $i \in [N]$.

$D(q^t) \leftarrow \mathsf{Shuffle}(D(q^t))$.

▷ θ- *and* $\widehat{\mathcal{L}}$*-update*

for $(\boldsymbol{x}_j, \boldsymbol{y}_j) \in D(q^t)$ **do**

Let k be the language pair of $(\boldsymbol{x}_j, \boldsymbol{y}_j)$.

$\theta^{t,j} \leftarrow \theta^{t,j-1} - \eta_{t,j} \nabla \ell(\boldsymbol{x}_j, \boldsymbol{y}_j; \theta^{t,j-1})$.

$\widehat{\mathcal{L}}_k \leftarrow \lambda \cdot \ell(\boldsymbol{x}_j, \boldsymbol{y}_j; \theta^{t,j-1}) + (1 - \lambda) \cdot \widehat{\mathcal{L}}_k$.

end

▷ q*-update*

$q^{t+1} \leftarrow \arg\max_{q : \chi^2(q, p^{\mathsf{train}})} \sum_{i \leq N} q_i \left(\widehat{\mathcal{L}}_i - b_i \right)$

$\theta^{t+1,0} \leftarrow \theta^{t, |D(q^t)|}$.

end

Return $\theta^{T,0}$.

root-finding procedure over the dual variable which we efficiently solve with a bisection. We provide the details in Appendix .11. Note that computation cost is negligible compared to computing the gradient of the loss. We implement the θ-update of equation 5.6 by running a training epoch on a *re-sampling* of the training set D according to q^{t+1}.

To compute the loss values $\{\mathcal{L}(\theta_t; D_i)\}_{i \leq N}$, necessary to perform the q-update, one needs to compute the loss $\ell(\boldsymbol{x}, \boldsymbol{y}; \theta_t)$ for every single example $(\boldsymbol{x}, \boldsymbol{y}) \in D$. This is prohibitively expensive to compute at every epoch. To avoid this, we keep track of the (approximate) historical values of the token-level loss $\hat{\mathcal{L}}_k$ on each language pair k with an exponential moving average (EMA) (see line 14 in Algorithm 2). We precisely describe our implementation in Algorithm 2. We see that it respects our desiderata and comes at no computational cost. In §5.5, we compare primal-dual and iterated best response for various uncertainty sets.

	Method	aze	bel	glg	slk	tur	rus	por	ces	Avg
		0.004	0.006	0.013	0.081	0.240	0.274	0.243	0.136	
any→en	ERM (τ=1)	13.76	19.26	31.56	32.37	26.83	25.65	45.12	29.81	28.05
	MultiDDS	13.95	19.52	31.82	32.29	25.82	25.09	43.90	29.58	27.75
	χ-IBR	**14.68**	**19.98**	**31.89**	**33.16**	**27.76**	**26.08**	**45.33**	**30.76**	**28.71**
en→any	ERM (τ=1)	6.85	11.83	24.26	24.56	16.13	20.44	**40.33**	22.24	20.83
	MultiDDS	7.16	12.98	24.37	23.72	15.13	19.54	38.92	21.42	20.41
	χ-IBR	**8.50**	**14.61**	**25.94**	**25.98**	**16.87**	**21.57**	40.20	**23.63**	**22.16**
	Method	**bos**	**mar**	**hin**	**mkd**	**ell**	**bul**	**fra**	**kor**	**Avg**
		0.007	0.017	0.0330	0.045	0.237	0.308	0.340	0.363	
any→en	ERM (τ=1)	24.58	12.10	23.96	33.59	38.85	39.89	40.83	19.65	29.18
	MultiDDS	**25.19**	11.65	23.66	34.04	38.90	39.14	40.13	19.36	29.01
	χ-IBR	25.12	**12.52**	**24.42**	**34.47**	**39.42**	**40.24**	**40.98**	**20.72**	**29.74**
en→any	ERM (τ=1)	14.41	5.34	16.34	25.44	32.61	35.17	39.04	8.67	22.13
	MultiDDS	16.58	5.36	15.99	25.43	31.66	33.03	36.33	8.29	21.58
	χ-IBR	**17.33**	**5.59**	**16.90**	**28.02**	**33.82**	**36.37**	**40.35**	**9.13**	**23.44**

Table 5.1: BLEU scores of the best ERM model (among τ=1/5/100, $\tau = 5/100$ are significantly worse than $\tau = 1$, thus we omit these results), MultiDDS and our approach on the test sets of the TED dataset. Bold (resp. underlined) values indicate the best (resp. second best) performance for each language pair. Values under the language codes are the proportion of the language in the training data.

Subtracting the baseline. To account for different *hardness* amongst groups, Oren et al. propose subtracting a per-group scalar—which we refer to as a baseline—to each group loss before taking the maximum over q. They learn this baseline using a generative bi-gram model on each group. Recall that $\widehat{\theta}^{\mathsf{ERM}}$ is the parameter we obtain when we minimize the average loss and define $\widehat{\theta}^{\tau}$ when optimizing $\mathcal{L}_\tau$. In this work, we propose using $b_i = \mathcal{L}(\widehat{\theta}^{\mathsf{ERM}}; D_i)$ or $b_i = \mathcal{L}(\widehat{\theta}^{\tau}; D_i)$. Intuitively, the baseline corresponds to the minimum performance we wish for our model on the given group and as such the loss obtained with ERM and its temperature variants are natural candidates. We show in §5.5 that these yield significant improvement and conveniently make our method more robust to the choice of ρ. We leave different (potentially learned) choices of baseline to future work.

5.4 Experiments

5.4.1 Datasets

We evaluate the proposed method on two datasets: the 58-languages TED talk corpus (Qi et al., 2018) and WMT datasets. For the TED corpus, we evaluate on two sets of languages with varying levels of

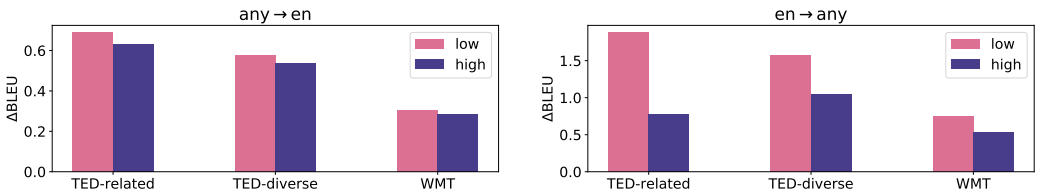


Figure 5.2: ΔBLEU of low- and high-resource language groups for the three language sets. ΔBLEU = difference of BLEU scores of χ-IBR and the best ERM model.

Method	any→en					en→any				
	deu	**fra**	**tam**	**tur**	**Avg**	**deu**	**fra**	**tam**	**tur**	**Avg**
ERM (τ=1)	**29.98**	30.32	15.81	19.85	23.99	**23.82**	33.09	9.28	13.29	19.87
ERM (τ=5)	29.25	31.60	16.31	21.89	24.76	22.67	32.36	10.04	16.09	20.29
ERM (τ=100)	28.75	30.71	15.80	**22.44**	24.43	22.02	31.65	10.41	**16.44**	20.13
MultiDDS	29.31	31.41	16.12	21.43	24.57	22.99	31.55	10.09	14.51	19.79
χ-IBR	29.67	**31.75**	**16.48**	22.33	**25.06**	23.45	**33.16**	**10.73**	15.53	**20.72**

Table 5.2: BLEU scores of the ERM (τ=1/5/100), MultiDDS and our method on the test sets of the WMT dataset. The ratios of training data of de, fr, ta and tr are (0.499, 0.359, 0.102, 0.039).

language diversity following Wang et al. (2020d): (1) *related* includes 4 LRLs (aze, bel, glg, slk) and their corresponding related HRL (tur, rus, pos, ces). (2) *diverse* includes 8 languages with varying amount of data without considering linguistic similarities (bos, mar, hin, mkd, ell, bul, fra, kor)[1]. Both of the related and diverse sets have around 760K sentences of training data.

For WMT, we consider 2 HRLs (German:deu and French:fra) and 2 LRLs (Tamil:tam and Turkish:tur). We subsample around 5M training sentences from the parallel corpus provided by the WMT shared task. Specifically, the training data of deu-eng, fra-eng is from WMT14, tam-eng is from WMT20 and tur-eng is from WMT18. We use the corresponding test and dev sets from each shared task for evaluation and validation.

We evaluate both *en-to-any* (translate English to a target language) and *any-to-en* (translate a source language into English) directions for all language sets. We provide dataset statistics in Appendix .13.

5.4.2 Experimental setup

For the translation models, we adopt the encoder-decoder Transformer (Vaswani et al., 2017) architecture with the implementation provided in fairseq (Ott et al., 2019). For the relatively smaller TED dataset, we follow (Wang et al., 2020d) and use a small-size Transformer that has 6 encoder and decoder layers and 4 attention heads.[2] For the WMT dataset, we use a Transformer-base architectures that also has 6 layers

[1] See Wang et al. (2020d) for the interpretation of the language codes.

[2] We train for more steps with larger batch size, which yields better better results than reported in Wang et al. (2020d).

Method	any→en					en→any				
	deu	**fra**	**tam**	**tur**	**Avg**	**deu**	**fra**	**tam**	**tur**	**Avg**
FastDRO	25.14	27.58	12.71	15.54	20.24	21.39	28.21	8.88	12.74	17.81
GDRO with PD	26.72	29.13	15.78	21.89	23.38	20.81	29.43	10.29	15.52	19.01
CVaR-GDRO with PD	28.62	30.70	15.94	20.61	23.97	22.81	32.44	9.68	14.33	19.82
CVaR-GDRO with IBR	29.14	31.65	16.31	20.98	24.52	22.34	31.97	10.15	14.82	19.82
χ^2-GDRO with PD	29.49	31.47	16.07	21.24	24.57	23.10	32.30	9.87	14.70	19.99
ERM (τ=5)	29.25	31.60	16.31	21.89	24.76	22.67	32.36	10.04	**16.09**	20.29
χ-IBR	**29.67**	**31.75**	**16.48**	**22.33**	**25.06**	**23.45**	**33.16**	**10.73**	15.53	**20.72**

Table 5.3: BLEU scores of different DRO objectives and algorithms—primal-dual (PD) and iterated best response (IBR)—on the WMT test sets.

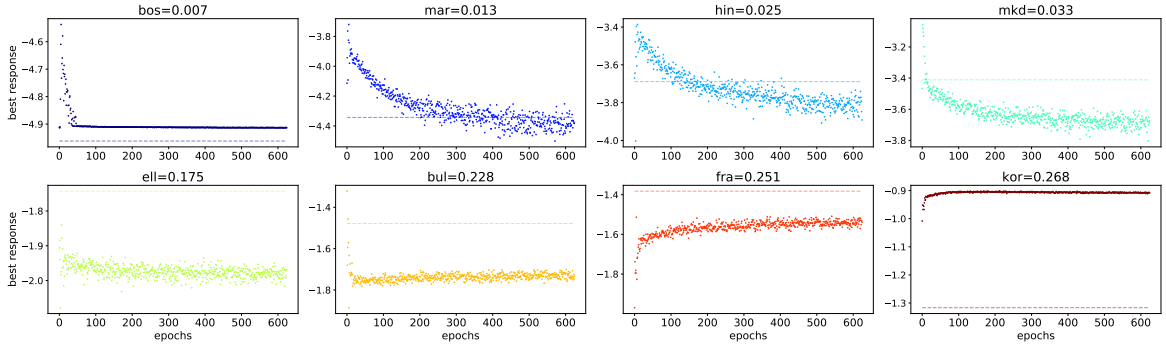


Figure 5.3: Best response q (in log-scale) across epochs on the TED diverse dataset for the any→en direction.

but with larger hidden dimension size and 8 attention heads. The model is trained for 200K and 300K steps for TED and WMT respectively with the batch size of 65,536 tokens. For both datasets, we learn the sentencepiece (Kudo and Richardson, 2018) vocabulary for the English and the combined corpus of other languages respectively. We use beam search with beam size 5 for decoding and report the SacreBLEU score (Post, 2018; Papineni et al., 2002a) on test sets for evaluation. For the TED and WMT datasets respectively, the constraint size ρ for the chi-square ball is set to be 0.05 and 0.3, and for the baseline losses we use the average token-level loss on each D_i computed from the ERM model with $\tau = 1$ and $\tau = 100$—see §5.5 for more analyses of these choices. We provide additional pre-processing and training details in Appendix .14.

Baselines. We compare with (1) **temperature-based sampling** method described in §5.2.1 in three standard settings ($\tau = 1/5/100$), where $\tau = 100$ approximates uniform sampling over language pairs, and (2) **MultiDDS** described in §5.2.1. In addition, we also perform extensive empirical studies over different DRO uncertainty sets and optimization procedures in §5.5.

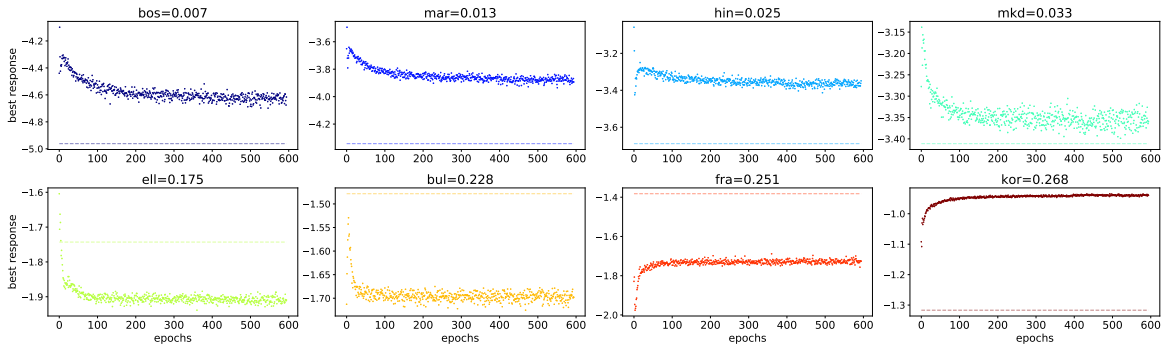


Figure 5.4: Best response q (in log-scale) across epochs on the TED diverse dataset for the en→any direction, the dashed line is the true data probability (in log-scale).

5.4.3 Main Results

We present the BLEU scores of en→any and any→en translation directions on TED and WMT data in Tab. 5.1 and 5.2 respectively. **First**, for both TED and WMT datasets, χ-IBR outperforms all the other baseline methods in terms of average BLEU score over all language pairs. Particularly, χ-IBR performs significantly better on the TED dataset, with up to 1.33 average BLEU score improvement over the best baseline method. By taking a closer look at the BLEU scores for each individual language pairs, χ-IBR improves over almost all the language pairs for both translation directions. Secondly, as expected from temperature-based sampling methods, different values of τ achieve different trade-offs between the performances on HRLs and LRLs. As we explained in §5.2.1, large values of τ favor LRLs as this results in data being sampled with equal probability from each language pair while small values of τ approach ERM and will benefit HRLs. As a result, τ needs to be carefully tuned to achieve adequate performance on both HRLs and LRLs.

Importantly, χ-IBR achieves a significantly better trade-off than the sampling method for any value of τ. We show in Fig. 5.2 the quantitative improvements of χ-IBR over the best τ for various datasets and in both translation directions. Surprisingly, while the improvements are larger on LRLs, we consistently observe improvements on HRLs. This indicates that finding the right sampling distribution over language pairs facilitates cross-lingual transfer.[3] We further observe that χ-IBR achieves more significant improvements in the en→any direction than in the any→en direction. This further supports our hypothesis. Indeed, it is attested in previous work on MNMT (Arivazhagan et al., 2019) that it is harder to decode to multiple languages than encode from multiple languages and as such, the en→any direction is a significantly harder multi-task learning problem. As such, this is where optimal cross-lingual transfer would yield the larger gains, which is what we observe in practice. We also note that our method incurs negligible computational overhead compared to ERM.

[3]The model we used for TED has a high capacity relative to its data size, this is probably why we observe more improvements on the TED data as it has more space for improvement.

5.5 Analysis

The importance of the sampling distribution. An advantage of our method over temperature-based sampling methods is that it dynamically adjusts the training distribution as the model evolves and does not compute it solely as a function of amount of training data. Our hypothesis is that this is important to achieve good performance across language pairs and that different sampling distributions will be adequate at different stages of training. We empirically check our hypothesis by studying how the training distribution q (the so-called best response) changes across training epochs. In Fig. 5.3 and 5.4, we plot the best response of χ-IBR across epochs on the TED-diverse dataset for both translation directions. In addition, we also plot the historical losses in Fig. 4. Our first observation is that the optimal q noticeably evolves across epochs which further showcases the need for dynamically adjusting the sampling distribution. We make the following observations (i) χ-IBR demonstrates the desired behavior and, at the early stages of training, always down-weights HRLs and up-weights LRLs; (ii) somewhat counter-intuitively, there is no direct correlation between $|D_i|$, the amount of data in language i and the final value $\mathcal{L}(\theta^{(t)}; D_i)$. The latter further evidences the limitations of sampling distributions only based on the amounts of training data $|D_i|$. Indeed, while `kor` is a HRL, it is typologically much farther from English so there is more inherent uncertainty in the task. Consequently, it has larger losses and is consistently up-weighted throughout training. On the other hand, while `hin` is a LRL, it achieves low loss after being up-weighted during the early stages of training and is consequently down-weighted after that.

Comparisons with other DRO objectives and optimization procedures. We demonstrate the benefits of χ-IBR over other DRO objectives by extensively evaluating a range of robust objectives and associated optimization algorithms. In terms of objective, we compare against (1) **Group DRO** Sagawa et al. (2020a), (2) **CVaR-Group DRO** Oren et al. (2019) and (3) **FastDRO** Levy et al. (2020). In terms of algorithms, we experiment with primal-dual methods and our proposed iterated best response procedure which we both described in §5.3.2. Note that in the case of Group DRO (i.e. $\mathcal{U} = \Delta^N$), iterated best response is not a sensible choice as it would result in each training epoch being spent on a single language pair. In the case of CVaR Group DRO, we follow the implementation of Oren et al. (2019), which is a hybrid of the two optimization algorithms with a primal update on θ and a best response update on q. We compare the performance of these methods on the WMT dataset. For fairness, we baseline losses in the same way for all the DRO objectives. We first observe that, outside of χ-IBR, none of the DRO objectives are competitive with temperature-weighted ERM. We also observe that for both uncertainty sets, iterated best response convincingly outperforms the same objective trained with primal-dual. We finally note that, for a fixed optimization algorithm, χ^2-Group DRO outperforms the CVaR objective on all but one language pairs. This validates both our choice of uncertainty set and of optimization procedure.

Dataset	Setting	Avg BLEU
TED	(a) ERM, $\tau = 1$	20.83
	(b) ERM, $\tau = 100$	19.78
	(c) Ours, $\rho = 0.05$, w/o BL	22.21
	(d) Ours, $\rho = 0.1$, w/o BL	21.09
	(e) Ours, $\rho = 0.05$, BL: $\tau = 1$	22.16
	(f) Ours, $\rho = 0.1$, BL: $\tau = 1$	22.13
	(g) Ours, $\rho = 0.05$, BL: $\tau = 100$	20.87
WMT	(h) Ours, $\rho = 0.1$, BL: $\tau = 1$	20.34
	(i) Ours, $\rho = 0.1$, BL: $\tau = 100$	20.62

Table 5.4: Average BLEU on the test sets of en→any direction, BL is short for baseline loss.

The effects of baselined losses. We study the effect of the choice of baseline on the performance across languages. In Tab. 5.4, we empirically evaluate different baseline choices and uncertainty sizes ρ. We observe that in the TED dataset, baseline-ing with $\mathcal{L}(\widehat{\theta}^{\mathsf{ERM}}; D_i)$ performs significantly better than baseline-ing with $\mathcal{L}(\widehat{\theta}^{\tau=100}; D_i)$ while it is reversed for WMT. We explain this by observing that the LRLs in TED consist of very small amounts of data (on the order of a few thousands) and using $\tau = 100$ results in a severe oversampling of LRLs, which the model then fits perfectly. As a result, recall the intuition that the baseline sets a lower bound on the performance we wish to achieve but because of the small training data and overfitting, the model disproportionately up-weighs the LRLs, which harms overall performance. This does not occur in WMT and uniform sampling across language pairs sets a good target performance for DRO methods. Finally, we see that with the right baseline loss, our method is more robust to different choices of ρ (e.g., comparing (c) and (d) versus (e) and (f)). We consistently observe this for other translation directions and datasets.

5.6 Conclusion

We showed how to successfully apply DRO to the MNMT setting and automatically adjust the sampling distribution over language pairs resulting in sizeable improvements in performance. We posit that this approach would also be successful in other multilingual scenarios, e.g. multilingual pre-training or fine-tuning a pre-trained multilingual model. Under these settings, we usually have models with larger capacity, e.g. Mixture-of-Experts models or using adapters for each task and each language in a multi-task multilingual setting, thus we hope the proposed framework can better balance the capacity based on the task difficulty or the resources. It is necessary to increase the strength of regularization to prevent the model from overfitting to small datasets, which can lead to more meaningful absolute loss values. Our work also raises a few questions: (i) what are the right baseline losses? (ii) surprisingly, χ-IBR also improves performance on HRLs; under what circumstances does cross-lingual transfer happen and which languages does it benefit most?

Part III

Efficient Transfer Learning of Pre-trained Language Models

Chapter 6

Towards a Unified View of Parameter-Efficient Transfer Learning

Fine-tuning large pretrained language models on downstream tasks has become the de-facto learning paradigm in NLP. Under this paradigm, distribution shift naturally arises which is the shift from the data distribution in the pre-training stage to the data distribution of the downstream task. Conventional approaches fine-tune all the parameters of the pretrained model for a downstream task, which is not only prohibitive as the model size and the number of tasks grow but also leads to catastrophic forgetting of previously acquired knowledge during pre-training. Recent work has proposed a variety of parameter-efficient transfer learning methods that only fine-tune a small number of (extra) parameters to attain strong performance. In this part, we explore paremeter-efficient fine-tuning methods for transfer learning of pre-trained language models, which naturally alleviate catastrophic forgetting. First, in §6, we break down the design of state-of-the-art parameter-efficient transfer learning methods and present a unified framework that establishes connections between them. Moreover, we instantiate new state-of-the-art parameter-efficient transfer learning method based on our unified framework. Next, in §7, we use parameter-efficient transfer learning to continually adapt a pre-trained model to improve zero-shot task generalization performance.

6.1 Introduction

Transfer learning from pre-trained language models (PLMs) is now the prevalent paradigm in natural language processing, yielding strong performance on many tasks (Peters et al., 2018; Devlin et al., 2019a; Qiu et al., 2020). The most common way to adapt general-purpose PLMs to downstream tasks is to fine-tune all the model parameters (*full fine-tuning*). However, this results in a separate copy of fine-tuned model parameters for each task, which is prohibitively expensive when serving models that perform a large number of tasks. This issue is particularly salient with the ever-increasing size of PLMs, which now

range from hundreds of millions (Radford et al., 2019; Lewis et al., 2020a) to hundreds of billions (Brown et al., 2020) or even trillions of parameters (Fedus et al., 2021).

To mitigate this issue, a few lightweight alternatives have been proposed to update only a small number of extra parameters while keeping most pretrained parameters frozen. For example, *adapter tuning* (Houlsby et al., 2019) inserts small neural modules called adapters to each layer of the pretrained network and only the adapters are trained at fine-tuning time. Inspired by the success of prompting methods that control PLMs through textual prompts (Brown et al., 2020; Liu et al., 2021b), *prefix tuning* (Li and Liang, 2021b) and *prompt tuning* (Lester et al., 2021) prepend an additional l tunable prefix tokens to the input or hidden layers and only train these soft prompts when fine-tuning on downstream tasks. More recently, Hu et al. (2021) learn low-rank matrices to approximate parameter updates. We illustrate these methods in Figure 6.1. These approaches have all been reported to demonstrate comparable performance to full fine-tuning on different sets of tasks, often through updating less than 1% of the original model parameters. Besides parameter savings, parameter-efficient tuning makes it possible to quickly adapt to new tasks without catastrophic forgetting (Pfeiffer et al., 2021) and often exhibits superior robustness in out-of-distribution evaluation (Li and Liang, 2021b).

However, we contend that the important ingredients that contribute to the success of these parameter-efficient tuning methods are poorly understood, and the connections between them are still unclear. In this chapter, we aim to answer three questions: (1) How are these methods connected? (2) Do these methods share design elements that are essential for their effectiveness, and what are they? (3) Can the effective ingredients of each method be transferred to others to yield more effective variants?

In order to answer these questions, we first derive an alternative form of prefix tuning that reveals prefix tuning's close connections with adapters (§6.3.1). Based on this we then devise a unified framework that frames the aforementioned methods as different ways to modify the hidden representations of frozen PLMs (§6.3.2). Our unified framework decomposes previous methods along a *shared* set of design dimensions, such as the function used to perform the modification, the position in which to impose this modification, and how to integrate the modification. This framework allows us to transfer design choices across approaches to propose new variants such as adapters with multiple heads (§6.3.3). In experiments, we first show that existing parameter-efficient tuning methods still lag behind full fine-tuning on higher-resource and challenging tasks (§6.4.2), as exemplified in Figure 6.2. Then we utilize the unified framework to identify critical design choices and validate the proposed variants empirically (§6.4.3-6.4.6). Our experiments on four NLP benchmarks covering text summarization, machine translation (MT), text classification, and general language understanding, demonstrate that the proposed variant uses less parameters than existing methods while being more effective, matching full fine-tuning results on all four tasks. Our code is available at https://github.com/jxhe/unify-parameter-efficient-tuning.

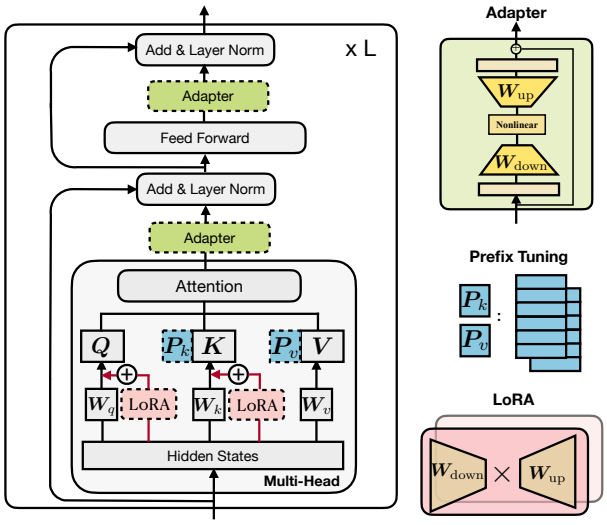


Figure 6.1: Illustration of the transformer architecture and several state-of-the-art parameter-efficient tuning methods. We use blocks with dashed borderlines to represent the added modules by those methods.

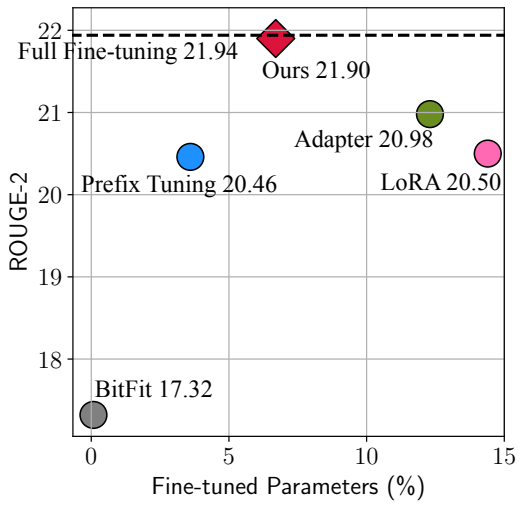


Figure 6.2: Performance of different methods on the XSum (Narayan et al., 2018) summarization task. The number of fine-tuned parameters is relative to the tuned parameters in full fine-tuning.

6.2 Preliminaries

6.2.1 Recap of the transformer Architecture

The transformer model (Vaswani et al., 2017) is now the workhorse architecture behind most state-of-the-art PLMs. In this section we recap the equations of this model for completeness. Transformer models are composed of L stacked blocks, where each block (Figure 6.1) contains two types of sub-layers: multi-head self-attention and a fully connected feed-forward network (FFN).[1] The conventional attention function maps queries $\boldsymbol{Q} \in \mathbb{R}^{n \times d_k}$ and key-value pairs $\boldsymbol{K} \in \mathbb{R}^{m \times d_k}, \boldsymbol{V} \in \mathbb{R}^{m \times d_v}$:

$$\text{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}, \tag{6.1}$$

where n and m are the number of queries and key-value pairs respectively. Multi-head attention performs the attention function in parallel over N_h heads, where each head is separately parameterized by $\boldsymbol{W}_q^{(i)}, \boldsymbol{W}_k^{(i)}, \boldsymbol{W}_v^{(i)} \in \mathbb{R}^{d \times d_h}$ to project inputs to queries, keys, and values. Given a sequence of m vectors $\boldsymbol{C} \in \mathbb{R}^{m \times d}$ over which we would like to perform attention and a query vector $\boldsymbol{x} \in \mathbb{R}^d$, multi-head

[1] In an encoder-decoder architecture, the transformer decoder usually has another multi-head cross-attention module between the self-attention and FFN, which we omit here for simplicity.

attention (MHA) computes the output on each head and concatenates them:[2]

$$\text{MHA}(\boldsymbol{C}, \boldsymbol{x}) = \text{Concat}(\text{head}_1, \cdots, \text{head}_\text{h})\boldsymbol{W}_o, \ \text{head}_\text{i} = \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q^{(i)}, \boldsymbol{C}\boldsymbol{W}_k^{(i)}, \boldsymbol{C}\boldsymbol{W}_v^{(i)}), \tag{6.2}$$

where $\boldsymbol{W}_o \in \mathbb{R}^{d\times d}$. d is the model dimension, and in MHA d_h is typically set to d/N_h to save parameters, which indicates that each attention head is operating on a lower-dimensional space. The other important sublayer is the fully connected feed-forward network (FFN) which consists of two linear transformations with a ReLU activation function in between:

$$\text{FFN}(\boldsymbol{x}) = \text{ReLU}(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2, \tag{6.3}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d\times d_m}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d_m\times d}$. Transformers typically use a large d_m, e.g. $d_m = 4d$. Finally, a residual connection is used followed by layer normalization (Ba et al., 2016).

6.2.2 Overview of Previous Parameter-efficient Tuning Methods

Below and in Figure 6.1, we introduce several state-of-the-art parameter-efficient tuning methods. Unless otherwise specified, they only tune the added parameters while the PLM's are frozen.

Adapters (Houlsby et al., 2019): The adapter approach inserts small modules (adapters) between transformer layers. The adapter layer generally uses a down-projection with $\boldsymbol{W}_\text{down} \in \mathbb{R}^{d\times r}$ to project the input $\boldsymbol{h}$ to a lower-dimensional space specified by bottleneck dimension r, followed by a nonlinear activation function $f(\cdot)$, and a up-projection with $\boldsymbol{W}_\text{up} \in \mathbb{R}^{r\times d}$. These adapters are surrounded by a residual connection, leading to a final form:

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + f(\boldsymbol{h}\boldsymbol{W}_\text{down})\boldsymbol{W}_\text{up}. \tag{6.4}$$

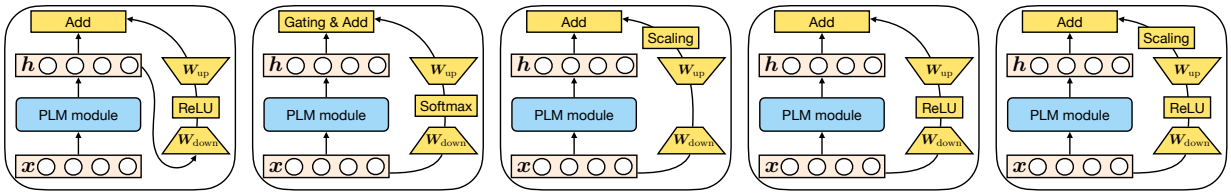
Houlsby et al. (2019) places two adapters sequentially within one layer of the transformer, one after the multi-head attention and one after the FFN sub-layer. Pfeiffer et al. (2021) have proposed a more efficient adapter variant that is inserted only after the FFN "add & layer norm" sub-layer.

Prefix Tuning (Li and Liang, 2021b): Inspired by the success of textual prompting methods (Liu et al., 2021b), prefix tuning prepends l tunable prefix vectors to the keys and values of the multi-head attention at every layer. Specifically, two sets of prefix vectors $\boldsymbol{P}_k, \boldsymbol{P}_v \in \mathbb{R}^{l\times d}$ are concatenated with the original key $\boldsymbol{K}$ and value $\boldsymbol{V}$. Then multi-head attention is performed on the new prefixed keys and values. The computation of head_i in Eq. 6.2 becomes:

$$\text{head}_i = \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q^{(i)}, \text{concat}(\boldsymbol{P}_k^{(i)}, \boldsymbol{C}\boldsymbol{W}_k^{(i)}), \text{concat}(\boldsymbol{P}_v^{(i)}, \boldsymbol{C}\boldsymbol{W}_v^{(i)})), \tag{6.5}$$

$\boldsymbol{P}_k$ and $\boldsymbol{P}_v$ are split into N_h head vectors respectively and $\boldsymbol{P}_k^{(i)}, \boldsymbol{P}_v^{(i)} \in \mathbb{R}^{l\times d/N_h}$ denote the i-th head vector. Prompt-tuning (Lester et al., 2021) simplifies prefix-tuning by only prepending to the input word embeddings in the first layer; similar work also includes P-tuning (Liu et al., 2021c).

[2] Below, we sometimes ignore the head index i to simplify notation when there is no confusion.



(a) Adapter (b) Prefix Tuning (c) LoRA (d) Parallel Adapter (e) Scaled PA

Figure 6.3: Graphical illustration of existing methods and the proposed variants. "PLM module" represents a certain sublayer of the PLM (e.g. attention or FFN) that is frozen. "Scaled PA" denotes scaled parallel adapter. We do not include multi-head parallel adapter here to save space.

LoRA (Hu et al., 2021): LoRA injects trainable low-rank matrices into transformer layers to approximate the weight updates. For a pre-trained weight matrix $\boldsymbol{W} \in \mathbb{R}^{d\times k}$, LoRA represents its update with a low-rank decomposition $\boldsymbol{W} + \Delta W = \boldsymbol{W} + \boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}$, where $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{d\times r}, \boldsymbol{W}_{\text{up}} \in \mathbb{R}^{r\times k}$ are tunable parameters. LoRA applies this update to the query and value projection matrices $(\boldsymbol{W}_q, \boldsymbol{W}_v)$ in the multi-head attention sub-layer, as shown in Figure 6.1. For a specific input $\boldsymbol{x}$ to the linear projection in multi-head attention, LoRA modifies the projection output $\boldsymbol{h}$ as:

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \boldsymbol{x}\boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}, \tag{6.6}$$

where $s \geq 1$ is a tunable scalar hyperparameter.[3]

Others: Other parameter-efficient tuning methods include BitFit (Ben Zaken et al., 2021), which only fine-tunes bias vectors in the pre-trained model, and diff-pruning (Guo et al., 2021), which learns a sparse parameter update vector.

6.3 Bridging the Gap – A Unified View

We first derive an equivalent form of prefix tuning to establish its connection with adapters. We then propose a unified framework for parameter-efficient tuning that includes several state-of-the-art methods as instantiations.

6.3.1 A Closer Look at Prefix Tuning

Eq. 6.5 describes the mechanism of prefix tuning which changes the attention module through prepending l learnable vectors to the original attention keys and values. Here, we derive an equivalent form of

[3]The public code of LoRA at https://github.com/microsoft/LoRA uses different s in different datasets, and we have verified the value of s could have a significant effect on the results.

Eq. 6.5 and provide an alternative view of prefix tuning:[4]

$$\begin{aligned}
&\text{head} = \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q, \text{concat}(\boldsymbol{P}_k, \boldsymbol{C}\boldsymbol{W}_k), \text{concat}(\boldsymbol{P}_v, \boldsymbol{C}\boldsymbol{W}_v)) \\
&= \text{softmax}\big(\boldsymbol{x}\boldsymbol{W}_q \text{concat}(\boldsymbol{P}_k, \boldsymbol{C}\boldsymbol{W}_k)^\top\big) \begin{bmatrix} \boldsymbol{P}_v \\ \boldsymbol{C}\boldsymbol{W}_v \end{bmatrix} \\
&= (1-\lambda(\boldsymbol{x}))\text{softmax}(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{W}_k^\top\boldsymbol{C}^\top)\boldsymbol{C}\boldsymbol{W}_v + \lambda(\boldsymbol{x})\text{softmax}(x\boldsymbol{W}_q\boldsymbol{P}_k^\top)\boldsymbol{P}_v \\
&= (1-\lambda(\boldsymbol{x}))\underbrace{\text{Attn}(\boldsymbol{x}\boldsymbol{W}_q, \boldsymbol{C}\boldsymbol{W}_k, \boldsymbol{C}\boldsymbol{W}_v)}_{\text{standard attention}} + \lambda(\boldsymbol{x})\underbrace{\text{Attn}(\boldsymbol{x}\boldsymbol{W}_q, \boldsymbol{P}_k, \boldsymbol{P}_v)}_{\text{independent of } \boldsymbol{C}},
\end{aligned} \tag{6.7}$$

where $\lambda(\boldsymbol{x})$ is a scalar that represents the sum of normalized attention weights on the prefixes:

$$\lambda(\boldsymbol{x}) = \frac{\sum_i \exp(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{P}_k^\top)_i}{\sum_i \exp(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{P}_k^\top)_i + \sum_j \exp(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{W}_k^\top\boldsymbol{C}^\top)_j}. \tag{6.8}$$

Note that the first term in Eq. 6.7, $\text{Attn}(\boldsymbol{x}\boldsymbol{W}_q, \boldsymbol{C}\boldsymbol{W}_k, \boldsymbol{C}\boldsymbol{W}_v)$, is the original attention without prefixes, whereas the second term is a position-wise modification independent of $\boldsymbol{C}$. Eq. 6.7 gives an alternative view of prefix tuning that essentially applies a position-wise modification to the original head attention output $\boldsymbol{h}$ through linear interpolation:

$$\boldsymbol{h} \leftarrow (1-\lambda(\boldsymbol{x}))\boldsymbol{h} + \lambda(\boldsymbol{x})\Delta\boldsymbol{h}, \quad \Delta\boldsymbol{h} := \text{softmax}(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{P}_k^\top)\boldsymbol{P}_v. \tag{6.9}$$

The Connection with Adapters: We define $\boldsymbol{W}_1$=$\boldsymbol{W}_q\boldsymbol{P}_k^\top$, $\boldsymbol{W}_2$=$\boldsymbol{P}_v$, f=softmax, and rewrite Eq. 6.9:

$$\boldsymbol{h} \leftarrow (1-\lambda(\boldsymbol{x}))\boldsymbol{h} + \lambda(\boldsymbol{x})f(\boldsymbol{x}\boldsymbol{W}_1)\boldsymbol{W}_2, \tag{6.10}$$

which reaches a very similar form to the adapter function in Eq. 6.4, except that prefix tuning is performing weighted addition while the adapter one is unweighted.[5] Figure 6.3b demonstrates the computation graph of prefix tuning from this view, which allows for abstraction of prefix tuning as a plug-in module like adapters. Further, we note that $\boldsymbol{W}_1 \in \mathbb{R}^{d_h \times l}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{l \times d_h}$ are low-rank matrices when l is small, and thus they function similarly to the $\boldsymbol{W}_{\text{down}}$ and $\boldsymbol{W}_{\text{up}}$ matrices in adapters. This view also suggests that the number of prefix vectors, l, plays a similar role to the bottleneck dimension r in adapters: they both represent the rank limitation of computing the modification vector $\Delta\boldsymbol{h}$. Thus we also refer l as the bottleneck dimension. Intuitively, the rank limitation implies that $\Delta\boldsymbol{h}$ is a linear combination of *the same* l (or $\leq l$) basis vectors for any $\boldsymbol{x}$.

The Difference from Adapters: In addition to the gating variable λ, we emphasize three differences between prefix tuning and adapters. (1) As demonstrated in Figure 6.3, prefix tuning uses $\boldsymbol{x}$, the input

[4] Without loss of generalization, we ignore the softmax scaling factor $\sqrt{d}$ for ease of notation.

[5] $\boldsymbol{h}$ in adapters and prefix tuning are usually different, as described more below. However, here we mainly discuss the functional form as adapters can, in principle, be inserted at any position.

Table 6.1: Parameter-efficient tuning methods decomposed along the defined design dimensions. Here, for clarity, we directly write the adapter nonlinear function as ReLU which is commonly used. The bottom part of the table exemplifies new variants by transferring design choices of existing approaches.

Method	$\Delta \boldsymbol{h}$ functional form	insertion form	modified representation	composition function
		Existing Methods		
Prefix Tuning	$\text{softmax}(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{P}_k^\top)\boldsymbol{P}_v$	parallel	head attn	$\boldsymbol{h} \leftarrow (1-\lambda)\boldsymbol{h} + \lambda\Delta\boldsymbol{h}$
Adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	sequential	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
LoRA	$\boldsymbol{x}\boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}$	parallel	attn key/val	$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \Delta\boldsymbol{h}$
		Proposed Variants		
Parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
Muti-head parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	head attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
Scaled parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \Delta\boldsymbol{h}$

of the PLM layer, to compute $\Delta\boldsymbol{h}$, while adapters use $\boldsymbol{h}$, the output of the PLM layer. Thus, prefix tuning can be thought of as a "parallel" computation to the PLM layer, whereas the typical adapter is "sequential" computation. (2) Adapters are more flexible with respect to where they are inserted than prefix tuning: adapters typically modify attention or FFN outputs, while prefix tuning only modifies the attention output of each head. Empirically, this makes a large difference as we will show in §6.4.4. (3) Eq. 6.10 applies to each attention head, while adapters are always single-headed, which makes prefix tuning more expressive: head attention is of dimension d/N_h – basically we have full rank updates to each attention head if $l \geq d/N_h$, but we only get full-rank updates to the whole attention output with adapters if $r \geq d$. Notably, prefix tuning is not adding more parameters than adapters when $l = r$.[6] We empirically validate such multi-head influence in §6.4.4.

6.3.2 The Unified Framework

Inspired by the connections between prefix tuning and adapters, we propose a general framework that aims to unify several state-of-the-art parameter-efficient tuning methods. Specifically, we cast them as learning a modification vector $\Delta\boldsymbol{h}$, which is applied to various hidden representations. Formally, we denote the hidden representation to be directly modified as $\boldsymbol{h}$, and the direct input to the PLM sub-module that computes $\boldsymbol{h}$ as $\boldsymbol{x}$ (e.g. $\boldsymbol{h}$ and $\boldsymbol{x}$ can be the attention output and input respectively). To characterize this modification process, we define a set of design dimensions, and different methods can be instantiated by varying values along these dimensions. We detail the design dimensions below, and illustrate how adapters, prefix tuning, and LoRA fall along them in Table 6.1:

Functional Form is the specific function that computes $\Delta\boldsymbol{h}$. We have detailed the functional form for adapters, prefix tuning, and LoRA in Eq. 6.4, 6.6, and 6.10 respectively. The functional forms of all

[6] We will detail in §6.4.1 the number of parameters added of different methods.

these methods are similar with a `proj_down` $\rightarrow$ `nonlinear` $\rightarrow$ `proj_up` architecture, while "nonlinear" degenerates to the identity function in LoRA.

Modified Representation indicates which hidden representation is directly modified.[7]

Insertion Form is how the added module is inserted into the network. As mentioned in the previous section and shown in Figure 6.3, traditionally adapters are inserted at a position in a sequential manner, where both the input and output are $\boldsymbol{h}$. Prefix tuning and LoRA – although not originally described in this way – turn out to be equivalent to a parallel insertion where $\boldsymbol{x}$ is the input.

Composition Function is how the modified vector $\Delta\boldsymbol{h}$ is composed with the original hidden representation $\boldsymbol{h}$ to form the new hidden representation. For example, adapters perform simple additive composition, prefix tuning uses a gated additive composition as shown in Eq. 6.10, and LoRA scales $\Delta\boldsymbol{h}$ by a constant factor and adds it to the original hidden representation as in Eq. 6.6.

We note that many other methods not present in Table 6.1 fit into this framework as well. For example, prompt tuning modifies the head attention in the first layer in a way similar to prefix tuning, and various adapter variants (Pfeiffer et al., 2021; Mahabadi et al., 2021) can be represented in a similar way as adapters. Critically, the unified framework allows us to study parameter-efficient tuning methods along these design dimensions, identify the critical design choices, and potentially transfer design elements across approaches, as in the following section.

6.3.3 Transferring Design Elements

Here, and in Figure 6.3, we describe just a few novel methods that can be derived through our unified view above by transferring design elements across methods: (1) *Parallel Adapter* is the variant by transferring the parallel insertion of prefix tuning into adapters. Interestingly, while we motivate the parallel adapter due to its similarity to prefix tuning, concurrent work (Zhu et al., 2021) independently proposed this variant and studied it empirically; (2) *Multi-head Parallel Adapter* is a further step to make adapters more similar to prefix tuning: we apply parallel adapters to modify head attention outputs as prefix tuning. This way the variant improves the capacity for free by utilizing the multi-head projections as we discuss in §6.3.1. (3) *Scaled Parallel Adapter* is the variant by transferring the composition and insertion form of LoRA into adapters, as shown in Figure 6.3e.

Our discussion and formulation so far raise a few questions: Do methods varying the design elements above exhibit distinct properties? Which design dimensions are particularly important? Do the novel methods described above yield better performance? We answer these questions next.

[7] Strictly speaking, all the hidden representations would be indirectly influenced by modifying the ones before them. Here we refer to the position being *directly* modified by the added module.

6.4 Experiments

6.4.1 General Setup

Datasets: We study four downstream tasks: (1) XSum (Narayan et al., 2018) is an English summarization dataset where models predict a summary given a news article; (2) English to Romanian translation using the WMT 2016 en-ro dataset (Bojar et al., 2016); (3) MNLI (Williams et al., 2018b) is an English natural language inference dataset where models predict whether one sentence entails, contradicts, or is neutral to another. (4) SST2 (Socher et al., 2013) is an English sentiment classification benchmark where models predict whether a sentence's sentiment is positive or negative.

Setup: We use $\text{BART}_{\text{LARGE}}$ (Lewis et al., 2020a) and a multilingual version of it, $\text{mBART}_{\text{LARGE}}$ (Liu et al., 2020d), as the underlying pretrained models for XSum and en-ro translation respectively, and we use $\text{RoBERTa}_{\text{BASE}}$ (Liu et al., 2019) for MNLI and SST2. We vary the bottleneck dimension within $\{1, 30, 200, 512, 1024\}$ if needed.[8] We mainly study adapters, prefix tuning (prefix), and LoRA which greatly outperform bitfit and prompt tuning in our experiments. In the analysis sections (§6.4.3-6.4.5) we insert adapters *either* at the attention or FFN layers for easier analysis, but include the results of inserting at both places in the final comparison (§6.4.6). We re-implement these methods based on their respective public code.[9] We use the huggingface transformers library (Wolf et al., 2020) for our implementation. Complete setup details can be found in Appendix .15.

Evaluation: We report ROUGE 1/2/L scores (R-1/2/L, Lin (2004)) on the XSum test set, BLEU scores (Papineni et al., 2002b) on the en-ro test set, and accuracy on the MNLI and SST2 dev set. For MNLI and SST2, we take the median of five random runs. We also report the number of tuned parameters relative to that in full fine-tuning (#params).

Number of Tunable Parameters: BART and mBART have an encoder-decoder structure that has three types of attention: encoder self-attention, decoder self-attention, and decoder cross-attention. RoBERTa only has encoder self-attention. For each attention sub-layer, the number of parameters used of each method is: (1) prefix tuning prepends l vectors to the keys and values and uses $2 \times l \times d$ parameters; (2) adapter has $\boldsymbol{W}_{\text{down}}$ and $\boldsymbol{W}_{\text{up}}$ thus uses $2 \times r \times d$ parameters; (3) LoRA employs a pair of $\boldsymbol{W}_{\text{down}}$ and $\boldsymbol{W}_{\text{up}}$ for query and value projections, hence uses $4 \times r \times d$ parameters. For the adapter modification at ffn, it uses $2 \times r \times d$ parameters which is the same as adapter at attention. Therefore, for a specific value of r or l, prefix tuning uses the same number of parameters as adapters, while LoRA uses more

[8] In some settings we use other values to match the number of added parameters of different methods.

[9] We verify that our re-implementation can reproduce adapter and prefix tuning on XSum, and LoRA on MNLI, by comparing with the results of running the original released code.

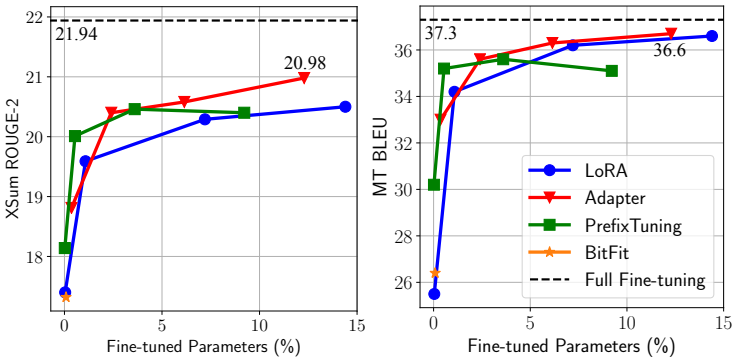


Figure 6.4: Performance of previous state-of-the-art parameter-efficient tuning methods on XSum (left) and en-ro (right).

Figure 6.5: Accuracy on the dev set of MNLI and SST2. MAM Adapter is proposed in §6.4.6. Bitfit numbers are from Ben Zaken et al. (2021).

Method (# params)	MNLI	SST2
Full-FT (100%)	$87.6_{\pm.4}$	$94.6_{\pm.4}$
Bitfit (0.1 %)	84.7	93.7
Prefix (0.5%)	$86.3_{\pm.4}$	$94.0_{\pm.1}$
LoRA (0.5%)	$87.2_{\pm.4}$	$94.2_{\pm.2}$
Adapter (0.5%)	$87.2_{\pm.2}$	$94.2_{\pm.1}$
MAM Adapter (0.5%)	$\mathbf{87.4}_{\pm.3}$	$94.2_{\pm.3}$

parameters. More details can be found in Appendix .16.

6.4.2 The Results of Existing Methods

We first overview the results of existing methods on the four tasks. As shown in Figure 6.4 and Table 6.5, while existing methods can achieve competitive performance on MNLI and SST2 by tuning fewer than 1% parameters, a large gap is still present if we add 5% parameters in XSum and en-ro. The gap remains significant even though we increase the relative parameter size to >10%. Even larger gaps have been observed in Raffel et al. (2020) on high-resource MT tasks. This shows that many methods that claimed comparable results to full fine-tuning on the GLUE benchmark with an encoder-only model (Guo et al., 2021; Ben Zaken et al., 2021; Mahabadi et al., 2021), or on relatively simple generation benchmarks such as E2E (Novikova et al., 2017) with an encoder-decoder model (Li and Liang, 2021b), may not generalize well to other standard benchmarks. The influencing factors could be complicated including the number of training samples, task complexity, or model architecture. We thus advocate for future research on this line to report results on more diverse benchmarks to exhibit a more complete picture of their performance profile. Below, our analysis will mainly focus on the XSum and en-ro datasets to better distinguish different design choices. We note that these two benchmarks are relatively high-resource performed with an encoder-decoder model (BART), while we will discuss the results on MNLI and SST2 with an encoder-only model (RoBERTa) in §6.4.6.

6.4.3 Which Insertion Form – Sequential or Parallel?

We first study the insertion form design dimension, comparing the proposed parallel adapter (PA) variant to the conventional sequential adapter (SA) over both the attention (att) and FFN modification. We also include prefix tuning as a reference point. As shown in Table 6.6, prefix tuning, which uses parallel

Figure 6.6: Comparison of different insertion forms for adapters, i.e. sequential adapter (SA) and parallel adapter (PA). We include the results of prefix tuning as a reference point.

Method	# params	XSum (R-1/2/L)	MT (BLEU)
Prefix, l=200	3.6%	43.40/20.46/35.51	35.6
SA (attn), r=200	3.6%	42.01/19.30/34.40	35.3
SA (ffn), r=200	2.4%	43.21/19.98/35.08	35.6
PA (attn), r=200	3.6%	43.58/20.31/35.34	35.6
PA (ffn), r=200	2.4%	**43.93/20.66/35.63**	**36.4**

Figure 6.7: Results on en-ro dataset.

Method	# params	MT (BLEU)
PA (attn), r=200	3.6%	35.6
Prefix, l=200	3.6%	35.6
MH PA (attn), r=200	3.6%	35.8
Prefix, l=30	0.1%	35.2
-gating, l=30	0.1%	34.9
PA (ffn), r=30	0.1%	33.0
PA (attn), r=30	0.1%	33.7
MH PA (attn), r=30	0.1%	**35.3**

insertion, outperforms attention sequential adapters. Further, the parallel adapter is able to beat sequential adapters in all cases,[10] with PA (ffn) outperforming SA (ffn) by 1.7 R-2 points on XSum and 0.8 BLEU points on en-ro respectively. Given the superior results of parallel adapters over sequential adapters, we focus on parallel adapter results in following sections.

6.4.4 Which Modified Representation – Attention or FFN?

Setup: We now study the effect of modifying different representations. We mainly compare attention and FFN modification. For easier analysis we categorize methods that modifies any hidden representations in the attention sub-layer (e.g. the head output, query, etc) as modifying the attention module. We compare parallel adapters at attention and FFN and prefix tuning. We also transfer the FFN modification to LoRA to have a LoRA (ffn) variant for a complete comparison. Specifically, we use LoRA to approximate the parameter updates for the FFN weights $\boldsymbol{W}_1 \in \mathbb{R}^{d\times d_m}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{d_m\times d}$. In this case $\boldsymbol{W}_{\text{up}}$ in LoRA for $\boldsymbol{W}_1$ (similar for $\boldsymbol{W}_{\text{down}}$ of $\boldsymbol{W}_2$) would have dimensions of $r \times d_m$, where $d_m = 4d$ as described in §6.2.1. Thus we typically use smaller r for LoRA (ffn) than other methods to match their overall parameter size in later experiments.

Results: As shown in Figure 6.8, *any* method with FFN modification outperforms *all* the methods with attention modification in all cases (the red markers are generally above all the blue ones, the only exception is ffn-PA with 2.4% params), often with fewer parameters. Second, the same method applied at FFN always improves over its attention counterpart. For example, LoRA (ffn) improves LoRA (attn) by 1 R-2 points on XSum. We also highlight that prefix tuning does not keep improving when we further increase the capacity, which is also observed in Li and Liang (2021b). These results suggest that *FFN modification can utilize the added parameters more effectively than attention, no matter what the functional*

[10] More results with different r can be found in Appendix .17, which exhibits similar observations.

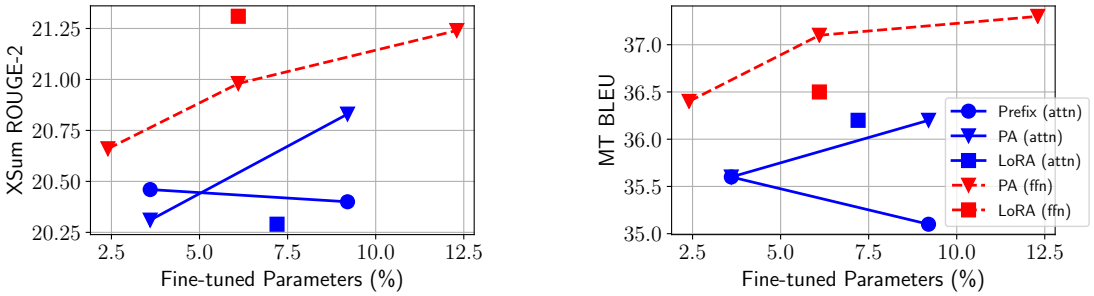


Figure 6.8: Results on XSum (left) and en-ro (right). PA represents parallel adapter. Blue and red markers apply modifications at attention and FFN sub-layers respectively (best viewed in color).

form or composition function is. We hypothesize that this is because the FFN learns task-specific textual patterns (Geva et al., 2021), while attention learns pairwise positional interactions which do not require large capacity for adapting to new tasks.

Is the story different when we use 0.1% parameters? In §6.3.1 we reason that prefix tuning is more expressive than adapters (attn), which, however, is not reflected in Figure 6.8. We conjecture that this is because multi-head attention is only superior when the parameter budget is small. To validate this hypothesis, we compare prefix tuning to parallel adapters when they add 0.1% of the pretrained parameters. To ablate the impact of the composition function, we also report the results of removing the gating in prefix tuning as $\boldsymbol{h} + \Delta\boldsymbol{h}$. We include the results of the multi-head parallel adapter variant (MH PA) described in §6.3.3. As shown in Table 6.7, the multi-head methods – prefix tuning and MH PA (attn) – outperform all others by at least 1.6 BLEU points when using 0.1% of the parameters. Surprisingly, reducing l from 200 to 30 only causes 0.4 BLEU loss for prefix tuning while PA (attn) loses 1.9 points. The gating composition function in prefix tuning slightly helps the results by 0.3 points. We highlight that the MH parallel adapter improves the single-headed version by 1.6 points, which again verifies the effectiveness of the multi-head formulation.

Combining the results in Figure 6.8 and Table 6.7, we conclude that *modifying head attention shows the best results when the parameter budget is very small, while the FFN can better utilize modifications at larger capacities.* This suggests that it may be effective to allocate a larger parameter budget to FFN modification instead of treating attention and FFN equally as in Houlsby et al. (2019).

6.4.5 Which Composition Function?

We have presented three composition functions in §6.3.2: simple addition (adapter), gated addition (prefix tuning) and scaled addition (LoRA). As it is unnatural to incorporate the exact gated addition into methods whose functional form does not use softmax, we examine the other two by ablating on LoRA and comparing with the proposed scaled parallel adapter (Scaled PA), we constrain modified representation to be FFN since it is generally more effective as shown in §6.4.4.

Table 6.3: Comparison of various parameter-efficient tuning methods and the proposed variants. "†" are results copied from Lewis et al. (2020a) and Liu et al. (2020b). We could not reproduce exactly the same full fine-tuning numbers with the same hyperparameters or even searching them. The reason may be the different libraries which the training code is based on – full fine-tuning is very sensitive to training hyperparameters. For the most performant methods we run with 3 random seeds and report mean and standard deviation.

Method	# params	XSum (R-1/2/L)	MT (BLEU)
Full fine-tuning†	100%	45.14/22.27/37.25	37.7
Full fine-tuning (our run)	100%	44.81/21.94/36.83	37.3
Bitfit (Ben Zaken et al., 2021)	0.1%	40.64/17.32/32.19	26.4
Prompt tuning (Lester et al., 2021)	0.1%	38.91/15.98/30.83	21.0
Prefix tuning (Li and Liang, 2021b), l=200	3.6%	43.40/20.46/35.51	35.6
Pfeiffer adapter (Pfeiffer et al., 2021), r=600	7.2%	$44.03/20.89/35.89_{\pm.13/.10/.08}$	$36.9_{\pm.1}$
LoRA (ffn), r=102	7.2%	$44.53/21.29/36.28_{\pm.14/.07/.10}$	$36.8_{\pm.3}$
Parallel adapter (PA, ffn), r=1024	12.3%	$44.71/21.41/36.41_{\pm.16/.17/.16}$	$37.2_{\pm.1}$
PA (attn, r=30) + PA (ffn, r=512)	6.7%	$44.29/21.06/36.12_{\pm.31/.19/.18}$	$37.2_{\pm.1}$
Prefix tuning (attn, l=30) + LoRA (ffn, r=102)	6.7%	$44.84/21.71/36.77_{\pm.07/.05/.03}$	$37.0_{\pm.1}$
MAM Adapter (our variant, l=30, r=512)	6.7%	$\mathbf{45.06/21.90/36.87}_{\pm.08/.01/.04}$	$\mathbf{37.5}_{\pm.1}$

Table 6.2: Results on XSum when using different composition functions. The modified representation is FFN. The bottleneck dimension $r = 512$ for (Scaled) PA and $r = 102$ for LoRA.

Method (# params)	XSum (R-1/2/LSum)
LoRA (6.1%), s=4	44.59/21.31/36.25
LoRA (6.1%), s=1	44.17/20.83/35.74
PA (6.1%)	44.35/20.98/35.98
Scaled PA (6.1%), s=4	**44.85/21.54/36.58**
Scaled PA (6.1%), trainable s	44.56/21.31/36.29

Table 6.2 reports the results on XSum. We set r as 512 for adapters and 102 for LoRA so that their tuned parameter sizes are the same. We select s based on the R-2 score on the dev set. We observe that LoRA ($s = 4$) performs better than parallel adapter. However, the advantage disappears if we remove the scaling by setting $s = 1$. Through plugging the composition function of LoRA into parallel adapter, the resulted Scaled PA improves the vanilla parallel adapter by 0.56 ROUGE-2 points. We also experiment with a learned scalar which does not give better results. Therefore, we conclude that *the scaling composition function is better than the vanilla additive one while being easily applicable.*

6.4.6 An Effective Integration by Transferring Favorable Design Elements

We first highlight three findings in previous sections: (1) Scaled parallel adapter is the best variant to modify FFN; (2) FFN can better utilize modification at larger capacities; and (3) modifying head attentions

like prefix tuning can achieve strong performance with only 0.1% parameters. Inspired by them, we mix and match the favorable designs behind these findings: specifically, we use prefix tuning with a small bottleneck dimension ($l = 30$) at the attention sub-layers and allocate more parameter budgets to modify FFN representation using the scaled parallel adapter ($r = 512$). Since prefix tuning can be viewed as a form of adapter in our unified framework, we name this variant as *Mix-And-Match adapter (MAM Adapter)*. In Table 6.3, we compare MAM adapter with various parameter-efficient tuning methods. For completeness, we also present results of other combination versions in Table 6.3: using parallel adapters at both attention and FFN layers and combining prefix tuning (attn) with LoRA (ffn) – both of these combined versions can improve over their respective prototypes. However, MAM Adapter achieves the best performance on both tasks and is able to match the results of our full fine-tuning by only updating 6.7% of the pre-trained parameters. In Table 6.5, we present the results of MAM Adapter on MNLI and SST2 as well, where MAM Adapter achieves comparable results to full fine-tuning by adding only 0.5% of pretrained parameters.

6.5 Discussion

We provide a unified framework for several performant parameter-tuning methods, which enables us to instantiate a more effective model that matches the performance of full fine-tuning method through transferring techniques across approaches. We hope our work can provide insights and guidance for future research on parameter-efficient tuning, and we also suggest these methods to perform experiments on tasks of varying resources and properties to exhibit a more complete view.

Chapter 7

Prompt Consistency for Zero-Shot Task Generalization

While the previous chapter focuses on the supervised transfer learning setting of pre-trained language models, one of the most impressive results of recent NLP history is the ability of pre-trained language models to solve new tasks in a *zero-shot* setting. To achieve this, NLP tasks are framed as natural language prompts, generating a response indicating the predicted output. Nonetheless, the performance in such settings often lags far behind its supervised counterpart, suggesting a large space for potential improvement. In this chapter, we explore methods to utilize unlabeled data to improve zero-shot performance with parameter-efficient fine-tuning.

7.1 Introduction

While the past decade has demonstrated that pretrained language models (PLMs) are powerful tools for improving generalization from training datasets to test datasets (Devlin et al., 2019b; Liu et al., 2019; Raffel et al., 2020), more recent work has shown that they can even perform *zero-shot generalization to new tasks* without any annotated examples (Brown et al., 2020; Wei et al., 2021; Sanh et al., 2021). These systems leverage natural language prompts that specify the task for the model and represent different tasks in a unified format (Liu et al., 2021b). Zero-shot task generalization suggests a path towards generic systems that perform a wide variety of NLP tasks with no annotated examples. However, while enticing conceptually, zero-shot performance often remains relatively low compared to systems trained using even a small amount of task-specific labeled data.

In this chapter, we examine methods to make PLMs better zero-shot learners using unlabeled text. Our work is motivated by consistency training methods that regularize model predictions to be invariant to perturbation (e.g. noise or paraphrasing) of the input examples. Consistency training is widely used in semi-supervised learning literature as an effective technique to utilize unannotated examples (Bach-

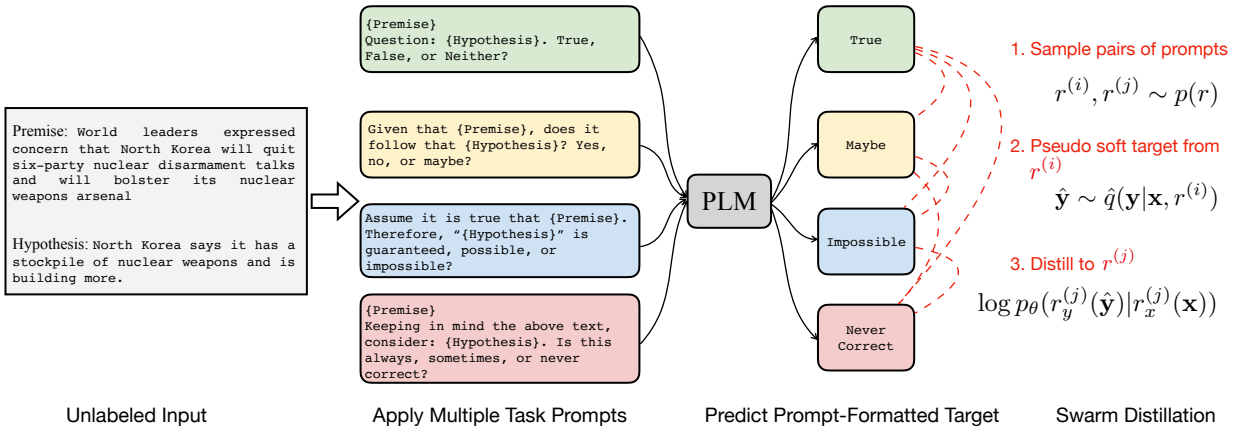


Figure 7.1: An example of the proposed approach in an NLI task. We apply multiple synonymous prompt templates to the unlabeled example, then we regularize the consistency of the predictions from different prompts, through our swarm distillation loss. Note that we are not regularizing the predicted text form $r_y^{(i)}(\mathbf{y})$ to be the same since different prompts have different target templates as shown above – we are actually regularizing the discrete labels $\mathbf{y}$ underneath to be consistent, as detailed in Eq. 7.2.

man et al., 2014; Sajjadi et al., 2016; Beyer et al., 2019; Xie et al., 2020a). It is often understood as a type of smoothness regularization or data augmentation (Xie et al., 2020a) and attains strong performance in semi-supervised learning. Instead of example-level consistency, we propose to regularize *prompt consistency*, where a model is regularized to make the same prediction across a diverse set of synonymous task prompts. Prompt consistency regularization makes sense intuitively since PLMs should be robust across synonymous prompts, whereas it is known that model predictions are empirically very sensitive to the wording of the task prompts (Jiang et al., 2020).

Specifically, we design a pairwise distillation loss that encourages consistency between every pair of prompts (Figure 6.1). We refer to our method as *swarm distillation*, and it has the advantage of being fully unsupervised, only requiring unannotated inputs. Notably, unannotated examples are often relatively easy to collect. Drafting several prompts for a task is also far cheaper than annotating labels for each example – in fact, there are already well-designed prompts available for a wide range of NLP tasks (Bach et al., 2022).

Previous work on example-level consistency regularization typically minimizes a consistency loss along with a supervised loss in a semi-supervised setting (Miyato et al., 2018; Xie et al., 2020a). Recently, Elazar et al. (2021) performed experiments optimizing a prompt consistency loss in the context of a relation prediction task, also incorporating a supervised version of the masked language model pretraining objective. In contrast, we (1) optimize a novel prompt consistency loss alone, making our approach completely unsupervised and agnostic to the model's pretraining objective, and (2) experiment on and demonstrate the practicality of such an approach for a broad variety of NLP tasks. Notably, this

unsupervised setting poses additional learning challenges: without explicit supervision, the model may suffer from catastrophic forgetting and even exhibit a form of collapse where the model always makes the same predictions for any input. To address this issue, we adopt two simple strategies: (1) we utilize parameter-efficient tuning techniques (Houlsby et al., 2019; Zhou et al., 2022) to only update a small number of extra parameters, naturally mitigating catastrophic forgetting by fixing the original PLM parameters; (2) we propose an unsupervised criterion to select the model checkpoint before it falls into a collapsed local optimum.

In experiments, we build our method on top of a state-of-the-art zero-shot task learner, T0 (Sanh et al., 2021), and validate its performance on 11 datasets from 4 NLP tasks: natural language inference, coreference resolution, word sense disambiguation, and sentence completion. We perform experiments under two scenarios: (1) training the model with unlabeled training data; or (2) tuning the model with unlabeled test inputs directly. In both settings, we show that our swarm distillation method improves the accuracy of the 3B-parameter T0 model on 9 out of 11 datasets by up to 10.6 absolute points. We further scale model size up to 11B parameters, and demonstrate that our approach outperforms the 11B-parameter T0 model on 4 out of 4 datasets. Remarkably, analysis implies that these gains are often possible with only tens of examples, suggesting a small computation overhead. Our code is available at https://github.com/violet-zct/swarm-distillation-zero-shot.

7.2 Prompt-based Zero-Shot Task Generalization

Given a task where the input is denoted as $\mathbf{x} \in \mathcal{X}$ and the goal is to predict $\mathbf{y} \in \mathcal{Y}$, we focus on the zero-shot task generalization setting: we aim to feed a PLM with $\mathbf{x}$ to predict $\mathbf{y}$, where the PLM is never trained on the specific task to be performed. Zero-shot task generalization goes beyond traditional dataset generalization, as the model must generalize to new functions $f : \mathcal{X} \rightarrow \mathcal{Y}$ as opposed to new input examples, $\mathbf{x}$. Recently, the development of prompting methods has advanced zero-shot task generalization by representing different tasks in a unified format (Liu et al., 2021b), and several prompt-based approaches have attained reasonable zero-shot performance (Brown et al., 2020; Sanh et al., 2021; Wei et al., 2021).

A prompt r consists of an input template r_x, an output template r_y, and metadata to re-format the original $\mathbf{x}$ and $\mathbf{y}$ into new prompt-formatted input and target, $r_x(\mathbf{x})$ and $r_y(\mathbf{y})$. For example, as shown in Figure 6.1, in a natural language inference task to predict "entailment", "neutral", or "contradiction" between two texts, the input includes the field `Premise` and `Hypothesis` and the target consists of the field `Label`. An input template could be `Given that {Premise}, does it follow that {Hypothesis}? Yes, no, or maybe?`, and the target template is `Choices[{label}]`. Here `Choices` is the metadata that is a list containing [`Yes`, `Maybe`, `No`] to correspond to the digit labels. We note that such metadata is prompt-specific and can differ with differ-

ent prompts for the same task – for instance, in Figure 6.1 each prompt actually has a different `Choices` list from others; the `Choices` list of the first prompt on the top is [`True`, `False`, `Neither`]]. In prompt-based approaches the PLM models the conditional probability $q(\mathbf{y}|\mathbf{x}, r)$ through $p_\theta(r_y(\mathbf{y})|r_x(\mathbf{x}))$ where θ denotes the model parameters. In classification tasks where $\mathcal{Y}$ is a finite label set, $q(\mathbf{y}|\mathbf{x}, r)$ is normalized over the possible labels at inference time to predict $\mathbf{y}$:

$$q(\mathbf{y}|\mathbf{x}, r) = \frac{p_\theta(r_y(\mathbf{y})|r_x(\mathbf{x}))}{\sum_{\mathbf{y}' \in \mathcal{Y}} p_\theta(r_y(\mathbf{y}')|r_x(\mathbf{x}))}. \tag{7.1}$$

In generation tasks where $\mathcal{Y}$ is an infinite sequence space, the target template is typically instantiated as the target itself, i.e. $p_\theta(r_y(\mathbf{y})|r_x(\mathbf{x})) = p_\theta(\mathbf{y}|r_x(\mathbf{x}))$, then the output can be directly decoded through sequence decoding approaches. Through designing such prompts for each task, all NLP tasks share the same data format, and models trained on one task may generalize to others.

7.3 Prompt Consistency Training

7.3.1 Problem Definition

In this chapter, we aim to explore unannotated examples to improve prompt-based zero-shot task generalization. Formally, we are given an unlabeled dataset in the task of interest $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, and we assume the dataset has K different prompts, $\{(r_x^{(1)}, r_y^{(1)}), \cdots, (r_x^{(K)}, r_y^{(K)})\}$. Our goal is to utilize these resources and adapt a PLM to predict $r_y(\mathbf{y})$ conditioned on $r_x(\mathbf{x})$. We note that unlabeled input and a diverse set of prompts are not difficult to collect practically – the inputs to most NLP tasks are plain text such as reviews, documents, or questions, and empirically our method is effective even with tens to hundreds of unlabeled examples as we will show in §7.4.4; drafting prompts for each task is much easier than annotating labels for each example, in fact, the community efforts have pushed out a Public Pool of Prompts (P3)[1] that contains thousands of prompts for hundreds of NLP datasets already (Sanh et al., 2021). In this chapter, we are going to focus our experiments on a subset of datasets supported by P3.

7.3.2 The Prompt Consistency Loss

Consistency regularization is a method that creates different views (e.g. paraphrases of text) of the input and regularizes the outputs to be close to each other, and has achieved significant success in semi-supervised learning (Clark et al., 2018; Xie et al., 2020a,b). While previous methods use an additional module to perturb each example and then optimize example-level consistency, we propose to optimize prompt-level consistency which (1) is conceptually simple, and (2) can mitigate the fact that the predictions of PLMs are typically inconsistent with different prompts for the same task (Jiang et al., 2020; Elazar et al., 2021). Intuitively, we propose to regularize the predictions of different prompts for a given input to

[1] https://github.com/bigscience-workshop/promptsource

be close to each other, using a pairwise distillation loss to draw the predictions from one prompt closer to those from the other. Concretely, we randomly sample a few pairs of prompts and distill the pseudo target $\hat{\mathbf{y}}$ from one prompt $r^{(i)}$ to the other prompt $r^{(j)}$, as illustrated in Figure 6.1. The loss function is defined as:

$$\mathcal{L} = -\mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}\mathbb{E}_{r^{(i)},r^{(j)}\sim p(r)}\mathbb{E}_{\hat{\mathbf{y}}\sim\hat{q}(\mathbf{y}|\mathbf{x},r^{(i)})} \log p_\theta(r_y^{(j)}(\hat{\mathbf{y}})|r_x^{(j)}(\mathbf{x})), \tag{7.2}$$

where $p_d(\mathbf{x})$ is the empirical data distribution, $p(r)$ is a uniform distribution over possible prompts in the prompt set, and $\hat{q}(\mathbf{y}|\mathbf{x}, r)$ is the conditional target distribution defined as in Eq. 7.1 but with a stopping gradient operator. We do not propagate gradients to $\hat{q}(\mathbf{y}|\mathbf{x}, r^{(i)})$ following Miyato et al. (2018) and Xie et al. (2020a).[2] Stopping the gradient of one side in a pairwise consistency loss is also shown to help mitigate the collapse issue where all inputs lead to the same predictions (Chen and He, 2021). Different from traditional distillation that distills from a teacher model to a student model (Hinton et al., 2015), or previous consistency training that a single teacher distills to several students (Clark et al., 2018; Xie et al., 2020a), we perform distillation among a swarm of prompts where each prompt is a teacher and student at the same time, thus we term our method as *swarm distillation*. In our implementation, we approximate the expectation over the paired prompts $(r^{(i)}, r^{(j)})$ with k randomly sampled pairs instead of enumerating all pairs for training efficiency.

Prompt consistency is related to example-level consistency when viewing different prompt-formatted inputs $r_x^{(i)}(\mathbf{x})$ as separated views of the same example, thus our swarm distillation approach shares spirit with previous work on example-level consistency training and can be understood similarly from the perspective of unsupervised data augmentation, smoothness regularization, or label propagation (Xie et al., 2020a). In this chapter, we focus on classification tasks where $\mathcal{Y}$ is a finite label set, while Eq. 7.2 can be directly applied to sequence generation tasks as well with sequence distillation (Kim and Rush, 2016b).

Our approach differs from previous consistency training methods which often combine an unsupervised consistency loss with a supervised loss in a semi-supervised setting (Miyato et al., 2018; Clark et al., 2018; Xie et al., 2020a). Elazar et al. (2021) try to improve prompt consistency for a relation filling task with a pairwise two-sided KL divergence loss, while they also optimize a supervised version of the original PLM objective that turns out to be important. In contrast, our approach minimizes the swarm distillation loss in Eq. 7.2 alone, and therefore is completely unsupervised and agnostic to the pretraining objective. However, this setting also poses challenges in learning, which we discuss next.

7.3.3 Training

Being trained without explicit supervision, the PLM may forget what it learns during pretraining since the unsupervised consistency loss is different from the pretraining objective. Also, we note that prompt

[2] Note that $\hat{q}(\mathbf{y}|\mathbf{x}, r)$ still changes as we train the model.

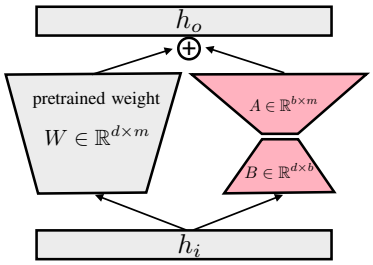


Figure 7.2: A diagram of LoRA in the FFN sublayer. Only the LoRA parameters, A and B, are updated during training while other parameters are fixed.

consistency may be achieved with a trivial solution – if the predictions from each example and each prompt collapse to the same label then maximal consistency among prompts can be reached. To mitigate such catastrophic forgetting and collapse issues, we propose two techniques:

Parameter-efficient tuning: Parameter-efficient tuning methods naturally mitigate catastrophic forgetting and collapse through fixing the original PLM parameters. Our unsupervised setting only provides weak learning signals with a relatively small data set, thus we expect that a high-capacity parameter-efficient tuning method, e.g. MAM Adapters in §6, will be less useful. Specifically, we use LoRA (Hu et al., 2021), a low-rank adaptation method for PLMs, which is introduced in §6.2.2. We briefly recap LoRA here. As shown in Figure 7.2, LoRA learns a low-rank approximation of the pretrained matrix updates: given a pretrained weight matrix $W \in \mathbb{R}^{d\times m}$, LoRA learns to update it as $W \leftarrow W + \alpha BA$, where $B \in \mathbb{R}^{d\times b}$, $A \in \mathbb{R}^{b\times m}$ are low-rank matrices and α is a hyperparameter, and only B and A are updated during training. $b \ll d$ is referred to as the *bottleneck dimension*. Following Zhou et al. (2022), we apply LoRA to the feed-forward weight matrices of every layer in the pretrained transformer (Vaswani et al., 2017) model. We emphasize that B (or A) needs to be initialized as a zero matrix to ensure the output distribution after adding LoRA layers is the same as the original PLM before training, otherwise, the zero-shot ability of PLMs would be broken upon initialization and there is no supervision to learn it back. In our preliminary experiments, we found that LoRA is less likely to suffer from collapse, while we still observe collapse sometimes. This motivates us to develop a criterion to select the model checkpoint before the model falls into a collapsed local optimum, which we describe next.

Unsupervised model selection criterion: In supervised learning, model selection is typically performed on a held-out validation set using supervised metrics. However, our zero-shot setting requires to develop an unsupervised selection criterion. Intuitively, it is straightforward to use a consistency metric as the criterion since we are optimizing towards prompt consistency, but a naive consistency metric would reach its maximum when the model is collapsed. Therefore, we would like to have a metric that encourages consistency but simultaneously penalizes collapse. With that in mind, we focus on Fleiss' kappa (Fleiss, 1971), a commonly used metric to assess the reliability of agreement between a fixed

number of raters. In our setting, Fleiss' kappa expresses the extent to which the amount of agreement among prompts exceeds what would be expected if all prompts made their predictions according to the marginalized distribution of labels. This design naturally penalizes collapse and is computing a notion of "relative consistency". Formally, let n_{ij} be the number of prompts that predict the j-th label for the i-th example. There are a total of NK predictions where N is the number of examples and K is the number of prompts. Given an example $\mathbf{x}_i$, the agreement probability p_i is to compute how many prompt pairs are in agreement, divided by the number of all possible pairs:

$$p_i = \frac{1}{K(K-1)} \sum\nolimits_j n_{ij}(n_{ij}-1), \tag{7.3}$$

then p_i is averaged across examples to obtain the "absolute consistency":

$$\bar{P} = \frac{1}{N} \sum\nolimits_{i=1}^{N} p_i. \tag{7.4}$$

It can be seen that $\bar{P}$ is maximized in the case of collapse. However, Fleiss' kappa considers the marginalized distribution of labels: how likely are two prompts consistent if they make predictions randomly according to the marginalized label distribution? This chance probability $\bar{P}_e$ is:

$$\bar{P}_e = \sum\nolimits_j p_j^2, \qquad p_j = \frac{1}{NK} \sum\nolimits_{i=1}^{N} n_{ij}, \tag{7.5}$$

where p_j represents the marginalized distribution of labels, i.e. $p(\mathbf{y} = j)$. $\bar{P}_e$ is large when collapse happens and one label dominates in the entire corpus. Finally, Fleiss' kappa is computed as:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}, \tag{7.6}$$

where $1 - \bar{P}_e$ gives the degree of consistency that is attainable above chance, $\bar{P} - \bar{P}_e$ gives the degree of consistency actually achieved above chance. κ ranges from -1 to 1. Eq. 7.6 naturally penalizes collapse, and in our experiments, we always observe a monotonic decrease of κ when collapse happens. Therefore, we select the model checkpoint after which κ monotonically decreases.[3] We emphasize that we perform validation on the data that the model is trained on and do not require an additional development dataset.

7.4 Experiments

Our experiments below are designed to (1) measure whether swarm distillation is able to improve zero-shot task generalization; and (2) analyze how much resource (number of prompts and unlabeled examples) our method demands.

[3] In most of the settings, this criterion is equivalent to using maximal κ as the criterion, except for few cases where the beginning of training exhibits large fluctuations in κ.

Task	Dataset	T0-3B		Self Dist. (train)		Swarm Dist. (train)		Swarm Dist. (test)	
		Ens.	Med.	Ens.	Med.	Ens.	Med.	Ens.	Med.
NLI	RTE	64.6	64.1	64.9±0.2	63.8±0.1	75.2±0.8 ↑10.6	73.9±0.8 ↑9.8	75.2±0.2 ↑10.6	73.5±0.1 ↑9.4
	CB	46.4	50.0	47.0±1.0	49.4±2.7	47.6±1.0 ↑1.2	48.2±0.0 ↓1.8	46.4±0.0 ↑0.0	48.8±1.0 ↓1.2
	ANLI R1	34.6	33.7	36.1±0.1	34.7±0.1	38.4±0.5 ↑3.8	35.7±0.4 ↑2.0	38.5±0.3 ↑3.9	35.7±0.5 ↑2.0
	ANLI R2	33.7	33.4	35.3±0.1	33.2±0.2	37.9±0.8 ↑4.2	36.6±0.5 ↑3.2	37.7±0.2 ↑4.0	35.4±0.4 ↑2.0
	ANLI R3	34.7	33.3	33.1±0.0	33.8±0.2	34.0±0.3 ↓0.7	34.6±0.1 ↑1.3	34.1±0.2 ↓0.6	33.5±0.0 ↑0.2
Compl.	COPA	78.0	79.0	82.3±0.6	78.2±0.3	82.7±0.6 ↑4.7	79.0±0.5 ↑0.0	83.0±1.0 ↑5.0	79.7±0.6 ↑0.7
	HellaSwag	27.8	27.5	32.5±0.2	32.7±0.3	34.2±0.2 ↑6.4	33.4±0.2 ↑5.9	33.7±0.6 ↑5.9	33.2±0.3 ↑5.7
	Story Cloze	86.5	85.1	89.6±0.0	88.7±0.0	–	–	87.3±0.1 ↑0.8	86.9±0.2 ↑1.8
Coref.	Wino.	50.9	50.5	51.1±0.1	50.7±0.1	52.0±0.3 ↑1.1	51.4±0.0 ↑0.9	52.1±0.3 ↑1.2	51.2±0.2 ↑0.7
	WSC	69.2	64.4	69.2±0.0	64.6±0.3	58.3±1.1 ↓10.9	59.3±2.0 ↓5.1	57.7±0.0 ↓11.5	58.8±0.6 ↓5.6
WSD	WIC	50.3	50.4	50.3±0.0	50.3±0.0	55.4±1.1 ↑5.1	54.4±0.7 ↑4.0	55.5±0.8 ↑5.2	54.8±0.5 ↑4.4

Table 7.1: Accuracy results on the validation set of 11 NLP datasets based on the T0-3B model. Swarm Distillation (train) and Swarm Distillation (test) use the unlabeled training split and validation split of datasets to train the model respectively, corresponding to training-time and test-time tuning. The Story Cloze dataset does not have a training split and its self distillation results are from tuning on the validation split. We report the mean and std across 3 random runs, and also denote the absolute accuracy change compared to the T0-3B baseline.

7.4.1 General Setup

Datasets: Following Sanh et al. (2021), we evaluate our method on 11 NLP datasets across 4 unseen tasks. They are (1) natural language inference: ANLI (Nie et al., 2020) (there are three versions of ANLI with different levels of difficulty, which we denote as ANLI R1/R2/R3), CB (De Marneffe et al., 2019), RTE (Wang et al., 2019); (2) sentence completion: COPA (Roemmele et al., 2011), HellaSwag (Zellers et al., 2019), Story Cloze (Mostafazadeh et al., 2016); (3) coreference resolution: WSC, Winogrande (Levesque et al., 2012); and (4) word sense disambiguation: WIC (Pilehvar and Camacho-Collados, 2019). We access them using Hugging Face Datasets (Lhoest et al., 2021) and most of them are from the SuperGLUE benchmark (Wang et al., 2019). All of these datasets are classification-based, predicting a discrete label from a finite set. Each of these datasets has a diverse set of prompts provided by the Public Pool of Prompts (Sanh et al., 2021) The number of prompts ranges from 4 to 15. Please refer to Appendix .18 for detailed statistics of these datasets.

Setup: We build our method on top of the PLM T0 (Sanh et al., 2021). T0 is an adapted version of the pretrained T5 model (Raffel et al., 2020) that is continually trained on multiple tasks with supervised, prompt-formatted examples. T0 outperforms GPT3 (Brown et al., 2020) and demonstrates state-of-the-

art performance in zero-shot task generalization. All the tasks that we are studying are not included in T0's training data. We focus our major study on the T0 model version with 3 billion parameters (T0-3B), while we also include results using the largest T0 model with 11 billion parameters (T0-11B) on some datasets, due to the high computational cost of training T0-11B. We tune the hyperparameters (e.g. the optimization hyperparameters) on the RTE dataset with its validation set and fix them for all other datasets. During optimization of Eq. 7.2, we randomly sample a batch of k pairs of prompts where k is the largest number that our GPU memory can fit and accumulate gradients for one update. We use a bottleneck dimension of 1 for LoRA. Complete setup details can be found in Appendix .19.

7.4.2 Evaluation

Metrics: We use accuracy as the metric for all datasets. We report two different types of accuracy given that we have multiple prompts. The *ensemble accuracy* (Ens.) averages the output distributions of multiple prompts and makes predictions according to it. Ensembling multiple prompts has been explored before and found superior to using a single prompt (Jiang et al., 2020; Qin and Eisner, 2021). The *median accuracy* (Med.) within the set of prompts serves as a proxy for the expected performance when users specify a single prompt and input a prompt-formatted example. As our approach assumes availability of a set of prompts for the downstream task, and it is relatively cheap to craft several prompts for a task, ensemble prediction is the better option given input $\mathbf{x}$, and it does empirically yield higher accuracy overall than the median for both the baseline and our method. Therefore, we will report both numbers but mainly discuss ensemble accuracy. We compute these metrics on the validation split of each dataset. We run the experiments with 3 random seeds and report the mean and standard deviation.

Evaluation scenarios: We provide our methods with different unlabeled sources which lead to two practical scenarios during evaluation: (1) *training-time tuning*: we use the unlabeled training split from the corresponding dataset to train the model. This is similar to traditional settings where training and test data are different; and (2) *test-time tuning* (Sun et al., 2020; Wang et al., 2021b): we directly adapt the PLM on the test data. This setting is reasonable, as we will always have access to the test inputs at test time. Intuitively, the unlabeled test sample $\mathbf{x}$ often provides hints about the distribution it was drawn, suggesting that we may update the model before making the prediction. This scenario is attractive since it alleviates the common distribution mismatch issue when there is a distribution shift between the training and test data. Compared to training-time tuning, test-time tuning typically uses less unlabeled data in our experiments since it uses the validation split itself. In the major experiments, we focus on the offline test-time tuning where we assume access to the entire test data[4] and train our approach on all test examples, while in §7.4.4 we will discuss the potential for online adaptation where data arrives in a stream.

[4]To clarify, test data is not the test split of the dataset, but the data that we evaluate on, i.e. the validation split.

	T0-11B		Swarm Dist.	
Dataset	Ens.	Med.	Ens.	Med.
WSC	63.5	62.5	65.4 ↑1.9	62.0 ↓0.5
RTE	83.8	82.0	86.6 ↑2.8	85.0 ↑3.0
HellaSwag	34.4	33.6	45.0 ↑10.6	43.0 ↑9.4
WIC	57.2	56.8	62.1 ↑4.9	60.7 ↑3.9

Table 7.2: Accuracy on the validation set based on T0-11B.

	RTE	CB	ANLI R1	ANLI R2	ANLI R3	COPA	HS	Story.	Wino.	WSC	WIC	Avg.
T0-3B	0.644	0.440	0.221	0.189	0.170	0.586	0.164	0.765	0.396	0.255	0.398	0.384
Swarm Dist.	0.662	0.254	0.145	0.156	0.177	0.699	0.402	0.862	0.509	0.462	0.517	**0.440**

Table 7.3: Fleiss' kappa on 11 datasets based on T0-3B. Swarm distillation is trained on training split of the respective dataset.

Baselines: As far as we know, there is no prior work studying unsupervised approaches for this prompt-based task generalization setting, thus T0 is the main baseline that we compare our approach against. However, we still implement an ablation baseline, self distillation, to separate the improvement brought by optimizing prompt consistency and the improvement brought by pseudo-label distillation. Specifically, self distillation minimizes the same loss as in Eq. 7.2 but with $r^{(i)} = r^{(j)}$ – instead of pairwise distillation, the prompt always distills its own prediction to itself. This baseline can be viewed as a prompt version of self-training, which has proven to effectively utilize unlabeled data and achieved success in various applications (He et al., 2020; Xie et al., 2020b; Zhang et al., 2020c). For simplicity, we report self distillation results in the training-time tuning setting only.

7.4.3 Results

How well does swarm distillation work? We first compare swarm distillation against the T0-3B baseline. We run our own evaluation using the released T0 weights to obtain the T0 baseline accuracy.[5] As shown in Table 7.1, the ensemble accuracy of swarm distillation exceeds the T0-3B baseline on 9 out of 11 datasets in both training- and test-time tuning settings. Particularly, our approach improves the zero-shot performance on RTE by around 10 absolute points in all cases. Our approach slightly hurts ensemble accuracy of ANLI R3 and median accuracy of CB, but is overall comparable on these two datasets. Compared to self distillation, swarm distilltion outperforms it on 9 out of 11 datasets in terms of ensemble accuracy, by up to 10.3 absolute points. These results further confirm the effectiveness of encouraging

[5]We are able to reproduce the numbers reported in Sanh et al. (2021), except for COPA where our T0 median number is higher than the originally reported one.

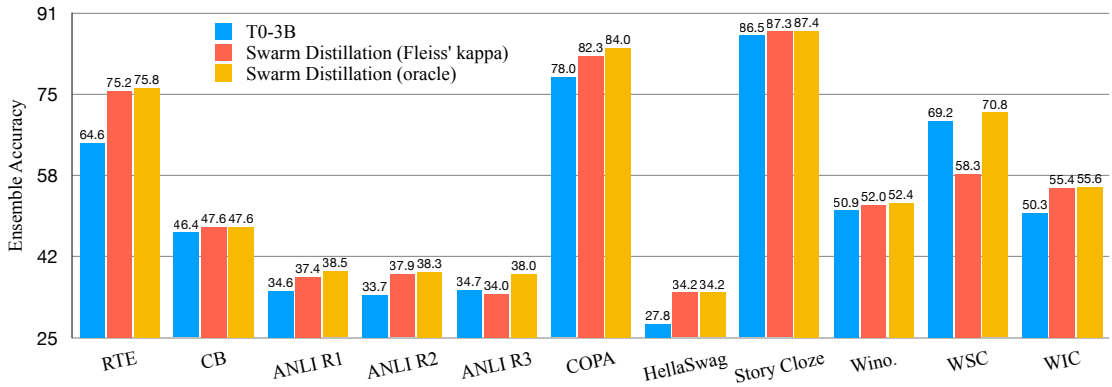


Figure 7.3: Analysis results to compare the model checkpoints selected by the unsupervised criterion Fleiss' kappa with the oracle model checkpoints selected by validation accuracy.

prompt consistency. We note that swarm distillation severely fails on WSC with a 10-point accuracy decrease compared to both T0 and self distillation, this is because Fleiss' kappa selects a bad model checkpoint, while our approach actually improves the performance on WSC in the middle of training as we will discuss more in §7.4.4. Although it may be argued that swarm distillation only works when the base PLM can attain reasonable performance in the first place, notably, our approach improves T0-3B greatly on several datasets where T0-3B only shows nearly chance accuracy, such as ANLI R1/R2/R3 (3 labels), HellaSwag (4 labels), Winogrande (2 labels), and WIC (2 labels). In addition, we observe that swarm distillation in the test-time tuning setting performs comparably well to the training-time one despite using much less training data, as shown in Appendix .18. It is worth noting that prompt-based zero-shot task generalization is challenging, for example, T0 with even 11 billion parameters reports a median accuracy of only $\sim$ 40 on ANLI R1/R2/R3, 33.7 on HellaSwag, and 57.2 on WIC (Sanh et al., 2021). These numbers are surely still far from satisfactory, yet we hope to inspire future research to explore prompt-formatted, unlabeled data to build better zero-shot learners.

Scaling to 11B parameters: We now evaluate our method based on the largest version of T0 model, T0-11B. T0-11B is a very powerful zero-shot baseline that greatly outperforms GPT3 with 175 billion parameters. Due to the expensive computation to train T0-11B, we use one dataset per task, a total of 4 datasets as our benchmark, and only run with one random seed in the test-time tuning setting. Results are shown in Table 7.2. Swarm distillation outperforms T0-11B on all 4 datasets in terms of ensemble accuracy, and notably, improves the ensemble accuracy on HellaSwag from 34.4 to 45.0 without any annotation. Table 7.1 and Table 7.2 demonstrate the effectiveness of swarm distillation across different model sizes.

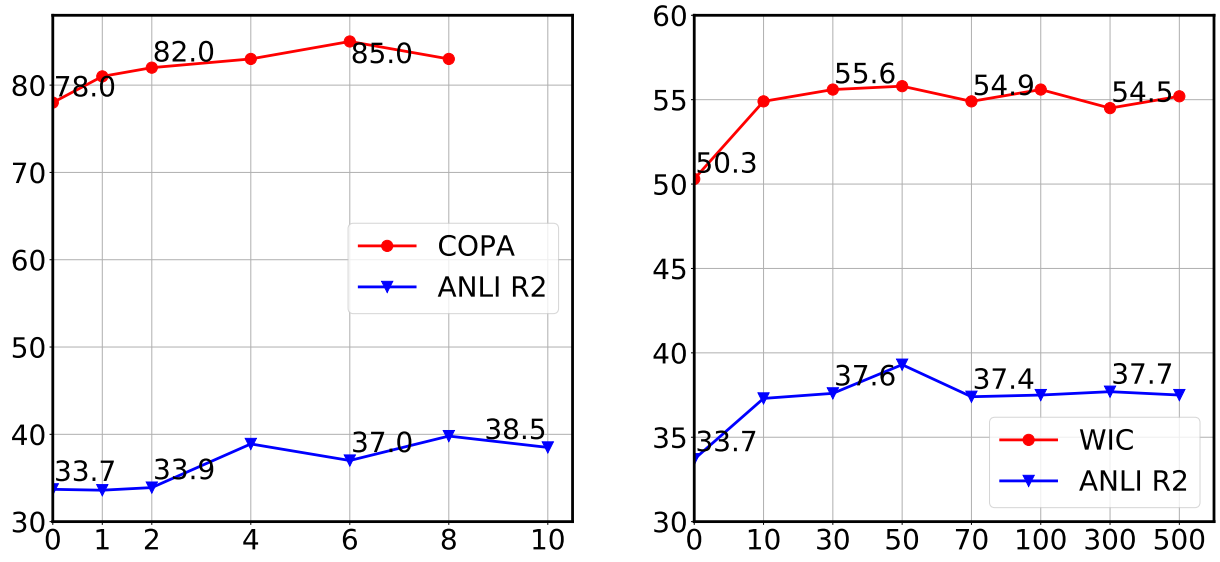
7.4.4 Analysis

Are predictions more consistent across different prompts after swarm distillation? We are interested to know whether the gains of swarm distillation are attained together with more consistent predictions across different prompts. To this end, we report Fleiss' kappa, a commonly used metric for group agreement as detailed in §7.3.3. Results are shown in Table 7.3. Fleiss' kappa on 8 out of 11 datasets increases after swarm distillation, which boosts the averaged Fleiss' kappa of T0-3B by 14.6% relatively. This implies that swarm distillation facilitates prompt consistency, and potentially improves the robustness of PLMs to different wording of prompts.

Does the unsupervised criterion select the best model checkpoint? In §7.3.3, we discussed using Fleiss' kappa to select the best model checkpoint for evaluation, here we report the oracle accuracy numbers obtained by selecting the model checkpoint with the best validation accuracy, and compare it to the one selected by Fleiss' kappa. We compare the ensemble accuracy using T0-3B in the training-time tuning setting, with results in Figure 7.3. On most of the datasets, Fleiss' kappa is able to achieve numbers close to the best ones. On all 11 datasets, our oracle number outperforms the T0-3B baseline. In Table 7.1 we show that swarm distillation hurts the performance on WSC a lot, while in Figure 7.3 swarm distillation (oracle) in fact outperforms T0-3B, implying that the issue lies on model selection. Therefore, swarm distillation could potentially work better if an annotated dev set is available or when it is combined with other techniques in few-shot learning settings, where good checkpoints may be selected out more easily.

How many prompts do we need? Our approach requires a diverse set of prompts to regularize prompt consistency. Here we perform ablation experiments to understand the effect of the number of prompts on the performance. We take COPA and ANLI R2 as example datasets which have 8 and 15 prompts, respectively. We then vary the number of available prompts by randomly sampling a subset of prompts before training. We report the ensemble accuracy of swarm distillation (train) in Figure 7.4a. On both COPA and ANLI R2, we observe gains as we increase the number of prompts from 0 (0 means the baseline), yet the performance saturates very quickly and relatively stabilizes when we provide 4 prompts. This implies that swarm distillation is not prompt-hungry and could work well with a small number of prompts. We note the with one prompt here Eq. 7.2 degenerates to a weaker version of self distillation compared to the one in Table 7.1 – self distillation in Table 7.1 utilizes all prompts during training while we assume access to only one prompt here.

How many unlabeled examples do we need? We measure the effect of unlabeled data size. Specifically, we randomly sample a subset of examples from the train split for training and report results on the entire validation dataset. Results on WIC and ANLI R2 are shown in Figure 7.4b. Notably, swarm



(a) Accuracy v.s. #prompts

(b) Accuracy v.s. #examples

Figure 7.4: Ensemble accuracy of swarm distillation on three example datasets, demonstrating the effect of prompt size and unlabeled data size. The PLM is T0-3B.

distillation is able to outperform the baselines (#examples=0) by a large margin on both datasets with only 10 unlabeled examples, and the performance starts to saturate quickly afterward. These results suggest that swarm distillation is not data-hungry and works reasonably well with few *unlabeled* examples, allowing swarm distillation to remain as a relatively light approach while typical unsupervised training (e.g. pretraining) often requires a large amount of data and computation. Also, we argue that the phenomenon demonstrated in the results implies that swarm distillation may be applied to the online setting of test-time tuning, where the batches of test data arrive in a stream. Online test-time tuning is a practical setting in real life, and we leave the study of swarm distillation in this setting as future work.

7.5 Discussion

In this chaper, we explore prompt consistency regularization to make PLMs better zero-shot learners. Our approach utilizes unlabeled examples to attain zero-shot gains. While we use it in a post-adaptation way to adapt PLMs with the proposed swarm distillation loss alone, our regularization loss could be potentially combined with the pretraining objectives in the pretraining stage, with the multi-prompt training loss (Sanh et al., 2021; Wei et al., 2021), or even with annotated data in few-shot learning settings. Combining the swarm distillation loss with these other losses may easily bypass the model collapse issue since the other loss typically discourages the collapsed local optimum. The potential applications of unsupervised swarm distillation on sequence generation tasks are also worth studying in the future.

Chapter 8

Conclusions and Future Directions

This thesis has tried to tackle distribution shift in NLP from different perspectives, focusing on the detection of hallucination errors made in neural sequence generation systems, and the central problem of learning models that are robust to distribution shift in the real world. We summarize the contributions of this thesis below:

- We made the first step towards detecting hallucinated content at a fine-grained level (token-level) in the outputs of conditional neural generation (Chapter 3). This hallucination detection tool also serves as a filter for noisy parallel training data, which can be broadly used in many data augmentation settings where the collected data is not clean (e.g. model generated or crawled from web). We empirically show that using our proposed truncated loss that exclude hallucinated tokens can bring significant gains on low-resource machine translation tasks.
- We developed and improved group distributionally robust optimization methods to deal with subpopulation shift. In particular, we found that group DRO fails if there are no clean group partitions of training data and we proposed a more flexible uncertainty set to overcome this issue (Chapter 4). Furthermore, we developed an efficient learning objective based on group DRO to alleviate the data balance across language pairs in multilingual neural machine translation (Chapter 5). It is also worth noting that we were among the first to use group DRO algorithms in large-scale real-world problems.
- We proposed methods that can more effectively adapt existing large-scale pre-trained language models to downstream tasks. Specifically, we connected several best-performing parameter-efficient transfer learning approaches of PLMs and proposed a new state-of-the-art method under our unified framework (Chapter 6). This framework provides insights for better understanding and developing parameter-efficient fine-tuning methods. Furthermore, we proposed a novel learning paradigm of continually adapting PLMs on unannotated examples for better zero-shot task generalization (Chapter 7). We obtain the state-of-the-art zero-shot task generalization performance

across 11 NLP datasets and show that our model is more robust to the variations of inputs in terms of producing consistent predictions.

As NLP systems are increasingly becoming parts of many real-world applications, their reliability and robustness are keys to building and deploying systems that users can trust. Towards this goal, we still face many challenges and open research questions are worth further investigation along this line.

- **First, interpretability matters especially in this era of increasingly larger models.** Interpretability includes knowing *why* a model makes certain predictions given a specific input and how it relates to the changes of model internal states. Only when we can interpret the model's behavior, can we know when it will fail and how we can fix it. To identify the salient input tokens that are responsible for some prediction, one class of prominent approach is exploiting gradient-based metrics (Li et al., 2016; Shrikumar et al., 2016) or attention weights-based metrics (Vashishth et al., 2019). Some recently works in causality attempt to interpret the predictions using counterfactual examples or manipulating the representation of the text (Feder et al., 2021; Karimi et al., 2021). Beyond the efforts to explain models' predictions, we also should call for more research work on "knows *when* the model does not know" in the future, which allows the deployment of a self-conscious system that knows when to "step back" at test time.
- **Second, we should embrace a dynamic evaluation setup in more and more NLP tasks.** It is great to see that recently there are researchers working on temporal generalization of pre-trained models in language modeling (Lazaridou et al., 2021) and knowledge probing (Dhingra et al., 2022). It is necessary to expand and develop such dynamic evaluation setups for other tasks such that we can evaluate or even learn a model in a dynamically changing environment. Such learning environment can be multimodal and experience-grounded, which is a combination of reinforcement learning, unsupervised learning and supervised learning.
- **Third, it is critical to design models and new learning paradigms that adapts to constantly changing world knowledge and environments.** First, knowledge representation is important, which has been explored in several previous works (Hitzler and Sarker, 2022; de Jong et al., 2021; Lewis et al., 2021b). This further affects how efficient it is for the model to interact with the knowledge base in terms of the efficiency in knowledge storage, retrieval and updates. The recent trends of combining parametric model with non-parametric knowledge base (Guu et al., 2020; Khandelwal et al., 2019, 2020; Borgeaud et al., 2021; Lewis et al., 2020b) not only alleviates the demand of large model parameters but also provides the possibility of utilizing future knowledge. We expect to see more powerful and efficient semi-parametric model in the future. Finally, We would like to design models with the ability of continual learning such that it can constantly update itself every time new examples are observed. We hope our work (Chapter 7) for zero-shot task generalization can pioneer this line of research. It is also worth noting that the sparse

models (e.g. Mixture-of-Experts) (Fedus et al., 2021; Zoph et al., 2022) and parameter-efficient tuning methods provide natural modular designs and could potentially be used in the future for efficient continual learning. We hope our works in Part III can provide some insights on using parameter-efficient tuning methods for this direction.

Appendix I: Hallucination Detection for Conditional Sequence Generation

.1 Human Evaluations

Setup We asked three bilingual speakers to annotate the Chinese-to-English evaluation set $\mathcal{D}_{eval}$, which is composed of 150 sentences from the test set of Zh-En multi-domain dataset (Wang et al., 2020e), and 100 sentences from the COVID-19 translation benchmark dataset (Anastasopoulos et al., 2020) – TICO. TICO contains 6 finegrained domains including *Wikisource, Wikivoyage, Wikinews, CMU, PubMed* and *Wikipedia*. we randomly sample 25 examples from each of the four domains – Wikisource, Wikinews, CMU and PubMed, use these 100 samples for evaluation. We train two varieties of models: a standard base Transformer Seq2Seq model and a model that finetunes the MBART (Liu et al., 2020c) model on the training data from $\mathcal{D}_{train}$. In the human evaluation, three bilingual annotators were presented the Chinese source sentence, the English reference sentence and the MT model generation.

Annotation Guidelines and Process We conducted the pilot study and practice sessions with annotators before annotating the final blind test set $\mathcal{D}_{eval}$. The pilot study was performed on a different evaluation set and we performed analysis on them. Then we conducted an education session with evaluators to make sure that they can fully understand and follow the guidelines. We find that it is important to define a clear workflow for annotators to execute. In the final evaluation, we ask each annotator to read the tokens in the sentence carefully and check if they can be supported by the source sentence in the following order:

(1) If there are tokens (or the entire sentence) that cannot be supported by the source, label all the span(s) with color and mark the sentence as a hallucinated one;

(2) If the annotator can not understand the entire translation, mark the sentence as incomprehensible;

(3) If all the tokens in the translation can be entailed from the source, mark the sentence as a faithful one.

We shuffled the order of sentences so that annotators did not know which translation model was used (TranS2S or MBART). Besides, we made out the following guidelines to help annotators identify

hallucinated spans and distinguish bad translations from hallucinated ones: (1) If a machine generation contains hallucinations, we ask annotators to minimally mask spans of words as hallucinations such that deleting these spans or replacing these spans with other words can dehallucinate the generation (make the generation a faithful one to the source input). For example, if T =“John likes Mary, but Mary does not like John.” and G =“John likes Mary, and Mary likes John.”, “and" and “likes" in the latter part of G should be marked as hallucinations. (2) We ask annotators not to consider the domain of sentences when marking hallucinations. For examples, if S=“今天我的胸部非常痛。" (Chinese), T=“My chest hurts badly today." and G=“My breast hurt badly today.", in this case, both the reference T and the MT G are valid translations of the source sentence because the word “胸部" in the source is a polysemy. Without considering the domain that sentences come from, the generation is a faithful one. (3) We ask annotators not to be “harsh", e.g. if a capitalized word in the reference is lowercased in the translation, we ask them not to mark it as hallucination under the rule that hallucinations should only be considered by the meaning of words and whether they are faithful to the source, instead of the surface form.

Note that annotations are performed on the raw sentences, i.e. punctuation marks can also be labeled as hallucinations along with the span and we did not apply special treatments to them. At test time, the model outputs are compared against the raw form of sentences, and model predictions on subwords are converted to labels on the raw sentences. Besides, based on our guidelines, the annotated span of hallucination words may also contain prepositions and other stop words.

Post-processing: We dropped all the translations that were labeled as incomprehensible (15 for TranS2S and 3 for MBART). To aggregate annotations from the three annotators, we assign the label to each token by majority voting, i.e. the label that two or more annotators agree on. We also aggregate the evaluation data from Maynez et al. (2020) in the same manner to produce our own test set for abstract text summarization.

.2 Training of NMT models

Tokenization For TranS2S, we first segment the Chinese corpus with a Chinese word segmentation tool (Luo et al., 2019), then we learn separate BPE vocabularies with 32k merge operations (Sennrich et al., 2016b) over the source (Zh) and the tokenized target (En) corpus respectively. For MBART, we directly apply the contained sentence-piece dictionary in the finetuned model to the raw data of Chinese and English corpus.

Model We use the implementation of Transformer from fairseq (Ott et al., 2019). Following the notations used in fairseq, we use a base transformer model for TranS2S and a large tranasformer model for MBART.

Models	Fleiss' Kappa	
	Token	Sent
MT		
TranS2S	0.58	0.72
MBART	0.54	0.62
XSum		
PtGen	0.81	-
TConvS2S	0.83	-
TranS2S	0.79	-
BERTS2S	0.79	-

Table 1: Fleiss's Kappa scores (↑): agreements on **token**-level hallucination labels or sentence-level (**sent**) ratings among different annotators. The token-level agreements for XSUM are computed on the released annotations by Maynez et al. (2020).

Training and Decoding For TranS2S, we apply the standard hyperparameters reported in the example of fairseq. We use the Adam optimizer (Kingma and Ba, 2014) using $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1e-8$. The learning rate is scheduled using `inverse_sqrt` with a maximum learning rate 0.0005 and 4000 warmup steps. We set the label smoothing as 0.1. We apply dropout of 0.1 and select the best model with validation BLEU scores. We run the model on 8 GPUs for $300,000$ updates with an effective batch size of around $64,000$ tokens. When finetuning MBART, we use learning rate of 3e-5, and use `polynomial_decay` for learning rate scheduling with warmup updates of 3,000. The effective batch size is 16,384. Dropout is set to be 0.3 and the attention dropout rate is 0.1. The label smoothing is set to be 0.2. We finetune MBart for 60,000 updates. We decode outputs with beam-search and beam size of 5.

.3 Experimental Details for Token-level Hallucination Prediction

Subword Tokenization Depending on the pretrained model (Roberta / XLM-Roberta) we finetune on, we apply corresponding subword segmentation to the synthetic data set (S, T, T') and calculate the edit-distance between the T and T' at the subword level. At evaluation time, the model predicts the hallucination labels for each subword in the sentence, thus we predict a word to be a hallucination word if any subword of it is predicted as a hallucinated one.

Synthetic data generation There are a couple of hyperparameters of noised functions in the BART implementation (Ott et al., 2019). The main noised functions include (1) random masking, (2) random replacement, (3) random insertion of masks. We found that random masking and random replacement are the two key factors affecting the generated sentences and we have provided their settings in the main paper. We apply a random insertion masks rate of 0.2 for all settings. In addition, the noise functions are

Input to $\mathcal{N}(\cdot)$	Precision	Recall	F1
MT			
raw	58.35	70.12	63.70
TranS2S distill	64.27	67.30	65.75
Summarization			
raw	57.02	67.23	61.70
Extractive distill	54.10	36.45	43.55
Abstractive distill	57.33	28.59	38.16

Table 2: Performance on the TranS2S benchmark from MT and summarization by using different data as the input to the noised function $\mathcal{N}(\cdot)$. "raw" refers to the original targets in the training data.

applied to words instead of spans in our setting.

Finetuning For MT, we finetune a large XLM-Roberta (Conneau et al., 2020a) released in fairseq (Ott et al., 2019). For summarization, we finetune a large Roberta (Ott et al., 2019) on the synthetic data where we truncate articles that exceed 512 tokens (allowed by the Roberta) to be 512. For both models, we use the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1e-6$ and weight decay of 0.1. We set the masking probability to be 0.35 for the $\mathcal{L}_{mlm}$ loss. The dropout and attention dropout rates are set to be 0.1. We adopt `polynomial_decay` for learning rate scheduling with learning rate of 2e-5.

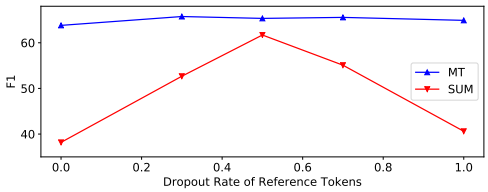


Figure 1: Performance on the TranS2S outputs from MT and summarization by varying the token dropout rate of in the reference at training time.

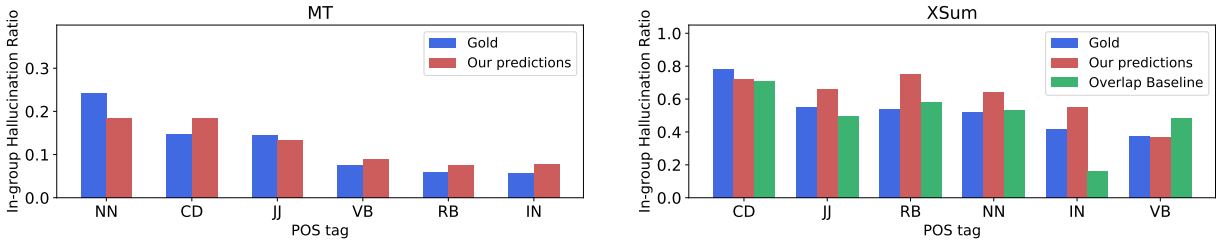


Figure 2: Analysis of part-of-speech tags and within-group percentage of hallucinations for MT (left) and summarization (right).

Methods	TranS2S	MBART
Alignment	(18.90, 66.82, 29.47)	(5.63, 42.09, 9.93)
Overlap-based	(7.02, 13.10, 9.14)	(1.98, 8.97, 3.24)
Synonym-based	–	–
Ours (w/o ref)	**(64.27, 67.30, 65.75)**	**(49.56, 36.32, 41.92)**
Ours (w/ ref)	(59.92, 74.27, 66.08)	(43.13, 53.63, 46.81)

Table 3: Triplets represent (Precision, Recall, F1 (x100)) of hallucination labels on the outputs of different systems from a MT task (§3.4.2). The first block are baseline methods and the second block are our results. We highlight the best results without using reference.

Source	信息组被称作页面数据。
Reference	the set of information is called page data.
Generation	the foreign[1] mix[1] is called the page data.
Source	金属线对应于第一电阻器。
Reference	the metal lines correspond to first resistors.
Generation	the wire corresponds with the first capital[1].
Source	驱动样本流过液流通路;
Reference	driving samples to flow through a flow channel;
Generation	driving samples pass the flow of people[1];

Table 4: Examples of partially hallucinated outputs from the teacher MT model used in self-training and the hallucinated labels predicted by our system. We only highlight words with hallucination labels with [1].

.4 Ablation Studies

Effects of including reference at training time Recall that we concatenate the source, reference and machine generation together as the input when learning hallucination predictions (Sec. 3.3.2). In Fig.1, we vary the dropout rate of tokens in the reference at training time and evaluate the models on the outputs from the TranS2S model for both tasks, where dropout rate of 1.0 indicates that we do not include the reference at all. **First**, different dropout rates do not signficinatly affect performance for MT, this is likely because we use the paraphrased target when creating the synthetic data instead of the reference sentences. Thus, the "hallucinated" sentences D' from BART do not resemble the reference T as closely as T', and the model will not learn spurious correlations between the T and D'. **Second**, for summarization we see that applying word dropout is crucial since we have used the reference more directly for generating synthetic data. On the other hand, if reference is removed at learning time (dropout = 1.0), the resulted model performs poorly, which shows that including reference at training time also

Methods	PtGen	TConvS2S	TranS2S	BERTS2S
Alignment	(60.66, 28.65, 38.92)	(66.14, 26.60, 37.94)	(56.24, 24.85, 34.47)	(50.68, 27.69, 35.81)
Overlap-based	(67.72, 49.54, 57.22)	(60.39, 49.24, 54.25)	(53.22, 54.37, 53.79)	(62.57, 49.26, 55.13)
Synonym-based	(50.52, 72.50, 59.55)	(57.06, 72.16, 63.73)	(50.29, 70.37, 58.66)	(41.80, 72.67, 53.07)
Ours (w/o ref)	(57.47, 71.35, 63.66)	(63.21, 68.93, 65.94)	(57.02, 67.23, 61.70)	(49.83, 62.50, 55.45)
Ours (w/o ref + syn)	**(50.33, 90.27, 64.72)**	**(56.86, 88.93, 69.37**	**(50.21, 87.78, 63.88)**	**(41.70, 87.52, 56.49)**
Ours (w/ ref)	(56.51, 73.48, 63.89)	(61.68, 71.63, 66.28)	(55.88, 70.19, 62.24)	(48.39, 66.11, 55.88)

Table 5: Triplets represent (Precision, Recall, F1 (x100)) of hallucination labels on the abstract summarization task (XSum dataset). The first block are baseline methods and the second block are our results. We highlight the best results without using reference.

has positive effects.

Effects of paraphrased data We investigate the effects of using paraphrased data in Tab. 2, where we apply the noise functions to different forms of targets when generating synthetic data. For MT, we create paraphrased targets via knowledge distillation (Kim and Rush, 2016a) where we use the output from TranS2S conditioned on the source sentence in the bi-text corpus as the paraphrased target. We can see that with distillation data for synthetic data generation, the model achieves better results compared to using the references. However, note that we need to choose a proper word dropout rate when using the reference-based synthetic data as discussed above. For abstractive summarization, we create paraphrased data out of an abstractive and an extractive summarization systems respectively. We finetune BART on the bi-text of XSum and create distillation data from this finetuned abstractive model. For the extractive system, we use the recent proposed MatchSum (Zhong et al., 2020) as the distillation model. We see a significant drop in the performance for both of the variants. This likely due to the fact that: (1) it has been shown that abstractive summarization systems are prone to hallucinate contents themselves (Maynez et al., 2020), thus we are not able to create reliable pseudo labels based on the generated summaries, and (2) the extractive system generates summaries out of the input article which diverge from the actual abstractive summaries we evaluate on, and the model cannot generalize well under such data shift.

.5 Supplymental Results and Analysis

.5.1 Full Results of Token-level Hallucination Predictions

We found the synonym and string-matching based methods are strong and effective baselines on the monolingual (summarization) token-level hallucination prediction task as an alternative to neural methods. However, previous work (Maynez et al., 2020; Wang et al., 2020a; Durmus et al., 2020) on hallucination assess did not study synonym-based non-neural baselines when measuring the faithfulness of the

Reference	the arrangement pattern of the projections 2 will now be explained with reference to figs. 5-7.
Annotation	next,[0] we[0] use[0] fig.[0] 5[0] -[0] 7[0] to[0] explain[0] the[0] disposition[0] pattern[0] with[0] pm-2.[1]
Prediction	next,[0] we[0] use[0] fig.[0] 5[0] -[0] 7[0] to[0] explain[0] the[0] disposition[0] pattern[0] with[1] pm-2.[1]
Reference	a swivel joint 557 is provided in a radially outer region, on an end surface of the drive plate 556.
Annotation	a[0] rotation[0] hinged[1] 557[0] is[0] provided[0] to[0] the[0] external[0] area[0] on[0] a[0] trail[1] that[0] has[0] a[0] preface[1] state.[1]
Prediction	a[0] rotation[0] hinged[0] 557[0] is[0] provided[1] to[0] the[0] external[0] area[0] on[0] a[0] trail[1] that[1] has[1] a[0] preface[1] state.[1]
Reference	if you have a fever of a hundred and two or higher.
Annotation	if[0] your[0] heat[0] reaches[0] 102.d[0] egree.[0] f.[0] or[0] above,[0]
Prediction	if[0] your[0] heat[0] reaches[0] 102.d[1] egree.[1] f.[1] or[0] above,[0]

Table 6: Examples of annotations and our hallucination detection model predictions, [0] and [1] respectively indicate faithful and hallucinated word.

summarization model outputs.

.5.2 Analysis on Part-of-speech tags and with-in Group Hallucination Percentage

We have shown that the macro Part-of-Speech tag distribution of hallucinated tokens in §3.5.3. In this section, we analyze the micro-percentage of hallucination labels within each POS tags. We show the gold annotations as well as our model predictions of hallucination words within each POS tags. For summarization, we also show the results from the string-matching baseline. From Fig. 2, we can see that for MT nouns are most likely hallucinated words while for summarization cardinal numbers (e.g. *one, two*) are most likely hallucinated words. And we can see that our model predictions align well with the gold annotations on the percentage of hallucinated words within each POS tags.

.5.3 Examples of Partially Hallucinated Outputs from Teacher MT Model

In Tab. 4, we randomly select some examples for which we present the source sentences from the patent monolingual Chinese dataset, the corresponding reference English sentences and the generations from a teacher model trained on the training data described in §3.4.2 where patent is a low-resource domain. We can see that in these examples, only parts of the model outputs are hallucinated and the rest of the outputs are good translations that are faithful to the source. Through our approach in §3.6, we can still make use of these good parts of translation during training.

.5.4 Examples of Hallucination Predictions on the MT test set

As shown Tab. 6, our model performs well in general but can be inaccurate in case of spelling errors of the translations. Besides, we also find some annotation errors while our model predicts correctly.

Appendix II: Examining and Combating Spurious Features under Distribution Shift

.6 Proofs of Theorem 1

Lemma 1. *If T is sufficient statistics, we have $p(Y, X|T) = p(Y|T) \cdot p(X|T)$.*

Lemma 2. *If T is sufficient statistics, we have $p(Y|T(X)) = p(Y|X)$.*

Proof. Find $T'(X)$, s.t. $S(x) = \langle T(X), T'(X)\rangle$ is an invertible mapping of X, thus $p(Y|X) = p(Y|S(X)) = p(Y|T(X), T'(X))$. We have,

$$p(Y, T(X), T'(X)|T(X)) = p(Y|T'(X), T(X))p(T'(X)|T(X)) \tag{1}$$

From Lemma 1, we have

$$p(Y, T(X), T'(X)|T(X)) = p(Y|T(X))p(T'(X)|T(X)) \tag{2}$$

By (1) and (2), we obtain $p(Y|T'(X), T(X)) = p(Y|T(X)) = p(Y|X)$. □

Theorem 1. *Suppose that there is only covariate shift in p_{train}, i.e. $\exists x \in \mathcal{X}_{train}$ s.t. $p_{train}(x) \neq p_{ideal}(x)$ but $p_{train}(Y|X = x) = p_{ideal}(Y|X = x),\ \forall x \in \mathcal{X}_{train}$. Let $T_{train}(X)$ be the MSS representation learned under p_{train}, then we have:*

$$H_{train}(T_{train}(x)|T_{ideal}(x)) = 0. \tag{4.2}$$

Proof. Since there is covariate shift between p_{ideal} and p_{train}, we have $p_{\text{train}}(Y|X) = p_{\text{ideal}}(Y|X), \forall x \in \mathcal{X}_{train}$. Since $T_{train}(X)$ is MSS of p_{train} and by Lemma 2, we have $p_{\text{train}}(Y|T_{train}(X)) = p_{\text{train}}(Y|X) =$

$p_{\text{ideal}}(Y|X) = p_{\text{ideal}}(Y|T_{ideal}(X)), \forall x \in \mathcal{X}_{train}$. Then $\forall x \in \mathcal{X}_{train}, y \in \mathcal{Y}$,

$$\begin{aligned} p_{\text{train}}(y|T_{ideal}(x)) &= \sum_{x':T_{ideal}(x')=T_{ideal}(x)} p_{\text{train}}(y|x')p_{\text{train}}(x'|T(x)) \\ &= \sum_{x':T_{ideal}(x')=T_{ideal}(x)} p_{\text{ideal}}(y|x')p_{\text{train}}(x'|T(x)) \\ &= \sum_{x':T_{ideal}(x')=T_{ideal}(x)} p_{\text{ideal}}(y|T(x))p_{\text{train}}(x'|T(x)) \\ &= p_{\text{ideal}}(y|T_{ideal}(x)) \end{aligned} \tag{3}$$

Then we have

$$\begin{aligned} H_{train}(Y|T_{train}(X)) &= \sum_{x,y} p_{\text{train}}(x,y)[-\log p_{\text{train}}(y|T_{train}(x))] \\ &= \sum_{x,y} p_{\text{train}}(x,y)[-\log p_{\text{ideal}}(y|T_{ideal}(x))] \\ &= \sum_{x,y} p_{\text{train}}(x,y)[-\log p_{\text{train}}(y|T_{ideal}(x))] \\ &= H_{train}(Y|T_{ideal}(X)) \end{aligned} \tag{4}$$

From equation 4 and the definition of sufficient statistics, we have

$$I_{train}(Y;T_{train}(X)) = I_{train}(Y;X) = I_{train}(Y;T_{ideal}(X)) \tag{5}$$

Thus, $T_{ideal}(X)$ is the sufficient statistics of X about Y under p_{train}. By definition, we have

$$H_{train}(T_{train}(X)|T_{ideal}(X)) = 0. \tag{6}$$

□

Corollary 1. *Suppose $\mathcal{X}_{train} = \mathcal{X}_{ideal}$ in Theorem 1, then $T_{train}(X)$ is also the MSS under p_{ideal}.*

Proof. Since $\mathcal{X}_{train} = \mathcal{X}_{ideal} = \mathcal{X}$, with the similar derivation of equation 3, we have $\forall x \in \mathcal{X}, y \in \mathcal{Y}$

$$p_{\text{ideal}}(y|T_{ideal}(x)) = p_{\text{ideal}}(y|T_{train}(x)) \tag{7}$$

Together with Theorem 1, we have $T_{train}(x)$ is also the MSS under p_{ideal}. □

.7 Connections between MLE and Learning Minimal Sufficient Statistics

.7.1 Information Bottleneck (IB) Method

The information bottleneck (IB) method (Tishby et al., 2000) is an information theoretic principle introduced to extract relevant information that an input $X \in \mathcal{X}$ contains about an output random variable

$Y \in \mathcal{Y}$. Defined on a joint distribution of X and Y, IB learns a mapping function $T(X)$ by optimizing the trade-off between the mutual information $I(X;T)$ and $I(Y;T)$ such that $T(X)$ is a compressed representation of X (quantified by $I(X;T)$) that is most informative about Y (quantified by $I(Y;T)$). Let T be parameterized by θ, the objective of IB optimizes the trade-off between $I(Y;T_\theta(X))$ and $I(X;T_\theta(X))$:

$$\min_{\theta} -I(Y, T_\theta(X)) + \beta I(X; T_\theta(X)) \tag{8}$$

where β is a positive Lagrange multiplier.

Schwartz-Ziv and Tishby (2017) casts finding of minimal sufficient statistics (MSS) $T(X)$ as a constrained optimization problem using data-processing inequality (Cover, 1999):

$$\begin{aligned} &\min_{T(X)} \quad I(T(X); X) \\ &s.t \;\; I(T(X); Y) = I(X; Y) \end{aligned} \tag{9}$$

This corresponds to the IB method (Eq. 8) which extends the notion of relevance between functions of samples and parameters in conventional MSS to any joint distribution of X and Y. The IB method provides a computational framework for finding approximate MSS in a soft manner by trading off the sufficiency for Y (I(Y; T(X))) and the minimality of the statistic ($I(X, T(X))$) with the Lagrange multiplier β (Schwartz-Ziv and Tishby, 2017; Shamir et al., 2010).

.7.2 Connections between MLE and IB

Given that the IB objective is approximately learning MSS in a soft manner, we next build the connections between the popularly adopted maximum likelihood estimation (MLE) in supervised learning and the IB objective. We show that under certain assumptions, MLE is approximating the IB objective defined on the joint distribution of $p_{\text{train}}(X, Y)$.

To facilitate the discussions, we decompose the model parameters into θ and ϕ that denote the parameters of the feature extractor $T_\theta(x)$ and the classifier respectively. MLE minimizes the expected negative log probability under $p_{\text{train}}(X, Y)$:

$$\min_{\theta,\phi} \mathbb{E}_{x,y\sim p_{\text{train}}(X,Y)}[-\log p_{\theta,\phi}(x, y)] \tag{10}$$

$$\iff \min_{\theta,\phi} \mathbb{E}_{x,y\sim p_{\text{train}}(X,Y)}[-\log p_\phi(y|T_\theta(x)) - \log p_\theta(x)] \tag{11}$$

Usually, we only model the conditional distribution $p_\phi(Y|X)$ and assume that $p_\theta(X) = p_{\text{train}}(X)$ which is independent from θ. With the assumption that $p_\theta(x) \propto p^\beta(T_\theta(x))), \beta > 0$, (11) can be rewritten as:

$$\min_{\theta,\phi} \mathbb{E}_{x,y\sim p_{\text{train}}(X,Y)}[-\log p_\phi(y|T_\theta(x))] + \beta \mathbb{E}_{x\sim p_{\text{train}}(X)}[-\log p(T_\theta(x))] \tag{12}$$

Assume that the neural network parameterized by ϕ is a universal function approximator, then we can replace $\min_{\theta,\phi}$ with $\min_\theta$ and (12) can be written as:

$$\min_\theta H(Y|T_\theta(X)) + \beta H(T_\theta(X)) \tag{13}$$

$$\text{by (1) } I(Y;T_\theta(X)) = H(Y) - H(Y|T_\theta(X))$$

$$\text{(2) } H(T_\theta(X)) = I(X;T_\theta(X)) + H(T_\theta(X)|X) = I(X;T_\theta(X))$$

$$\iff \min_\theta -I(Y;T_\theta(X)) + \beta I(X;T_\theta(X)) \tag{14}$$

We can see that under the assumption of $p_\theta(x) \propto p^\beta(T_\theta(x)))$, the MLE objective can be converted into the same form as the IB objective. In practice, we usually do not model $p_{\text{train}}(X)$ and only optimize the first term $I(Y;T_\theta(X))$ in (14). However, previous work (Schwartz-Ziv and Tishby, 2017; Geiger, 2020) has shown that deep neural networks (DNNs) are implicitly minimizing $I(X;T_\theta(X))$ with a wide range of activation functions and architectures, which are manifested as a second compression phase during learning with SGD. Thus, we can presumably consider MLE as approximating the IB objective, which is equivalent to learning the MSS on the train distribution $p_{\text{train}}(X,Y)$.

.8 Details of the Online Greedy Algorithm for Group DRO

Algorithm 3: Online greedy algorithm for group DRO (Oren et al., 2019)

Input : α; m: total number of groups

Initialize historical average group losses $\hat{L}^{(0)}$; historical estimate of group probabilities $\hat{p}^{train(0)}$; learning rate η

for $t = 1, \cdots, T$ **do**

Sample a mini-batch batch $B = (\mathbf{x}, \mathbf{y}, \mathbf{g})$ uniformly from p_{train}

▷ Update the historical vectors of $\hat{L}^{(t)}$ and $\hat{p}^{train(t)}$ for each group $g \in \{1, \cdots, m\}$

$\hat{L}^{(t)}(g) \leftarrow \text{EMA}(\{\ell(\mathbf{x}_i, \mathbf{y}_i; \theta^{(t-1)}) : \mathbf{g}_i = g\},\ \hat{L}^{(t-1)}(g))$

$\hat{p}^{train(t)} \leftarrow \text{EMA}(\#\text{samples of each group in B},\ \hat{p}^{train(t-1)})$

▷ Update the worst-case distribution $q^{(t)}$

Sort $\hat{p}^{train(t)}$ in the order of decreasing $\hat{L}^{(t)}$ and denote the sorted group indexes $\boldsymbol{\pi}$

$q^{(t)}(g_{\boldsymbol{\pi}_i}) = \min\{\frac{\hat{p}^{train(t)}(g_{\boldsymbol{\pi}_i})}{\alpha}, 1 - \sum_{j=1}^{i-1} \frac{\hat{p}^{train(t)}(g_{\boldsymbol{\pi}_j})}{\alpha}\}$

▷ Update model parameters θ

$\theta^{(t)} = \theta^{(t-1)} - \frac{\eta}{|B|} \sum_{i=1}^{|B|} \frac{q^{(t)}(\mathbf{g}_i)}{\hat{p}^{train(t)}(\mathbf{g}_i)} \nabla \ell(\mathbf{x}_i, \mathbf{y}_i; \theta^{(t-1)})$

end

EMA refers to exponential weighted moving average such that $\text{EMA}(v_1, v_2) = \gamma v_1 + (1-\gamma)v_2$, where $\gamma \in (0, 1)$.

.9 Synthetic Experiments: on Investigation Spurious Features under Covariate Shift

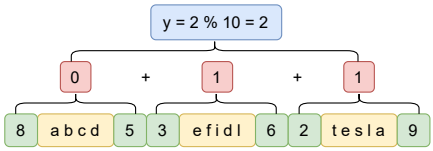


Figure 3: An illustrative example of the synthetic task.

Synthetic Experiments We design synthetic experiments where data is generated based on the ground-truth rules and different biases are injected. We show that even in the presence of necessary information to learn the rules, the ERM model (specifically, we examine MLE) can still learn spurious features or miss robust features under covariate shift. The synthetic task aims to predict an integer $y \in \{0, \cdots, 9\}$ conditioned on a sequence x as shown in Fig. 3. Concretely, x is composed of m chunks, where each chunk c_i has $|c_i|$ characters that are randomly sampled from an alphabet $\mathcal{V}$. We prepend an integer c_i^1 and append an integer c_i^2 to each chunk c_i, and both c_i^1 and c_i^2 are uniformly sampled from $[1, 10]$. The target integer y is predicted following the rules: each triple of (c_i^1, c_i, c_i^2) produces an indicator value d_i; $d_i = c_i^2 - c_i^1$ if $c_i^2 > c_i^1$, otherwise $d_i = 0$; then $y = (\sum_{i=1}^{m} d_i) \mod 10$. We set $3 \leq m \leq 6, 3 \leq |c_i| \leq 5$ and $|\mathcal{V}| = 26$,. We use a one-layer bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to model the input sequence and use the final hidden states of the LSTM to predict the target value. We create training data following the the above description and design two settings that introduce covariate shift to examine if the model can learn the rules with ERM.

(a) Setting 1 — ERM-trained models can miss robust features under covariate shift: We create the training data by imposing $c_m^2 > c_m^1$ on the last chunk c_m of all the training samples. When we create the training data, the rules applied to each chunk are the same as described above, which means that the model does not need to learn additional rules for the last chunk. We are interested in examining whether the model trained with ERM will apply the rules learned from other chunks to the last one or it will miss the robust features of the last chunk. At test time, we evaluate on two groups of test sets: $\mathcal{D}_{out}$ where $c_m^2 \leq c_m^1$, different from the training data, and $\mathcal{D}_{in}$ where $c_m^2 > c_m^1$, consistent with the training data. From Tab. 7, we see that the test accuracy on $\mathcal{D}_{out}$ is much lower that that on $\mathcal{D}_{in}$. This demonstrates that the model only learns robust features from chunks c_1^{m-1} but misses the robust features of the last chunk c_m. We conjecture that the model trained with ERM learns in a lazy way where it tries to minimize the entropy of learned features by memorizing patterns and taking shortcuts as discussed further in Appendix .7.2.

(b) Setting 2 — ERM-trained models can learn spurious features under covariate shift: In the second setting, we inject spurious patterns into the training data that co-occur with the rules we

	$\mathcal{D}_{in}$	$\mathcal{D}_{out}$
Setting 1	99.93 $\pm$ 0.02	14.68 $\pm$ 2.60
Setting 2	100.00 $\pm$ 0.00	10.26 $\pm$ 0.25

Table 7: Test accuracy of the synthetic task.

aim to learn. As both robust rules and spurious patterns co-exist in the training data, we would like to see whether the model picks up the spurious ones or the robust ones. Specifically, each training input sequence has a chunk c_j that includes a special segment of characters, e.g. *a b*. The remainder of $d_j = c_j^2 - c_j^1$ and the sum of all indicators $\sum_{i=1}^{m} d_i$ mod by 10 are the same such that the target label y is always the same as the indicator d_j. Similarly, we test on two cases: i) $\mathcal{D}_{in}$ where every sequence includes a special chunk as in the training set; ii) $\mathcal{D}_{out}$ where characters in each chunk are uniformly sampled. We can see from Tab. 7 that the model learns to use the spurious patterns to predict the target label instead of the general rules.

.10 Experimental Details

.10.1 Models and Training Details

Model Specific Settings In our method, we adopt two criterions in GC-DRO to determine when to update $q(x, y|g)$ for each groups: (1) update when the robust validation accuracy drops (2) update at every epoch. With (2), $q(x, y|g)$ is updated more frequently. For MNLI and Celeb-A, we use the second criterion. For FDCL18, we use the first criterion, because this is a relatively smaller dataset and updating $q(x, y|g)$ less frequently makes training more stable. Every time $q(x, y|g)$ is updates, we clear the historical losses in EMA that is used for updating $q(g)$. We use exponentially weighted moving average (EMA) to compute the historical losses for both $q(g)$ and $q(x, y|g)$, for which we denote EMA_{G} and and EMA_{CG} respectively. As shown above, we use γ to denote the coefficient for current value in EMA, thus $1 - \gamma$ is used to the historical value. We found that the value of γ is an important hyperparameter in some cases to achieve better performance, since the final q distribution is computed through sorting the losses accumulated via EMA. Basically, a higher γ pays more attention to the current value. We search over $\{0.1, 0.5\}$ for both γ used in EMA_{G} and EMA_{CG} respectively. Through the robust accuracy on the validation set, we set both γ's to be 0.5 for the NLP tasks except that for the imperfect partition of toxicity detection we set γ used in EMA_{G} to be 0.1. For the image task, we set both γ's to be 0.1. For the γ used in accumulating the historical fractions of groups, we always use a small value 0.01.

Training Details For the NLP tasks, we finetune a base Roberta model (Liu et al., 2019; Ott et al., 2019) and we segment the input text into the sub-word tokens using the tokenization described in (Liu et al., 2019). During training, we sample minibatches that contain at most 4400 tokens. We train MNLI using

Adam (Kingma and Ba, 2014) with an intitial learnig rate of $1e-5$ for 35 epochs and FDCL18 for 45 epochs, and we linearly decay the learning rate at every step until the end of training. For the image task, we fine-tune a ResNet-18 (He et al., 2016) for 50 epochs with batch size of 256. We use SGD with learning rate of $1e-4$. At the end of every epoch, we evaluate the robust accuracy on the validation set. We train on one Volta-16G GPU and it takes around 2 - 5 hours to finish one experiments for different datasets.

.10.2 Implementation of the Group DRO Loss

We referred to the implementation of greedy group DRO in Sagawa et al. (2020a), where they use the exact formulation in Eq. 4.5 to compute the expected loss, which leads to inferior performance compared to the exponentiated-gradient based optimization as reported in Sagawa et al. (2020a). The implementation computed the final loss by first computing the average loss over instances for each group (MC for the inner expectation), then compute the full expected value over the averaged group loss, as shown below:

$$\ell(\mathbf{x}, \mathbf{y}, \mathbf{g}; \theta) = \sum_g q(g)\bar{\ell}(g) = \sum_g q(g)\frac{1}{C_g}\sum_{\{i, \forall \mathbf{g}_i = g\}} \ell(\mathbf{x}_i, \mathbf{y}_i; \theta), \tag{15}$$

where $(\mathbf{x}, \mathbf{y}, \mathbf{g})$ is a mini-batch and C_g is the number of samples that belong to group g in the mini-batch. We can see that instances that belong to different groups are weighted correspondingly by the number of group size in a mini-batch. This causes that instances in large group get unfairly lower weights, especially when its probability in the q distribution is low. We fix this by directly computing the expected loss over the joint distribution of $q(x, y, g)$, i.e.

$$\mathbb{E}_{(x_i, y_i, g_i)\sim q(x,y,g)}\ell(x_i, y_i, g_i; \theta) = \mathbb{E}_{(x_i, y_i, g_i)\sim p_{\text{train}}(x,y,g)}\frac{q(x_i, y_i, g_i)}{p_{\text{train}}(x_i, y_i, g_i)}\ell(x_i, y_i, g_i).$$

Specifically, we do this by summing over all the importance weighted instance losses using corresponding group weights and taking average. This allows us to obtain unbiased gradient estimates of θ.

$$\frac{1}{N}\sum_i \frac{q(\mathbf{g}_i)}{p_{\text{train}}(g_i)}\ell(\mathbf{x}_i, \mathbf{y}_i; \theta) \tag{16}$$

Appendix III: Distributionally Multilingual MT

.11 Best response

In this section, we describe the bisection procedure we use to solve the best response and update q shown in equation 5.7. This derivation generically exists in the literature (e.g. Appendix A.1.2 in Levy et al. (2020)) but we specialize it to the χ^2-ball centered at p^{train} and include it here for completeness.

For $v \in \mathbb{R}^m$, the optimization problem we wish to solve is

$$\begin{aligned} \underset{q \in \Delta^m}{\text{maximize}} \quad & q^\top v \\ \text{subject to} \quad & \chi^2(q, p^{\mathsf{train}}) \leq \rho, \end{aligned} \tag{17}$$

Let us consider the Lagrangian of this problem

$$\Lambda(q; \nu, \eta) \coloneqq q^\top v - \eta(\mathbf{1}^\top q - 1) - \lambda(\chi^2(q, p^{\mathsf{train}}) - \rho)$$

Maximizing over q yields that the solution as a function of η is $q^\star(\eta)_i \propto p_i^{\mathsf{train}}(v_i - \eta)_+$, where $(u)_+ \coloneqq \max\{0, u\}$. Since the objective is linear, it holds that the maximum is attained on the extreme points of the constraint sets. Consequently, we need to find $\eta^\star$ such that $\chi^2(q^\star(\eta^\star), p^{\mathsf{train}}) = \rho$. This is a simple root finding procedure that we solve to accuracy ϵ in $(1/\epsilon)$ steps with a bisection. Given that each evaluation of $q^\star(\eta)$ requires $O(m)$ operations, the runtime of the algorithm is $O(m \log(1/\epsilon))$.

.12 Primal-dual methods

Primal-dual algorithms Nemirovski (2004); Nemirovski et al. (2009) are the methods of choice to efficiently solve min-max problems.

.12.1 The primal-dual algorithm

Let us assume that $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}^p$ are closed, bounded convex sets and let $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be a function such that $F(x, \cdot)$ is concave for all $x \in \mathcal{X}$ and $F(\cdot, y)$ is convex for all $y \in \mathcal{Y}$. We wish to solve

the following min-max problem

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} F(x,y). \tag{18}$$

This is a well-studied problem of optimization and there the literature provides optimal algorithms for many choices of F, $\mathcal{X}$ and $\mathcal{Y}$. A standard approach are the so-called *primal-dual methods*, where one keeps a pair of iterates (x_t, y_t) performs a gradient descent (resp. ascent) step on x_t (resp. y_t). In all generality, these updates are often mirror descent updates to properly exploit the geometric structures of $\mathcal{X}, \mathcal{Y}$ and F. Let $h^{\mathsf{x}} : \mathbb{R}^d \to \mathbb{R}$ (resp. $h^{\mathsf{y}} : \mathbb{R}^p \to \mathbb{R}$) be a 1-strongly-convex function w.r.t. a given norm $\|\cdot\|_{\mathsf{x}}$ (resp. $\|\cdot\|_{\mathsf{y}}$). We denote $D_{h^{\mathsf{x}}}$ and $D_{h^{\mathsf{y}}}$ their associated Bregman divergences. The primal-dual algorithms initialize $x_0 \in \mathcal{X}, y_0 \in \mathcal{Y}$ and for a stepsize $\eta > 0$, iterates

$$\begin{aligned} x_{t+1} &= \arg\min_{x\in\mathcal{X}}\left\{ g_{\mathsf{x},t}^\top x + \frac{1}{\eta} D_{h^{\mathsf{x}}}(x, x_t)\right\} \\ y_{t+1} &= \arg\max_{y\in\mathcal{Y}}\left\{ g_{\mathsf{y},t}^\top y - \frac{1}{\eta} D_{h^{\mathsf{y}}}(y, y_t),\right\}, \end{aligned}$$

where $g_{\mathsf{x},t} \in \partial_x F(x_t, y_t)$ and $g_{\mathsf{y},t} \in \partial_y F(x_t, y_t)$. After T steps, return $(\bar{x}_T, \bar{y}_T)$ with $\bar{z}_T \coloneqq 1/T \sum_{t\le T} z_t$. Assuming F is appropriately Lipschitz and that $\mathcal{X}$ and $\mathcal{Y}$ are bounded, one finds an ϵ-approximate saddle point in $O(\epsilon^{-2})$ steps. Importantly, these guarantee still holds even when only having *stochastic unbiased estimates* of g_{x} and g_{y} Nemirovski et al. (2009) which is essential in large-scale settings.

.12.2 Primal-dual algorithms for Group DRO

The problem equation 5.3 can be cast as an instance of equation 18 by setting $\mathcal{Y} = \mathcal{U}$ and $F(\theta, q) \coloneqq \sum_{i\le N} q_i \mathcal{L}(\theta; D_i)$. Note that the problem is unconstrained in θ but it is generally not an issue in practice. The gradient in θ and q are

$$\begin{aligned} \nabla_\theta F(\theta, q) &= \sum_{i\le N} q_i \nabla_\theta \mathcal{L}(\theta; D_i) \\ [\nabla_q F(\theta, q)]_i &= \mathcal{L}(\theta; D_i). \end{aligned}$$

Obtaining stochastic gradients

To run primal-dual, we require stochastic gradient estimates $\tilde{g}_\theta$ and $\tilde{g}_q$. First of all, note that an unbiased estimate of $\nabla_\theta \mathcal{L}(\theta; D_i)$ is just $\nabla_\theta \ell(\boldsymbol{x}, \boldsymbol{y}; \theta)$ where $(\boldsymbol{x}, \boldsymbol{y})$ is sampled at random from D_i. To obtain stochastic gradient estimates, we have two choices: sampling from q or sampling from an arbitrary $p_0 \in \Delta^N$ and importance-weighting.

Sampling from q Let B be the mini-batch size and $(I_1, \ldots, I_B)$ be B indices sampled from q and $(\boldsymbol{x}_{I_j}, \boldsymbol{y}_{I_j})$ be randomly sampled examples from D_{I_j}. In this case,

$$\tilde{g}_\theta = \frac{1}{B} \sum_{j \leq B} \nabla_\theta \ell(\boldsymbol{x}_{I_j}, \boldsymbol{y}_{I_j}; \theta)$$
$$[\tilde{g}_q]_i = \frac{1}{B} \sum_{j \leq B} \frac{1}{q_{I_j}} \ell(\boldsymbol{x}_{I_j}, \boldsymbol{y}_{I_j}; \theta),$$

are clearly unbiased stochastic gradient estimates for $(\nabla_\theta F(\theta, q), \nabla_q F(\theta, q))$.

Sampling from an arbitrary $p_0 \in \Delta^N$ Assume p_0 is in Δ^n and that $p_{0,i} > 0$ for $i \in [N]$. As previously, assume we sample a mini-batch of indices $(J_1, \ldots, J_B)$ from p_0 and that $(\boldsymbol{x}_{J_j}, \boldsymbol{x}_{J_j})$ is a randomly sampled example from D_{J_j}. We can obtain stochastic gradient estimates by importance-weighting (IW)

$$\tilde{g}_\theta^{\mathsf{IW}} = \frac{1}{B} \sum_{j \leq B} \frac{q_{J_j}}{p_{0,J_j}} \nabla_\theta \ell(\boldsymbol{x}_{J_j}, \boldsymbol{y}_{J_j}; \theta)$$
$$[\tilde{g}_q^{\mathsf{IW}}]_i = \frac{1}{B} \sum_{j \leq B} \frac{1}{p_{0,I_j}} \ell(\boldsymbol{x}_{I_j}, \boldsymbol{y}_{I_j}; \theta),$$

In terms of implementation, it is often impractical to change the distribution of batches at each iteration of the optimizer.

Choice of Bregman divergence

θ-update. The update in θ is unconstrained and it is standard to optimize the objective with Stochastic Gradient Descent so as a result, we pick $h^\theta(\theta) = \frac{1}{2}\|\theta\|_2^2$. This results in the standard SGD update

$$\theta_{t+1} = \theta_t - \eta \tilde{g}_\theta,$$

where $\tilde{g}_\theta$ is an unbiased stochastic gradient estimate that we compute following the previous section.

q-update for $\mathcal{U} = \Delta^N$. In the case where $\mathcal{U}$ is the full simplex, it is standard to choose $h^{\mathsf{q}}(q) = \sum_{i \leq N} q_i \log q_i$ (the negative Shannon entropy). This choice yields the familiar Exponentiated Gradient update

$$q_{t+1,i} \propto q_{t,i} \exp(\eta [\tilde{g}_q]_i)$$

q-update for $\mathcal{U} = \{q \in \Delta^N : \chi^2(q, p^{\mathsf{train}}) \leq \rho\}$. As the χ^2-uncertainty set is essentially the intersection between a (weighted)-norm-2 ball and the simplex, it is a natural choice to pick $h^{\mathsf{q}}(q) = \frac{1}{2}\|q\|_2^2$. This leads to the following update

$$q_{t+1} = \operatorname*{arg\,min}_{q \in \Delta^N : \chi^2 q, p^{\mathsf{train}} \leq \rho} \|(q_t + \eta \tilde{g}_q) - q\|_2^2,$$

or in other words, the projection of the gradient ascent step $(q_t + \eta \tilde{g}_q)$ onto the χ^2-ball. We explain in Appendix .12.3 how to efficiently compute this projection for arbitrary p^{train}.

q-update for $\mathcal{U} = \mathcal{U}_\alpha$. We follow the implementation of Oren et al. (2019) which runs a hybrid between primal-dual methods and best response and thus we do not need to explicit include the projection onto the CVaR uncertainty set. We discuss the option here for completeness. For the CVaR uncertainty set, it is standard to also use the negative Shannon entropy $h^{\mathsf{q}} = \sum_i q_i \log q_i$. We refer to Appendix F.6.2 of Levy et al. (2020) and their provided code for more details on this projection.

.12.3 Projecting on the χ^2-ball

To run the primal-dual algorithm, the gradient update on q requires projecting on the χ^2-ball centered at p^{train}. For $v \in \mathbb{R}^m$, we wish to solve the following

$$\begin{array}{ll} \underset{q \in \Delta^m}{\text{minimize}} & \|q - v\|_2^2 \\ \text{subject to} & \chi^2(q, p^{\mathsf{train}}) \leq \rho, \end{array} \tag{19}$$

where Δ^m is the m-dimensional simplex and $\chi^2(q, p^{\mathsf{train}}) \coloneqq \frac{1}{2}\sum_{i \leq m} p_i(q_i/p_i - 1)^2$, the f-divergence corresponding to $t \mapsto \frac{1}{2}(t-1)^2$ (Csiszár, 1967).

While this projection is standard in the literature, it is often derived in the case of $p^{\mathsf{train}} = \mathbf{1}/m$ (see e.g. Namkoong and Duchi (2016)). We show here how to efficiently do the projection for arbitrary $p^{\mathsf{train}} \in \Delta^m$.

First, for $\lambda \geq 0, \eta \in \mathbb{R}$, the Lagrangian of equation 19 is

$$\Lambda(q, \lambda, \eta) \coloneqq \frac{1}{2}\|q - v\|_2^2 + \lambda(\chi^2(q, p^{\mathsf{train}}) - \rho) + \eta(q^\top \mathbf{1} - 1).$$

Taking the partial dual $g(\lambda, \eta) \coloneqq \inf_{q \succeq 0} \Lambda(q, \lambda, \eta)$ yields

$$g(\lambda, \eta) = -\inf_{\lambda \geq 0, \eta \in \mathbb{R}} \frac{1}{2}\sum_{i \leq m} \frac{p_i}{p_i + \lambda}(v_i - \eta)_+ + \frac{\lambda}{2}(1 + 2\rho) + \eta$$

In contrast to the uniform case (i.e. $p^{\mathsf{train}} = \mathbf{1}/m$), one cannot derive the optimal dual variable λ^* in closed-form and we have to solve for both dual variables. Finding an ϵ-accurate solution takes order $m \log(1/\epsilon)$ time using cutting plane-type methods when the dimension is $O(1)$ Bubeck (2015). In the large-scale applications we consider, this is negligible in comparison to computing the gradient of the loss with respect to the network parameters and thus the primal-dual algorithm incurs (almost) no additional computational overhead.

In practice, we implement this with two nested bisections; while this adds an extraneous $\log(1/\epsilon)$ factor, this is significantly more convenient to implement.

Appendix III: Distributionally Robust Multilingual Machine Translation

.13 Data Statistics

related	#sents	diverse	#sents
bel	4,509	bos	5,664
aze	5,946	mar	9,840
glg	10,017	hin	18,798
slk	61,470	mkd	25,335
cse	103,093	ell	134,327
tur	182,470	fra	192,304
por	184,755	bul	174,444
rus	208,458	kor	205,640

Table 8: Number of training sentences in the TED related and diverse sets respectively.

We present the number of training sentences in the TED datasets in Tab. 8. The number of training sentences in the WMT dataset is 2.5M (`deu`), 1.8M (`fra`), 512,608 (`tam`) and 195,762 (`tur`).

.14 Preprocessing and Training Details

We describe the preprocessing and training details in this section.

Preprocessing We download WMT datasets from the WMT official websites[1]. The TED data is downloaded from the link[2] provided by Wang et al. (2020d). We learn sentencepiece (Kudo and Richardson,

[1] e.g. http://www.statmt.org/wmt18/

[2] https://github.com/cindyxinyiwang/multiDDS.

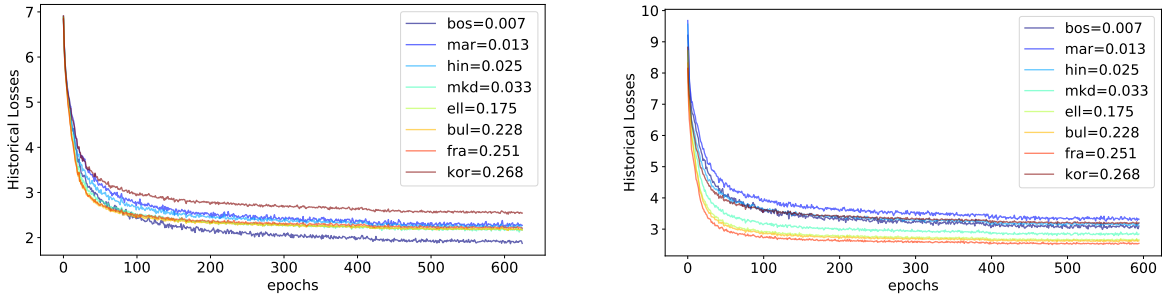


Figure 4: The historical (EMA) training losses on the TED-diverse dataset (left: any→en, right: en→any).

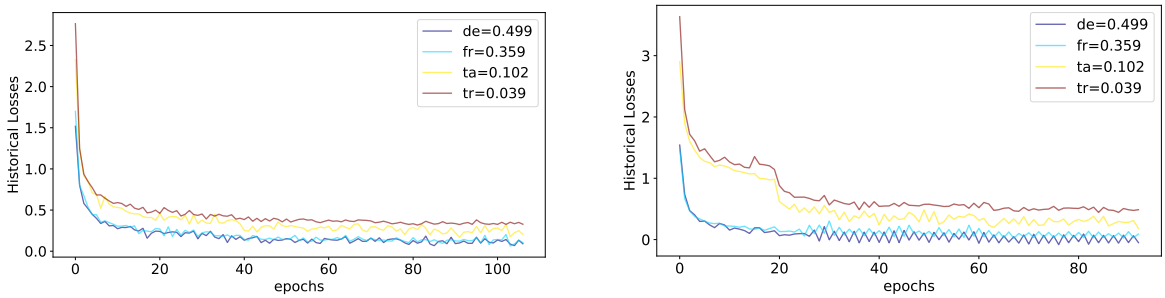


Figure 5: The historical (EMA) training losses on the WMT dataset (left: any→en, right: en→any).

2018) for English and other languages respectively. Specifically, we combine all the English sentences from individual parallel datasets. Following Arivazhagan et al. (2019), we also resample the sentences of other languages based on the temperature-based distribution ($\tau = 5$) to learn the vocabulary. For TED, the vocabulary sizes for English and other languages are set to be 10K and 30K respectively. For WMT, the vocabulary sizes for English and other languages are set to be 24K and 34K respectively. We also remove sentences in the training data that are longer than 250 tokens after bpe. For validation set, to ensure the validation loss are fairly comparable across different languages, we cap the number of the sentences in the validation set to be the same for each language (800 for TED and 1,500 for WMT).

Models	*small*	*base*
d_{model}	512	512
d_{hidden}	1024	2048
n_{layers}	6	6
n_{heads}	4	8
p_{dropout}	0.3	0.3

Table 9: Basic hyper-parameters of Transformer.

Training details The hyperparameters of Transformer models are described in Tab. 9. For all experiments, we adopt the Adam optimizer (Kingma and Ba, 2014) using $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1e-8$.

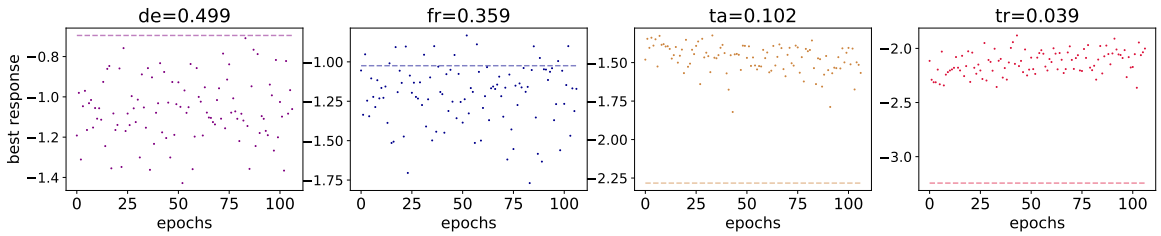


Figure 6: Best response q (in log-scale) across epochs on the WMT dataset for the any→en direction, the dashed line is the true data probability (in log-scale).

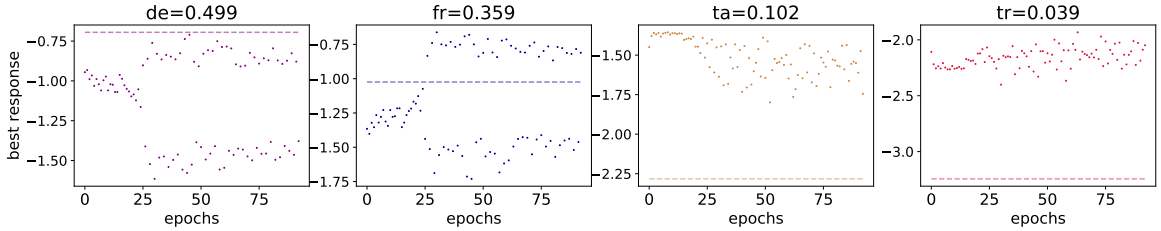


Figure 7: Best response q (in log-scale) across epochs on the TED diverse dataset for the en→any direction, the dashed line is the true data probability (in log-scale).

We set the label smoothing as 0.1[3]. For ERM models, we use the default learning rate scheduling. The learning rate is scheduled using `inverse_sqrt` with a maximum learning rate of $5e-4$ and $2e-4$ (following (Wang et al., 2020d)) for WMT and TED respectively. The warmup steps is set to be 4000 for both datasets. For χ-IBR , because the training distribution is dynamically changing across epochs, we use step learning rate decay to retain a higher learning rate during training. Specifically, we use the same maximum learning rate and warm up steps as ERM, but decay the learning rate by half every 100K training steps for both datasets. Recall that we use exponential moving average to compute the historical loss values, and we set the hyperparameter λ to be 0.1 throughout. We train on 4 and 8 V100 GPUs for TED and WMT experiments respectively. The training time for one experiment takes around 1 - 2 days.

[3]Note that we also use the label smoothing loss as the baseline loss from trained ERM models

Appendix IV: Towards a Unified View of Parameter-Efficient Transfer Learning

.15 Experiments

.15.1 Setups

Table 10: Dataset Statistics of the four tasks.

Dataset	#train	#dev	#test
XSum	204,045	113,332	113,334
WMT16 en-ro	610,320	1,999	1,999
MNLI	392,702	9815	9832
SST-2	67,349	872	1,821

We implement all the parameter-efficient tuning methods using the huggingface transformers library (Wolf et al., 2020). We use $\text{BART}_{\text{LARGE}}$(Lewis et al., 2020a) and $\texttt{mBART}_{\text{LARGE}}$ (Liu et al., 2020b) (mBART-cc25) for the summarization and machine translation tasks respectively, and we use $\text{RoBERTa}_{\text{BASE}}$ (Liu et al., 2019) for MNLI and SST2. $\text{BART}_{\text{LARGE}}$ and $\texttt{mBART}_{\text{LARGE}}$ have the same encoder-decoder architectures. $\texttt{mBART}_{\text{LARGE}}$ is pre-trained on 25 languages. We use their public checkpoints from the transformers library in experiments. For MT and classifications tasks, the max token lengths of training data are set to be 150 and 512 respectively. For XSum, we set the max length of source articles to be 512 and the max length of the target summary to be 128. The detailed dataset statistics is present in Table 10. In our summarization experiments, we only use 1600 examples for validation to save time.

While we vary the bottleneck dimension within $\{1, 30, 512, 1024\}$ as mentioned in §6.4.1, we test bottleneck dimension 1024 only when the modified representation is FFN, because the training of prefix tuning does not fit into 48GB GPU memory when $l = 1024$. While other methods do not have memory issues, we keep the bottleneck dimension of attention modification at most 512 to have a relatively fair

Table 11: Training hyperparameters of parameter-efficient tuning methods on the four tasks. lr and ls represents learning rate and label smoothing respectively.

Tasks	lr	batch size	ls	max grad norm	weight decay	train steps
XSum	5e-5	64 sents	0.1	0.1	0.01	100K
enro MT	5e-5	16384 tokens	0.1	1.0	0.01	50K
MNLI/SST2	1e-4	32 sents	0	1.0	0.1	10 epochs

comparison with prefix tuning. For LoRA we always tune its scaling hyperparameters s on the dev set.

.15.2 Training and Evaluation

We present some training hyperparameters of parameter-efficient tuning methods in Table 11. For all the tasks, we train with the Adam optimizer (Kingma and Ba, 2014), and use a polynomial learning rate scheduler that linearly decays the learning rate throughout training. We set the warm up steps of learning rate to be 0 for both MT and summarization tasks, and for the classification tasks, learning rate is linearly warmed up from 0 for the first 6% of the total training steps before decay. For full fine-tuning we set these training hyperparameters following Lewis et al. (2020a) (XSum), Liu et al. (2020b) (en-ro), and (Liu et al., 2019) (MNLI and SST2). We also did hyperparameter search in the full fine-tuning case to try to reproduce their results. We set dropout rate to be 0.1 for all the tasks. We use ROUGE-2 and perplexity as the validation metrics for summarization and MT respectively.

For MT and text summarization, we use beam search for decoding and set the number of beams to be 6 and 5 following previous work (Li and Liang, 2021b; Liu et al., 2020b). The min and max generation lengths for summarization and MT are set to be (10, 60) and (1, 200) respectively.

.15.3 Other Experimental Details

Prefix Tuning: Following Li and Liang (2021b), we reparameterize the prefix vectors by a MLP network which is composed of a small embedding matrix and a large feedforward neural network. This is conducive for learning due to the shared parameters across all layers.

LoRA: LoRA and adapter employ different parameter initialization methods: LoRA uses a random Kaiming uniform (He et al., 2015) initialization for $\boldsymbol{W}_{\text{down}}$ and zero for $\boldsymbol{W}_{\text{up}}$ (LoRA init), while adapters use the same initialization as BERT (Devlin et al., 2019a). We found it beneficial to use the same initialization method as LoRA in scaled PA.

.16 Computation of Tunable Parameters

Figure 8: Number of attention or FFN sub-layers in each layer of the pre-trained models.

	BART/mBART$_{\text{LARGE}}$	RoBERTa$_{\text{BASE}}$
N_{attn}	3	1
N_{ffn}	2	1

Figure 9: Number of parameters used at each sub-layer for different methods.

	$N_{\text{W}}^{\text{attn}}$	$N_{\text{W}}^{\text{ffn}}$
Prefix Tuning	$2ld$	–
Adapter variants	$2rd$	$2rd$
LoRA	$2 \times 2rd = 4rd$	$2 \times (rd + 4dr) = 10rd$

We compute the number of tunable parameters based on where the tunable module is inserted into and how it is parameterized. The pretrained-models for summarization or MT have an encoder-decoder structure and each has L layers, whereas RoBERTa$_{\text{BASE}}$ for classification tasks only has L encoder layers. To simplify the computation of tunable parameters, we compute the sum of parameter used in one encoder layer and one decoder layer as the parameter overhead of one single layer of the pre-trained encoder-decoder model. Each layer has N_{attn} sub-layers and N_{ffn} sub-layers. For the encoder-decoder models, $N_{\text{attn}} = 3$: the encoder self-attention, the decoder self-attention and the decoder cross-attention. For the classification tasks, RoBERTa$_{\text{BASE}}$ only has the encoder self-attention, thus $N_{\text{attn}} = 1$. We present the number of attention and ffn sub-layers for different pre-trained models in Table 9. For modifications applied at the attention sub-layers, the number of tunable parameters is computed by $|\Theta|_{\text{attn}} = N_{\text{W}}^{\text{attn}} \times N_{\text{attn}} \times L$, where $N_{\text{W}}^{\text{attn}}$ denotes the number of parameters ($\boldsymbol{W}_{\text{down}}$ or $\boldsymbol{W}_{\text{up}}$) used for one attention sub-layer. Similarly, the number of tunable parameters for the FFN sub-layers is computed by $|\Theta|_{\text{ffn}} = N_{\text{W}}^{\text{ffn}} \times N_{\text{ffn}} \times L$. In Table 9, we show the number of parameters for one sub-layer. As we have explained in §6.4.4, LoRA approximates the update of each weight matrix with a pair of $\boldsymbol{W}_{\text{down}}$ and $\boldsymbol{W}_{\text{up}}$, thus LoRA typically uses more parameters with the same r as other methods. Finally, the total number of tunable parameters for prefix tuning, adapter variants and LoRA is $|\Theta| = |\Theta|_{\text{attn}} + |\Theta|_{\text{ffn}}$ as applicable. Prompt tuning prepends l tunable vectors at the input layer and uses $l \times d$ number of parameters. Using MBART/BART as an example, we present the number of parameters used by several representative methods throughout our paper in Table 12, where adapter variants include sequential adapter, parallel adapter, scaled adapter and multi-head adapter.

.17 Full Results on Different Bottleneck Dimensions

Table 12: Number of tunable parameters of various parameter-efficient tuning methods with BART/M-BART models ($L = 12$) as an example.

Method	number of parameters
Prompt Tuning	$l \times d$
Prefix Tuning (attn)	$2ld \times 3 \times 12$
Adapter variants (attn)	$2rd \times 3 \times 12$
Adapter variants (ffn)	$2rd \times 2 \times 12$
LoRA (attn)	$4rd \times 3 \times 12$
LoRA (ffn)	$10rd \times 2 \times 12$
MAM Adapter (our proposed model)	$2ld \times 3 \times 12 + 2rd \times 2 \times 12$

Table 13: Performance on the test sets of abstractive summarization (XSum) and WMT EN-RO translation.

Method	# params (%)	XSum (R-1/2/L)	MT BLEU
Modified Representation: attention			
Prefix Tuning, $r = 200$	3.6	43.40/20.46/35.51	35.6
Prefix Tuning, $r = 512$	9.2	43.29/20.40/35.37	35.1
LoRA, $r = 200$	7.2	43.09/20.29/35.37	36.2
Sequential Adapter, $r = 200$	3.6	42.01/19.30/34.40	35.3
Sequential Adapter, $r = 512$	9.2	41.05/18.87/33.71	34.7
Parallel Adapter, $r = 200$	3.6	43.58/20.31/35.34	35.6
Parallel Adapter, $r = 512$	9.2	43.99/20.83/35.77	36.2
Modified Representation: FFN			
LoRA, $r = 102$	6.1	44.59/21.31/36.25	36.5
Sequential Adapter, $r = 200$	2.4	43.21/19.98/35.08	35.6
Sequential Adapter, $r = 512$	6.1	43.72/20.75/35.64	36.3
Sequential Adapter, $r = 1024$	12.3	43.95/21.00/35.90	36.7
Parallel Adapter, $r = 200$	2.4	43.93/20.66/35.63	36.4
Parallel Adapter, $r = 512$	6.1	44.35/20.98/35.98	37.1
Parallel Adapter, $r = 1024$	12.3	44.53/21.24/36.23	37.3

Appendix V: Prompt Consistency for Zero-Shot Task Generalization

.18 Datasets

Task	Dataset	#train set	#validation set	#labels	#prompts
NLI	RTE	2,490	277	2	10
	CB	250	57	3	15
	ANLI R1	16,946	1000	3	15
	ANLI R2	45,460	1000	3	15
	ANLI R3	100,459	1200	3	15
Compl.	COPA	400	100	2	8
	HellaSwag	39,905	10,042	4	4
	Story Cloze	-	1,871	2	5
Coref.	Winogrande	40,398	1,267	2	5
	WSC	554	104	2	10
WSD	WIC	5,428	637	2	10

Table 14: Statistics of the datasets

We present the statistics of the 11 datasets in Table 14. For the training-time tuning scenario, we use up to 10,000 data points from the training set for training if the train set contains more than 10,000 data points.

.19 Training Details

We use LoRA (Hu et al., 2021) as our parameter-efficient tuning model and set the bottleneck dimension of LoRA weight matrices to be 1 for both 3B and 11B models. For both models, we set the dropout probability for the the LoRA intermediate representations to be 0.3. Let α denote the scaling factor of LoRA that is used to scale the output of the LoRA layer before adding to the hidden states of the pre-trained model. We set α to be 4 and 2 respectively for the 3B and 11B model. The peak learning rates of the 3B and 11B models are set to be 3e-5 and 5e-5 respectively with a warm-up stage of 100

steps and polynomial learning rate scheduler. We train for a maximum of 1,500 steps. Note that the hyperparameters for the 3B model is tuned on the RTE dataset and used for other datasets. We did not tune the hyperparameters of the 11B model.

With respect to implementation details, at each update we first sample one input example $\mathbf{x}$ and apply multiple prompts to reformat it as $r_x^1(\mathbf{x}), \cdots, r_x^K(\mathbf{x})$, then we perform inference for them and randomly shuffle the predictions. Next we iterate over them with a batch size of 5/10 (3B/11B)[4] and use the shuffled predictions to supervise them to compute the distillation loss, this implements the swarm distillation mechanism in Eq. 7.2 and in fact approximates the expectation over paired prompts with K random pairs. We accumulate the gradients for 16 steps for one update so that each gradient descent is computed from 16 data examples. And we use 1 A40 GPU (45GB memory) to train the 3B model and 4 A40 GPUs with DeepSpeed Zero-2 (Ren et al., 2021) to train the 11B model. In general, training converges pretty fast and takes around 1 - 3 GPU hours for the 3B model and 2 - 6 hours for the 11B model depending on early stop points of different datasets. We use Adam (Kingma and Ba, 2014) as the optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 1e-6$.

For the Transformer (Vaswani et al., 2017) models with model dimension d, the feed-forward intermediate dimension m and number of layers l, the additional parameters used in LoRA with bottleneck dimension b is calculated as $b*(m+d)*2*l*2$. As we set b to be 1 for both the 3B and 11B models, the additional number of LoRA parameters is 1,671,168 for the T0-3B model ($d = 1024, m = 16384, l = 24$) and 6,389,760 for the T0-11b model ($d = 1024, m = 65536, l = 24$).

.20 Limitations of Our Work

We propose an unsupervised method for better zero-shot learner. There are two limitations of our work: (1) Because our method is operated in a fully unsupervised manner, there is no supervised development data for us to either select the best model or tune hyperparameters. Thus, we propose to use Fleiss' Kappa as our unsupervised development metric for model selection, which attains decent performance in most cases. However, we also see on very few datasets that the proposed metric fails to select the best checkpoints and hurt the model's performance. As discussed in §7.4.4, our method can be combined with few-shot learning where a few labeled data are provided and we believe this can largely alleviate the issues of model selection in the unsupervised setting. (2) The other limitation and at the same time an advantage of our method is that the proposed method can work well even with 10 unlabeled data points. This certainly makes our method a good candidate for the online setting where batches of test data come in a stream. However, as we discussed in §7.4.4, the performance of our model saturates quickly as we increase the number of unlabeled data, which means the performance of our method cannot scale well with tons of unlabeled data like self-supervised pretraining. As discussed in §7.5, we expect

[4]Because the GPU memory sometimes cannot handle all the prompts within one batch.

combining our method with few-shot learning setting / pre-training can lead to further improvements as the supervised signals may guide the model to a better local optimum.

Bibliography

Alessandro Achille and Stefano Soatto. 2018. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980. 4.2

Kofi P Adragni and R Dennis Cook. 2009. Sufficient dimension reduction and prediction in regression. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1906):4385–4405. 4.2

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884. 5.1

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*. 2.1.2

Antonios Anastasopoulos, Alessandro Cattelan, Zi-Yi Dou, Marcello Federico, Christian Federman, Dmitriy Genzel, Francisco Guzmán, Junjie Hu, Macduff Hughes, Philipp Koehn, et al. 2020. Tico-19: the translation initiative for covid-19. *arXiv preprint arXiv:2007.01788*. 3.4.2, .1

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, et al. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*. 1.1, 2.2.3, 5.1, 5.2.1, 5.4.3, .14

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*. 2.1.1, 4.1, 3

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*. 6.2.1

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike

Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts. 7.1

Philip Bachman, Ouais Alsharif, and Doina Precup. 2014. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3365–3373. 7.1

Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. 2020. Learning de-biased representations with biased representations. In *International Conference on Machine Learning*, pages 528–539. PMLR. 4.1

Sara Beery, Grant Van Horn, and Pietro Perona. 2018. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473. 4.1

Alexis Bellot and Mihaela van der Schaar. 2020. Generalization and invariances in the presence of unobserved confounding. *arXiv preprint arXiv:2007.10653*. 4.1

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*. 3.5.2

Aharon Ben-Tal, Dick Den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. 2013a. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357. 1.1, 4.4.1, 4.4.1, 4.5, 5.1, 5.2.2

Aharon Ben-Tal, Dick den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. 2013b. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357. 5.2.2, 5.3.1

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv e-prints*, pages arXiv–2106. (document), 6.2.2, 6.5, 6.4.2, 6.3

Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. 2018. Data-driven robust optimization. *Mathematical Programming, Series A*, 167(2):235–292. 5.2.2

Lucas Beyer, Xiaohua Zhai, Avital Oliver, and Alexander Kolesnikov. 2019. S4L: self-supervised semi-supervised learning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1476–1485. IEEE. 7.1

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of african-american english. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1119–1130. 4.6.1

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the

2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*. 6.4.1

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2021. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*. 8

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press. 5.3.2

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*. 1.1, 2.1.2, 2.1.2, 2.2.2, 3, 6.1, 7.1, 7.2, 7.4.1

Sébastien Bubeck. 2015. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357. .12.3

Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. 2.1.1, 4.3

Vishrav Chaudhary, Yuqing Tang, Francisco Guzmán, Holger Schwenk, and Philipp Koehn. 2019. Low-resource corpus filtering using multilingual sentence embeddings. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 261–266. 3.7

Jiaao Chen, Dinghan Shen, Weizhu Chen, and Diyi Yang. 2021a. Hiddencut: Simple data augmentation for natural language understanding with better generalizability. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4380–4390. 2.2.1

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157, Online. Association for Computational Linguistics. 2.2.1

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021b. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*. 1.1

Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758. 7.3.2

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, and et al. 2022. Palm: Scaling language modeling with pathways. 2.1.2, 2.2.2

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised se-

quence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics. 7.3.2, 7.3.2

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. 2.2.3, 3.1, 3.3.2, 3.5.1, .3

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451. 5.1, 5.2.1

Thomas M Cover. 1999. *Elements of information theory*. John Wiley & Sons. 4.2, .7.1

Imre Csiszár. 1967. Information-type measures of difference of probability distributions and indirect observation. *Studia Scientifica Mathematica Hungary*, 2:299–318. 5.3.1, .12.3

Milan Cvitkovic and Günther Koliander. 2019. Minimal achievable sufficient statistic learning. volume 97, pages 1465–1474. PMLR. 4.1, 4.2

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*. 1.1

Shai Ben David, Tyler Lu, Teresa Luu, and Dávid Pál. 2010. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 129–136. JMLR Workshop and Conference Proceedings. 4.3

Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. 2019. Racial bias in hate speech and abusive language detection datasets. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 25–35. 1.1

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William Cohen. 2021. Mention memory: incorporating textual knowledge into transformers through entity mention attention. *arXiv preprint arXiv:2110.06176*. 8

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124. 7.4.1

Erick Delage and Yinyu Ye. 2010. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58(3):595–612. 5.2.2

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. Bert: Pre-training of deep

bidirectional transformers for language understanding. In *NAACL-HLT (1)*. 1.1, 2.1.2, 3.3.2, 3.5.2, 6.1, .15.3

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics. 1.1, 2.2.2, 7.1

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019c. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186. 5.1, 5.2.1

Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273. 1.1, 8

Zi-Yi Dou, Junjie Hu, Antonios Anastasopoulos, and Graham Neubig. 2019. Unsupervised domain adaptation for neural machine translation with domain-aware feature embeddings. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Hong Kong. 2.2.2

John Duchi, Peter Glynn, and Hongseok Namkoong. 2016. Statistics of robust optimization: A generalized empirical likelihood approach. *arXiv preprint arXiv:1610.03425*. 1.1, 4.4.1, 5.1

John C Duchi, Tatsunori Hashimoto, and Hongseok Namkoong. 2019. Distributionally robust losses against mixture covariate shifts. 4.4.1, 4.5

John C. Duchi and Hongseok Namkoong. 2019. Variance-based regularization with convex objectives. *Journal of Machine Learning Research*, 20(68):1–55. 5.3.1

John C. Duchi and Hongseok Namkoong. 2020. Learning models with uniform performance via distributionally robust optimization. *Annals of Statistics*, to appear. 5.3.1

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. 3.1, .5.1

Evgeniui Borisovich Dynkin. 2000. Necessary and sufficient statistics for a family of probability distributions. *Selected Papers of EB Dynkin with Commentary*, 14:393. 4.1

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*. 1.1, 2.1.2

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transac-*

tions of the Association for Computational Linguistics, 9:1012–1031. 7.1, 7.3.2, 7.3.2

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, et al. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48. 1.1

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567. 2.1.2

Amir Feder, Katherine A Keith, Emaad Manzoor, Reid Pryzant, Dhanya Sridhar, Zach Wood-Doughty, Jacob Eisenstein, Justin Grimmer, Roi Reichart, Margaret E Roberts, et al. 2021. Causal inference in natural language processing: Estimation, prediction, interpretation and beyond. *arXiv preprint arXiv:2109.00725*. 2.2.4, 8

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*. 6.1, 8

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988. 2.2.1

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875. 5.1

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378. 3.2, 7.3.3

Erick Fonseca, Lisa Yankovskaya, André F. T. Martins, Mark Fishel, and Christian Federmann. 2019. Findings of the WMT 2019 shared tasks on quality estimation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 1–10, Florence, Italy. Association for Computational Linguistics. 3.1, 3.5.4

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):1–30. 4.6.1

Wee Chung Gan and Hwee Tou Ng. 2019. Improving the robustness of question answering systems to question paraphrasing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6065–6075. 2.1.2

Rui Gao and Anton J Kleywegt. 2016. Distributionally robust stochastic optimization with wasserstein distance. *arXiv preprint arXiv:1604.02199*. 4.4.1

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830. 2.1.2

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323. 2.2.4

Sahaj Garg, Vincent Perot, Nicole Limtiaco, Ankur Taly, Ed H Chi, and Alex Beutel. 2019. Counterfactual fairness in text classification through robustness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 219–226. 2.2.4

Bernhard C Geiger. 2020. On information plane analyses of neural network classifiers–a review. *arXiv preprint arXiv:2003.09671*. .7.2

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166. 2.1.2

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of EMNLP*. 1.1, 6.4.4

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*. 2.1.2

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6904–6913. 2.1.2, 4.1

Demi Guo, Yoon Kim, and Alexander Rush. 2020. Sequence-level mixed sample data augmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5547–5552, Online. Association for Computational Linguistics. 2.2.1

Demi Guo, Alexander M Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of ACL*. 6.2.2, 6.4.2

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112. 2.1.2, 4.1, 4.3, 4.6.1

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*. 8

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The flores evaluation datasets for low-resource machine translation: Nepali–english and sinhala–english. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6100–6113. 3.7

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*. 5.1

Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. 2018a. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*, pages 1929–1938. 2.1.1, 2.2.3, 4.1

Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. 2018b. Fairness without demographics in repeated loss minimization. In *Proceedings of the 35th International Conference on Machine Learning*. 5.2.2, 5.3.1

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. (document), 2.2.1, 3.4, 3.6.1, 3.6.2, 7.4.2

Junxian He, Zhisong Zhang, Taylor Berg-Kirkpatrick, and Graham Neubig. 2019. Cross-lingual syntactic transfer through unsupervised adaptation of invertible projections. In *The 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, Florence, Italy. 2.2.2

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of ICCV*. .15.3

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 4.6.2, .10.1

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *ArXiv preprint*, abs/1503.02531. 7.3.2

P Hitzler and MK Sarker. 2022. Answering natural-language questions with neuro-symbolic knowledge bases. *Neuro-Symbolic Artificial Intelligence: The State of the Art*, 342:126. 8

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780. .9

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text

degeneration. *arXiv preprint arXiv:1904.09751*. 1.1

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *Proceedings of ICML*. 2.2.2, 6.1, 6.2.2, 6.2.2, 6.4.4, 7.1

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *10th International Conference on Learning Representations, ICLR 2022*, abs/2106.09685. 2.2.2, 6.1, 6.2.2, 7.3.3, .19

Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. 2018. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning*, pages 2029–2037. PMLR. 4.4.1, 4.5

Zhaolin Hu and L Jeff Hong. 2013. Kullback-leibler divergence constrained distributionally robust optimization. *Available at Optimization Online*. 4.4.1

Hisham Husain. 2020. Distributional robustness with ipms and links to regularization and gans. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems (NIPS)*. 4.5

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of NAACL-HLT*, pages 1875–1885. 2.1.2

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*. 1.1

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031. 2.1.2

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. 2015. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2.2.3

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438. 1.1, 2.2.2, 7.1, 7.3.2, 7.4.2

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351. 5.1

Marcin Junczys-Dowmunt. 2018. Dual conditional cross-entropy filtering of noisy parallel corpora. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 888–895. 3.6.1,

3.7

Daniel Kang and Tatsunori Hashimoto. 2020. Improved natural language generation via loss truncation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. (document), 3.4, 3.6.1, 3.6.2

Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. 2021. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 353–362. 8

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*. 2.2.4

Fabio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André FT Martins. 2019. Openkiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 117–122. (document), 3.3, 3.5.4

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Nearest neighbor machine translation. In *International Conference on Learning Representations*. 8

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*. 8

Yoon Kim and Alexander M Rush. 2016a. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327. 3.3.2, 3.5.1, .4

Yoon Kim and Alexander M. Rush. 2016b. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics. 7.3.2

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. .2, .3, .10.1, .14, .15.2, .19

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526. 1.1

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*. 3.5.2

Philipp Koehn, Francisco Guzmán, Vishrav Chaudhary, and Juan Pino. 2019. Findings of the wmt 2019

shared task on parallel corpus filtering for low-resource conditions. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 54–72. 3.1, 3.7

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39. 3.4.2

Allison Koenecke, Andrew Nam, Emily Lake, Joe Nudell, Minnie Quartey, Zion Mengesha, Connor Toups, John R Rickford, Dan Jurafsky, and Sharad Goel. 2020. Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, 117(14):7684–7689. 2.1.1, 4.1

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery, et al. 2020. Wilds: A benchmark of in-the-wild distribution shifts. *arXiv preprint arXiv:2012.07421*. 2.1.1, 4.1, 4.1

Artemy Kolchinsky, Brendan D Tracey, and Steven Van Kuyk. 2019. Caveats for information bottleneck in deterministic scenarios. In *International Conference on Learning Representations*. 4.2

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71. 5.4.2, .14

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86. 4.2

Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed Chi. 2020. Fairness without demographics through adversarially reweighted learning. *Advances in neural information processing systems*, 33:728–740. 2.2.3

Samuel Läubli, Rico Sennrich, and Martin Volk. 2018. Has machine translation achieved human parity? a case for document-level evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4791–4796, Brussels, Belgium. Association for Computational Linguistics. 3

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. 2021. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34. 1.1, 8

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of EMNLP*. 2.2.2, 6.1, 6.2.2, 6.3

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. 7.4.1

Daniel Levy, Yair Carmon, John C. Duchi, and Aaron Sidford. 2020. Large-scale methods for distribu-

tionally robust optimization. In *Advances in Neural Information Processing Systems 33*. 2.2.3, 5.2.2, 5.5, .11, .12.2

David Lewis. 2013. *Counterfactuals*. John Wiley & Sons. 4.1

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. (document), 1.1, 3.3.1, 3.5.1, 6.1, 6.4.1, 6.3, .15.1, .15.2

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474. 8

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021a. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1000–1008. 2.1.2

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021b. Paq: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115. 8

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. 7.4.1

Jiwei Li, Xinlei Chen, Eduard H Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *HLT-NAACL*. 8

Xiang Lisa Li and Percy Liang. 2021a. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics. 2.1.2, 2.2.2

Xiang Lisa Li and Percy Liang. 2021b. Prefix-tuning: Optimizing continuous prompts for generation. In

Proceedings of ACL. 6.1, 6.2.2, 6.4.2, 6.4.4, 6.3, .15.2, .15.3

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. 6.4.1

Chin-Yew Lin and Eduard Hovy. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics. 3.1

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988. 2.2.3

Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. 2021a. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pages 6781–6792. PMLR. 2.2.3

Haochen Liu, Wentao Wang, Yiqi Wang, Hui Liu, Zitao Liu, and Jiliang Tang. 2020a. Mitigating gender bias for neural dialogue generation with adversarial learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 893–903, Online. Association for Computational Linguistics. 2.1.2

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*. 1.1, 6.1, 6.2.2, 7.1, 7.2

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021c. GPT understands, too. *arXiv:2103.10385*. 6.2.2

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020b. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742. (document), 6.3, .15.1, .15.2

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020c. Multilingual denoising pre-training for neural machine translation. 3.4.2, .1

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020d. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*. 6.4.1

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 preprint arXiv:1907.11692*. 2.1.2, 2.2.2, 3.1, 3.3.2, 3.5.1, 4.6.2, 6.4.1, 7.1, .10.1, .15.1, .15.2

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738. 4.6.1

David Lopez-Paz. 2016. *From dependence to causation.* Ph.D. thesis, University of Cambridge. 4.1

Charles Lovering, Rohan Jha, Tal Linzen, and Ellie Pavlick. 2021. Predicting inductive biases of pre-trained models. In *International Conference on Learning Representations (ICLR).* 4.1

Ruixuan Luo, Jingjing Xu, Yi Zhang, Xuancheng Ren, and Xu Sun. 2019. Pkuseg: A toolkit for multi-domain chinese word segmentation. *CoRR*, abs/1906.11455. .2

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Proceedings of NeurIPS.* 6.3.2, 6.4.2

Gary Marcus and Ernest Davis. 2020. Gpt-3, bloviator: Openai's language generator has no idea what it's talking about. *MIT Technology Review.* 1.1, 2.1.2, 3

Marianna Martindale, Marine Carpuat, Kevin Duh, and Paul McNamee. 2019. Identifying fluently inadequate output in neural and statistical machine translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 233–243. 3

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan Thomas Mcdonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. (document), 1.1, 2.1.2, 3, 3.1, 3.2, 3.2, 3.4.1, 3.5.3, .1, 1, .4, .5.1

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448. 2.1.2, 4.1, 4.1

Cade Metz and Adam Satariano. 2020. An algorithm that grants freedom, or takes it away. *The New York Times.* 1.1

Paul Michel, Tatsunori Hashimoto, and Graham Neubig. 2021. Modeling the second player in distributionally robust optimization. In *International Conference on Learning Representations (ICLR).* 4.5, 4.6.1

Paul Michel, Tatsunori Hashimoto, and Graham Neubig. 2022. Distributionally robust models with parametric likelihood ratios. In *International Conference on Learning Representations (ICLR)*, Virtual. 2.2.3

Paul Michel, Xian Li, Graham Neubig, and Juan Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Minneapolis, USA. 1.1, 2.1.2

George A Miller. 1998. *WordNet: An electronic lexical database.* MIT press. 3.5.2

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993. 7.1, 7.3.2

Nafise Sadat Moosavi, Marcel de Boer, Prasetya Ajie Utama, and Iryna Gurevych. 2020. Improving

robustness by augmenting training sentences with predicate-argument structures. *arXiv preprint arXiv:2010.12510*. 2.2.1

John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020. Reevaluating adversarial examples in natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3829–3839. 2.1.2

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics. 7.4.1

Mathias Müller, Annette Rios, and Rico Sennrich. 2019. Domain robustness in neural machine translation. *arXiv preprint arXiv:1911.03109*. 3.4.2

Hongseok Namkoong and John C. Duchi. 2016. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in Neural Information Processing Systems 29*. .12.3

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. (document), 3.4.1, 6.2, 6.4.1

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609. 5.3.2, .12, .12.1

Arkadi Nemirovski. 2004. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251. .12

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880. 2.2.2, 5.1

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics. 7.4.1

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. 6.4.2

Yonatan Oren, Shiori Sagawa, Tatsunori Hashimoto, and Percy Liang. 2019. Distributionally robust language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4218–4228. 2.2.3, 4.1, 4.4.1, 4.4.1, 4.5, 4.6.2, 5.2.2, 5.2.2, 5.3.1, 5.3.2, 5.5, 3, .12.2

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 *Proceedings of NAACL-HLT 2019: Demonstrations*. 3.5.1, 5.4.2, .2, .3, .3, .10.1

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. 2.1.2

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002a. Bleu: a method for automatic evaluation of machine translation. In *ACL 2002*. 3.1, 3.6.2, 5.4.2

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002b. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*. 6.4.1

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*. 6.1

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of EACL*. 6.1, 6.2.2, 6.3.2, 6.3

Minh Quang Pham, Josep-Maria Crego, and François Yvon. 2021. Revisiting multi-domain machine translation. *Transactions of the Association for Computational Linguistics*, 9:17–35. 1.1

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*. 2.1.2

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics. 7.4.1

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001. 5.1

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics. 3.1, 3.7, 5.4.2

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915. 3

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA. 5.4.1

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics. 7.4.2

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*. 6.1

Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2008. *Dataset shift in machine learning*. Mit Press. 2.1.1

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. technical report. 6.1

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. *Journal of Machine Learning Research*, 21(140):1–67. 6.4.2, 7.1, 7.4.1

Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 4.6.4

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. Zero-offload: Democratizing billion-scale model training. *arXiv preprint arXiv:2101.06840*. .19

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097. 2.1.2

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912. 2.2.4

R Tyrrell Rockafellar, Stanislav Uryasev, et al. 2000. Optimization of conditional value-at-risk. *Journal*

of risk, 2:21–42. 4.4.1

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*. 7.4.1

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280. 1.1, 2.1.2, 3, 3.4.1

Tim Roughgarden. 2016. *Twenty Lectures on Algorithmic Game Theory*. Cambridge University Press. 5.3.2

Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18. 2.2.2

Masoud Jalili Sabet, Philipp Dufter, and Hinrich Schütze. 2020. Simalign: High quality word alignments without parallel training data using static and contextualized embeddings. *arXiv preprint arXiv:2004.08728*. 3.5.2

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. 2020a. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia. 1.1, 2.2.3, 4.1, 4.4.2, 4.6.1, 4.6.1, 4.6.2, 5.1, 5.2.2, 5.5, .10.2

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020b. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning (ICML)*. 4.1

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1163–1171. 7.1

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *ArXiv preprint*, abs/2110.08207. 2.1.2, 7.1, 7.1, 7.2, 7.3.1, 7.4.1, 7.4.1, 5, 7.4.3, 7.5

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A Smith. 2019. The risk of racial bias in hate speech detection. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1668–1678. 1.1, 4.6.1

Maarten Sap, Swabha Swayamdipta, Laura Vianna, Xuhui Zhou, Yejin Choi, and Noah A Smith. 2021.

Annotators with attitudes: How annotator beliefs and identities bias toxic language detection. *arXiv preprint arXiv:2111.07997*. 2.1.2

Ravid Schwartz-Ziv and Naftali Tishby. 2017. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*. 4.2, .7.1, .7.1, .7.2

H Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371. 3.1, 3.6.1

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. 3.4.1

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. 3.6.2

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96. 2.2.1

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics. .2

Ohad Shamir, Sivan Sabato, and Naftali Tishby. 2010. Learning and generalization with the information bottleneck. volume 411, pages 2696–2711. Elsevier. 4.2, 4.3, .7.1

Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich, Aurélie Herbelot, Moin Nabi, Enver Sangineto, and Raffaella Bernardi. 2017. Foil it! find one mismatch between image and language caption. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 255–265. 2.2.4

Dinghan Shen, Mingzhi Zheng, Yelong Shen, Yanru Qu, and Weizhu Chen. 2020. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *arXiv preprint arXiv:2009.13818*. 2.2.1

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244. 2.1.1

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. 2016. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*. 8

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*. 6.4.1

Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. 2020. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *34th Conference on Neural Information Processing Systems (NeurIPS)*, 33. 4.1

Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramón Astudillo, and André Martins. 2018. Findings of the wmt 2018 shared task on quality estimation. Association for Computational Linguistics. (document), 3.1, 3.5.4, 3.3

Lucia Specia, FrÃ©dÃ©ric Blain, Marina Fomicheva, Erick Fonseca, Vishrav Chaudhary, Francisco GuzmÃ¡n, and AndrÃ© F. T. Martins. 2020. Findings of the wmt 2020 shared task on quality estimation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 743–764, Online. Association for Computational Linguistics. 3.1, 3.5.4

Lucia Specia, Najeh Hajlaoui, Catalina Hallett, and Wilker Aziz. 2011. Predicting machine translation adequacy. In *Machine Translation Summit*, volume 13, pages 19–23. 3.1

Statista. 2022. Distribution of reddit users worldwide as of january 2022, by gender. 1.1

Jiao Sun, Xuezhe Ma, and Nanyun Peng. 2021. AESOP: Paraphrase generation with adaptive syntactic control. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5176–5189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. 2.2.1

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9229–9248. PMLR. 7.4.2

Josh Tenenbaum. 2018. Building machines that learn and think like people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 5–5. 4.1

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*. 3.1

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*. 4.2, 4.3, .7.1

Antonio Torralba and Alexei A Efros. 2011. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE. 4.1

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-

speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the North American chapter of the association for computational linguistics on human language technology-volume 1*, pages 173–180. Association for Computational Linguistics. 3.5.3

Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, and Manaal Faruqui. 2019. Attention interpretability across nlp tasks. *arXiv preprint arXiv:1909.11218*. 8

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 *Advances in neural information processing systems*, pages 5998–6008. 1.1, 2.1.2, 3, 3.4.1, 3.4.2, 5.1, 5.3.2, 5.4.2, 6.2.1, 7.3.3, .19

Tom Veniat, Ludovic Denoyer, and MarcAurelio Ranzato. 2021. Efficient continual learning with modular networks and task-driven priors. In *International Conference on Learning Representations*. 1.1

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020a. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. 3.1, .5.1

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275. 7.4.1

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. 2021a. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840*. 1.1

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. 2.1.2, 3.1, 3.4.2

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. 2021b. Tent: Fully test-time adaptation by entropy minimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. 7.4.2

Rose E Wang, Esin Durmus, Noah Goodman, and Tatsunori Hashimoto. 2022. Language modeling via stochastic processes. In *International Conference on Learning Representations*. 2.1.2

Serena Wang, Wenshuo Guo, Harikrishna Narasimhan, Andrew Cotter, Maya R Gupta, and Michael I Jordan. 2020b. Robust optimization for fairness with noisy protected groups. *34th Conference on Neural Information Processing Systems (NeurIPS)*. 4.1

Xinyi Wang, Hieu Pham, Paul Michel, Antonios Anastasopoulos, Jaime Carbonell, and Graham Neu-

big. 2020c. Optimizing data usage via differentiable rewards. In *International Conference on Machine Learning*, pages 9983–9995. PMLR. 2.2.3

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020d. Balancing training for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8526–8537. 2.2.3, 5.1, 5.2.1, 5.4.1, 5.4.2, 1, 2, .14, .14

Yong Wang, Longyue Wang, Shuming Shi, Victor OK Li, and Zhaopeng Tu. 2020e. Go from the general to the particular: Multi-domain translation with domain transformation networks. In *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, New York, USA. 3.4.2, 3.6.2, .1

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. 2017. Learning to model the tail. *Advances in Neural Information Processing Systems*, 30. 2.2.3

Zirui Wang, Zachary C Lipton, and Yulia Tsvetkov. 2020f. On negative interference in multilingual language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4438–4450. 5.1

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. 2021c. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*. 5.1

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *ArXiv preprint*, abs/2109.01652. 7.1, 7.2, 7.5

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388. 2.2.1

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*. 3

Junfeng Wen, Chun-Nam Yu, and Russell Greiner. 2014. Robust learning under uncertain test distributions: Relating covariate shift to model misspecification. In *ICML*, pages 631–639. 4.3

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018a. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. 4.6.1

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018b. A broad-coverage challenge corpus for

sentence understanding through inference. In *Proceedings of NAACL*. 6.4.1

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP: System Demonstrations*. 6.4.1, .15.1

Tongshuang Wu, Marco Túlio Ribeiro, Jeffrey Heer, and Daniel S. Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6707–6723. Association for Computational Linguistics. 2.2.4

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*. 3

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020a. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33. 7.1, 7.1, 7.3.2, 7.3.2

Qizhe Xie, Minh-Thang Luong, Eduard H. Hovy, and Quoc V. Le. 2020b. Self-training with noisy student improves imagenet classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10684–10695. IEEE. 7.3.2, 7.4.2

Depeng Xu, Shuhan Yuan, and Xintao Wu. 2019. Achieving differential privacy and fairness in logistic regression. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 594–599. 1.1

John R Zech, Marcus A Badgeley, Manway Liu, Anthony B Costa, Joseph J Titano, and Eric Karl Oermann. 2018. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study. *PLoS medicine*, 15(11):e1002683. 2.1.1

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics. 7.4.1

Runtian Zhai, Chen Dan, Zico Kolter, and Pradeep Ravikumar. 2022. Understanding why generalized reweighting does not improve over erm. *arXiv preprint arXiv:2201.12293*. 2.2.3

Boliang Zhang, Ajay Nagesh, and Kevin Knight. 2020a. Parallel corpus filtering via pre-trained language models. *arXiv preprint arXiv:2005.06166*. 3.7

Marvin Mengxin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. 2020b. Adaptive risk minimization: A meta-learning approach for tackling group shift. 2.2.3

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*. 3.1

Yu Zhang, James Qin, Daniel S Park, Wei Han, Chung-Cheng Chiu, Ruoming Pang, Quoc V Le, and Yonghui Wu. 2020c. Pushing the limits of semi-supervised learning for automatic speech recognition. *ArXiv preprint*, abs/2010.10504. 7.4.2

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. .4

Chunting Zhou, Junxian He, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations (ICLR)*, Virtual. 1.2, 2.1.2, 7.1, 7.3.3

Chunting Zhou, Daniel Levy, Xian Li, Marjan Ghazvininejad, and Graham Neubig. 2021a. Distributionally robust multilingual machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Punta Cana, Dominican Republic. 1.2

Chunting Zhou, Xuezhe Ma, Paul Michel, and Graham Neubig. 2021b. Examining and combating spurious features under distribution shift. In *International Conference on Machine Learning (ICML)*, Virtual. 1.2

Chunting Zhou, Xuezhe Ma, Di Wang, and Graham Neubig. 2019. Density matching for bilingual word embedding. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Minneapolis, USA. 3.5.2

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Francisco Guzmán, Luke Zettlemoyer, and Marjan Ghazvininejad. 2021c. Detecting hallucinated content in conditional neural sequence generation. In *Findings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP Findings)*, Virtual. 1.2

Yaoming Zhu, Jiangtao Feng, Chengqi Zhao, Mingxuan Wang, and Lei Li. 2021. Serial or parallel? plugable adapter for multilingual machine translation. *arXiv preprint arXiv:2104.08154*. 6.3.3

Ran Zmigrod, Sabrina J Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1651–1661. 2.2.4

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William

Fedus. 2022. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*. 8

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575. 5.1