Data-Centric Methods for Decentralizing Large Language Models

Suchin Gururangan

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2024

Reading Committee:
Noah A. Smith, Co-Chair
Luke Zettlemoyer, Co-Chair
Katharina Reinecke

Program Authorized to Offer Degree:
Computer Science and Engineering

© Copyright 2024

Suchin Gururangan

University of Washington

Abstract

Data-Centric Methods for Decentralizing Large Language Models

Suchin Gururangan

Co-chairs of the Supervisory Committee:

Noah A. Smith
Computer Science and Engineering

Luke Zettlemoyer
Computer Science and Engineering

Large language models (LMs) rely on massive textual datasets crawled from the Internet. In this thesis, I argue that many fundamental limitations of LMs (e.g., extreme costs, legal risks, and harmful behavior) are a direct result of monolithic, centralized, and homogeneous treatment of data. I first deconstruct the notion of a general-purpose corpus; I empirically show that current pretraining corpora implicitly favor text from the most powerful authors in society, and cannot feasibly represent all possible downstream use cases. Given this result, I highlight the importance of customizing LMs to new language variations using adaptive pretraining. I then propose a new class of LMs that are fundamentally decentralized, where components (or experts) of the LM are specialized to distinct domains in the training corpus, and experts are conditionally updated based on the domain of the incoming document. Our new models address the limitations of centralization by being rapidly customizable (with the ability to mix, add, or remove experts after training), embarrassingly parallel (requiring no communication between experts), and sparse (needing only a few experts active at a time for inference). Key to these proposals are their data-centric nature; for example, I carefully explore what constitutes the domains to which experts specialize, and reflect on the data sources used to train LMs. I close by describing avenues for future work on decentralization techniques, with a focus on providing options for data opt-out, efficient customization, and cheaper scaling into massive, heterogeneous datasets.

Acknowledgements

Thank you to my advisors, Noah A. Smith and Luke Zettlemoyer; I am deeply grateful to them for all the unequivocal support over the years. Due to their guidance, I have come out of my PhD a much improved writer, researcher, and mentor.

My life changed dramatically because Noah believed in my potential as a researcher six years ago and invited me into his group as a research assistant. I thank him for teaching me to stay grounded and value-driven regardless of larger trends in the field and research community.

Luke taught me to be bold to push boundaries of machine learning and NLP. He also taught me to carry a basic love and passion for wild scientific ideas, regardless of where you are in your career.

I'd also like to thank my mentor at FAIR, Mike Lewis. Mike taught me to appreciate and strive for simple yet elegant ideas and to embrace the exciting world of scale.

Thank you to Emily Bender and the Linguistics Department for creating the CLMS program. This program changed my life, as it gave me an opportunity to restart my career in research well after I left my undergraduate institution. Programs like CLMS that reduce the barrier to entry into NLP research are critical to the future of the field.

I would also like to thank Katharina Reinecke for her ethics class. This course gave me an avenue to study sociotechnical aspects of NLP which motivate this thesis. Thank you Katharina and Shane Steinert-Threlkeld for being on my thesis committee as well!

Thank you to the early mentors of my NLP research career: Roy Schwartz, Dallas Card, Omer Levy, and Jesse Dodge. I learned so much about how to be an effective researcher from these folks early on, and I would not be here without their guidance and support.

And thank you to my *earliest* research mentors at UChicago: Jason Maclean, Mukta Vaidya, Alex

Sadovsky, and Melissa Runfeldt. You helped me find my passion for science and to learn about exciting applications of machine learning.

Thank you so much to all my friends, mentors, and collaborators at UW and beyond: Margaret Li, Sewon Min, Mitchell Wortsman, Gabriel Ilharco, Mengzhou Xia, Niklas Stöhr, Eric Wallace, Hila Gonen, Victor Zhong, Sofia Serrano, Sarah Dreier, Emily Gade, Iz Beltagy, Doug Downey, Ludwig Schmidt, Julian Michael, Ari Holtzman, Tim Dettmers, Lucy Li, Maarten Sap, Waleed Ammar, Ana Marasovic, Kyle Lo, Luca Soldaini, Amandalynne Paullada, Terra Blevins, Alisa Liu, Weijia Shi, Ari Morcos, Pradeep Dasigi, Kiron Lebeck, Niel Lebeck, and many more folks.

I'd like to thank the undergraduate and masters students I had the deep pleasure of working with, and whom I learned a lot from: Kai Nylund, Sam Gehman, Aashna Sheth, Karishma Mandyam, Tam Dang, Machel Reid, and Leroy Wang.

And last but not least, my family:

Thank you so much Mom, for teaching me that no mountain is insurmountable. Dad, thank you for being my role model of work ethic, inquisitiveness, and healthy skepticism. Thanks to Sunjeev for all the good times and laughter.

Dexy, Kili, and Indy, thank you for your unconditional love, for teaching me how to be patient and responsible human, and for teaching me how to appreciate the simple things in life.

Finally, thank you to Swabha: you were there from the very beginning of this journey; you have been my greatest champion, and my greatest rock when things get hard. Your belief in me is the foundation of this PhD, and none of this would be possible without you by my side.

DEDICATION

To my dear family: Swabha, Kili, and Dexy

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To build a system that understands or generates natural language, today's best approach is to start with a *language model* pretrained on large quantities of text. Language models, driven by deep neural networks [Goodfellow et al., 2016], are trained to estimate the probability of the next token of a sequence. They have proved to be powerful tools that can learn to perform complex natural language tasks, such as programming assistance [Rozière et al., 2023], strategic dialogue [Bakhtin et al., 2022], and instruction following [Zhang et al., 2023].

The strength of this simple task — next-token prediction — is a reflection of the complexities of the data that underpins it. Natural language is eclectic and multifaceted; it varies along a number of axes, including sociolinguistic [Biber, 1988] and demographic [Rickford, 1985a] variables, community membership [Lucy and Bamman, 2021a], end-tasks [Wang et al., 2022b], and time [Lazaridou et al., 2021a]. The diversity of language use reflects the diversity of human experience. However, despite the heterogeneity of natural language, conventional approaches to pretrain language models treat data *homogeneously*.

Most language models are pretrained on data from a select few sources on the Internet, especially from large web crawl services like Common Crawl. Web crawl is perceived to be wholly representative of activity on the Internet [Brügger, 2011], and the enormity of data available from web crawl snapshots contributes to this perception.[1] As we show in Chapter 2, the data is in fact heavily influenced by the messy, imprecise, and opaque nature of the web archival process, as well as the use of automated filtration

[1] A COMMON CRAWL monthly snapshot contains almost 1TB of text.

methods to identify *high quality* documents to train on. We demonstrate that quality filters, in particular, place implicit value judgements on whose language is more likely to be included in the training data, favoring authors in hegemonic social positions. Therefore, we argue that despite its size, a language model's training corpus cannot sufficiently capture all possible language variations that the model might be exposed to during inference.

Exacerbating the *laissez faire* [Bender et al., 2021a] treatment of data is the fact that conventional language models are also trained *densely*: all training data is used to update all parameters. This leaves variation in the data to be implicitly discovered after training, assuming that the language model will fit all distributions equally well. This assumption does not hold in practice. Even if training data is sourced from many domains, dense training can in practice emphasize subsets of the data in proportion to their ease of access [Oren et al., 2019b; Fan et al., 2020], limiting generalization to less prevalent domains. Indeed, due to inherent biases in the makeup of Internet users that produce the data that models are trained on, even the best performing language models frequently engage in toxic language commonly found on Internet platforms [Gehman et al., 2020a], fail to properly capture many dialectical variations [Ziems et al., 2023; Mengesha et al., 2021], emphasize the most high-resource language families [Lin et al., 2022], and are implicitly biased towards the time period that dominates their training corpus [Luu et al., 2022; Lazaridou et al., 2021b]. These shortcomings are difficult and expensive to rectify after training.

In this thesis, we envision a new framework for *decentralized* language models which addresses these core issues in data selection and training. Our insight is that *flexibility* and *customization* after training are key mechanisms to tackle these problems: we want users, not model designers, to specify the data coverage of the model based on their use case, with strong guarantees on model behavior at inference time. We re-design the training process to build a language model can be collectively contributed to by researchers with diverse interests and data access, not just those with the most amount of resources. As opposed to other mechanisms for decentralized training [Borzunov et al., 2023; Yuan et al., 2023], the methods we focus on in this thesis are borne out of making use of language variation in the data.

We motivate this framework by first demonstrating the effectiveness of *continued pretraining* to improve performance of the language model on downstream domains and tasks, especially those that are not represented well in the pretraining data (Chapter 3). We propose multiple ways of adapting language models

to these targets: for example, through continued pretraining on unlabeled data, or, when such data is not available, targeted selection of domain or task-specific data from larger corpora.

This basic technique leads us to explore new ways to incorporate language variation during pretraining, as an alternative to dense training. Our proposed methodology is based on *conditional computation*, a technique in which we dynamically update parameters of the language model based on the domain of the incoming document (Chapter 4). This enables us to disentangle parameters into specialized *experts* that can be added or removed at any time to cheaply customize model behavior. In Chapter 5, we build on this technique to create language models that can be built asynchronously (where users can contribute experts completely independently), sparse (where users only need to interact with a subset of experts at a time), and modular (where users can shape the data coverage of the model after training). Our new class of language models improve over dense baselines as data and compute grows, providing a simple and effective way to train customizable language models on massive, heterogeneous datasets.

As datasets and computational budgets for training language models increase, and the impact of language models on society grow, the results of this thesis imply that the issues of treating data homogeneously cannot be ignored. We demonstrate that respecting language variation when training language models can lead to more efficient scaling into massive datasets, as well as exciting new modular capabilities for natural language processing systems that empower end-users. We close in Chapter 6 with many areas of future work.

Chapter 2

Deconstructing the Idea of a General-Purpose Corpus

The last few years of language modeling research have been driven by a quest for scale. Scale has materialized in three axes: the amount of training data, the amount of compute (or time) used to train the model, and the number of model parameters fitted to the data [Kaplan et al., 2020]. Recent studies have emphasized the importance of the *amount of training data*; one can train a smaller model on many more tokens and achieve similar (or better) downstream performance as compared to much larger models [Hoffmann et al., 2022].

As such, language model providers now source massive amounts of text to train language models. The Internet serves as the primary source of large training corpora, typically gathered from community resources (e.g., Wikipedia; Liu et al. 2019a) or web dumps (e.g., WebText, Common Crawl; Radford et al. 2019a, Brown et al. 2020). The sheer size of these corpora has contributed to the notion that web crawl data are in some sense *general purpose*, or that they enable generalization to any number of new distributions at test time. Indeed, web dumps like Common Crawl offer the promise of more diverse text than what is available in curated resources. However, we argue in this chapter that regardless of their size, it is impossible for any corpus to be universal.

We deconstruct this notion by first tracing research and philosophy that suggest archives are not just passive records or artifacts of human activity, but are rather active agents that shape and legitimize certain epistemologies and ideologies (§2.1). We then survey modern approaches to web archiving. We highlight

that, like the techniques that came before it, the web archival process involves subjectivity about what and how data gets preserved (§2.2).

We then turn our attention to the process of *quality filtering*, which we argue introduces even more subjectivity into web corpora. Much of the web consists of frequently replicated boilerplate (e.g., privacy policies), code (e.g., HTML and Javascript), pornography, hate speech, and more. Automated approaches, typically referred to as **quality filters**,[1] are often applied in an effort to remove this undesirable content from training data. These filters include code removers [Gao et al., 2020], heuristics [Rae et al., 2021a], stopwords [Raffel et al., 2020], and classifiers [Brown et al., 2020; Wenzek et al., 2020].

Although quality filtering is often treated as a relatively neutral preprocessing step, it necessarily implies a value judgment: which data is assumed to be of sufficiently high quality to be included in the training corpus? More concretely, when a quality filter is a classifier trained on instances assumed to be of high (and low) quality, the selection of those examples will impact the language model and any downstream technology that uses it. Many filters use Wikipedia, BOOKCORPUS, and newswire to represent high quality text. But what texts are excluded as a result? Because natural language varies with social and demographic variables [Rickford, 1985b; Eckert, 1989; Labov, 2006; Blodgett et al., 2016; Hovy and Yang, 2021; Lucy and Bamman, 2021b, *inter alia*], we can also ask *whose* language will be excluded.

We then analyze the handful of data sources used to construct training corpora for many language models and assumed to be of high quality (§2.3). The systematic authorship biases in these datasets motivate the study that follows, in which we replicate the quality filter from Brown et al. [2020]. We apply this filter to a new dataset of U.S. high school newspapers, augmented (via ZIP codes and counties) with demographic data from the U.S. Census and the National Center for Education Statistics (§5.3). We demonstrate that the filter has strong topical and stylistic preferences, and favors text from authors who originate from regions with better educational attainment, urban centers, larger schools, and higher valued homes.

In sociolinguistics, the term **language ideology** refers to common (but often unspoken) presuppositions, beliefs, or reflections about language that justify its social use and structure [Craft et al., 2020]. Our analysis begins to characterize the language ideology encoded in the quality filter used by Brown et al. [2020], a

[1] We note that the term *quality* is often ill-defined in the NLP literature. For example, Brown et al. [2020] and Wenzek et al. [2020] refer to "high-quality text" or "high-quality sources"—both citing Wikipedia as an example—but without explaining precisely what is meant.

representative of a wider set of filtering methods. We also observe in §2.5 that the filter is unaligned with other notions of quality familiar from human endeavors: factuality ratings for news sources, standardized test scores, and literary awards. Of course, these institutions hold their own language ideologies. We argue that when constructing a corpus, one cannot avoid adopting some language ideology; the language ideology which is appropriate depends on the goals of the work, and one language ideology may conflict with another. In short, there is no truly general-purpose corpus.

Our code and analysis is publicly available.[2] This chapter is based on the work from:

Suchin Gururangan, Dallas Card, Sarah Dreier, Emily Gade, Leroy Wang, Zeyu Wang, Luke Zettlemoyer, and Noah A. Smith. 2022. Whose Language Counts as High Quality? Measuring Language Ideologies in Text Data Selection. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 2562–2580, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

2.1 Archives as Subjective Agents

Traditional definitions of archives are often pragmatic in nature and disassociated from both the archiver and their subjects. For example, Decker [2013a] describes that archives are traditionally seen as “materials specifically collected for the purpose of preserving a history and housed in a distinct location".

Mills and Helms [2017], on the other hand, note that “knowledge is plural and relies heavily on underlying ontological and epistemological assumptions" of the archiver. Postmodern archival theory was generally embodied by a skepticism towards the objectivity of archives, instead emphasizing its incompleteness and constructed nature, governed by social dynamics and language [Ogden et al., 2017; Mills and Helms, 2017; Prasad, 2015]. The *archival turn*, as this skepticism has been referred to, denotes a shift viewing archives as merely objective records of history to an active agent that shapes and legitimizes knowledge [Stoler, 2009].

Much of this subjectivity materializes in the *silencing* of marginalized groups, by removing or manipulating their voices during history production. Harmful, or even violent [Derrida and Prenowitz, 1995], silencing is evident in colonial archives, which were used to drive political strategies and renew myths of colonizers [Stoler, 2009]. Trouillot [1995] details one such case in the narrativization of Christopher

[2] https://github.com/kernelmachine/quality-filter

Columbus' conquest of the Americas. Trouillot [1995] creates a comprehensive categorization of where archive silencing can be introduced: fact creation, fact assembly, fact retrieval, and narrativization. Trouillot's primary argument is that archives reproduce narrative histories that favor power asymmetries and silence the experience of minority groups.

In a wide analysis of business archive data collection, Decker [2013a] suggests that archival silencing further occurs in data collection, analysis, and mode of reporting. Archival silencing is not always consequence of conscious selection, but "a result of a haphazard process of disinterest, forgetting, and the loss of organization" [Decker, 2013a]. These issues suggest that knowledge expresses the interests and biases of its producer [Spivak, 2003].

2.2 Subjectivity in Web Archives

The post-colonial period lead to an some improvements in historical archives, to take into account the marginalization of voices that were written out of colonial histories [Robin and Gallagher, 1967]. The advent of the Internet was poised to improve the preservation of content from silenced voices even further.

Web archives were developed in the late 1990s in part to avoid a "digital dark age", in which technological development outpaces our ability to preserve its history [Kunny, 1997]. Kunny [1997] predicts the daunting challenge of archiving web documents in the face of rapidly changing technologies; for example, software and hardware may become out of date, the growth of digital history is exponential as humans gain more access to the Internet, and sociocultural phenomena change quickly in online communities.

Kunny was prescient about the factors that would make web archiving difficult today. To further understand these modern technical and ethical challenges to web archiving, Ogden et al. [2017] take an ethnographic approach to study the work of web archivists. Over a four week period in collaboration with web archivists, engineers and managers at the INTERNET ARCHIVE, the authors reveal previously undocumented activities in preserving the Web.

The major finding corroborated by this study is the critical duty that human intervention plays in shape the preservation and maintenance of web archives. For example, the authors describe the use of zone files, partner-submitted seed lists, links embedded in social media streams, and look-ups of previously archived content held by the Archive to identify domains for crawling. Other sources for selection include surveying

Twitter to determine which YouTube videos get linked to, as well as scouring Wikipedia for certain outlinks that get added. The authors note that domain selection is both a problem of resources, and also “popularity", “novelty", and the “risk of going away". In general, Ogden et al. [2017] state that at any given time, web crawling services are only capturing about 20% of the Web.

Human intervention and technical issues lead to web archives that are incapable of tracking the constantly evolving and intricate Internet. Rather, they are *subjective constructions* where, as Brügger [2011] states, "the process of archiving itself may change what is archived, thus creating something that is not necessarily identical to what was once online."

Further complicating the efforts of clean, transparent archiving is the lack of knowledge of how web archiving practices are actually performed. Ogden et al. [2017]'s ethnographic study of web archiving practices occured only at the Internet Archive. However, much of modern language technologies are built on Common Crawl, a private organization with limited transparency on their processes.[3] These limit the ability to make broad claims about generalizability of technology built on such data.

2.3 Data Sources for Language Models

Across the many language models recently reported in the literature, the same small group of datasets have been routinely used as training corpora—Wikipedia, collections of books, and popular online articles (Table 2.1). These data are often treated as exemplars of high quality text [Devlin et al., 2019a; Liu et al., 2019a; Radford et al., 2019a; Raffel et al., 2020; Brown et al., 2020]. Although these datasets include text from many sources, extensive research suggests that the voices they represent are drawn from a relatively small, biased sample of the population, over-representing authors from hegemonic social positions.

Wikipedia Wikipedia serves as a backbone for language models because of its scale, ease of use, permissive license, and goal of providing comprehensive coverage of human knowledge. However, although anyone can edit Wikipedia content, not everyone does. In practice, there are significant biases in Wikipedia authorship, content, and perspectives. For instance, despite efforts by Wikimedia, the site has been unable to resolve a persistent gender imbalance among its editors [Huang, 2013; Meta-wiki, 2018]. This imbalance is reflected in

[3] Indeed, the author could not find any documentation of Common Crawl's scraping practices.

Model	Pretraining Data Sources	Citation
ELMo	1B Word benchmark	[Peters et al., 2018]
GPT-1	BookCorpus	[Radford et al., 2018b]
GPT-2	WebText	[Radford et al., 2019a]
BERT	BookCorpus + Wikipedia	[Devlin et al., 2019a]
RoBERTa	BookCorpus + Wikipedia + CC-news + OpenWebText + Stories	[Liu et al., 2019a]
XL-Net	BookCorpus + Wikipedia + Giga5 + ClueWeb 2012-B + Common Crawl	[Yang et al., 2019]
ALBERT	BERT, RoBERTa, and XL-net's data sources	[Lan et al., 2020]
T5	Common Crawl (filtered)	[Raffel et al., 2020]
XLM-R	Common Crawl (filtered)	[Conneau et al., 2020]
BART	BookCorpus + Wikipedia	[Lewis et al., 2020]
GPT-3	Wikipedia + Books + WebText (expanded) + Common Crawl (filtered)	[Brown et al., 2020]
ELECTRA	BookCorpus + Wikipedia + Giga5 + ClueWeb 2012-B + Common Crawl	[Clark et al., 2020]
Megatron-Turing NLG	The Pile + Common Crawl (filtered) + RealNews + Stories	[Kharya and Alvi, 2021]
Switch-C	Common Crawl (filtered)	[Fedus et al., 2021]
Gopher	MassiveWeb + Books + Common Crawl (filtered) + News + GitHub + Wikipedia	[Rae et al., 2021a]

Table 2.1: Overview of recent language models and their training corpora. All studies tend to draw from the same core data sources: Wikipedia, Books, News, or filtered web dumps.

who gets written about, and how [Bamman and Smith, 2014; Graells-Garrido et al., 2015; Wagner et al., 2015]. There is also a pervasive urban bias; editors are less likely to come from rural areas, and coverage of these areas in Wikipedia tends to be more limited [Mandiberg, 2020]. Although coverage in English Wikipedia is not limited to those places where English is a majority language, an Anglo-American perspective dominates coverage.[4] Lastly, a relatively small number of people are responsible for most of the content [Panciera et al., 2009; Matei and Britt, 2017]. Wikipedia is thus less representative of language of the population than one might expect given its size and design.

Books Language models are also frequently trained on book corpora. BERT [Devlin et al., 2019a] used the Toronto BookCorpus [Zhu et al., 2015a], which consists of 7,185 self-published novels, a dataset criticized for copyright violation, poor quality control, imbalanced representation, and lack of documentation [Bandy and Vincent, 2021].

GPT-3 [Brown et al., 2020] and The Pile [Gao et al., 2020] both use much larger corpora of books (although the former do not identify the source of this data). However, the Pile's books (also called Books3) are not a random selection. Rather, they appear to be drawn from a torrent file containing hundreds of thousands of copyrighted eBooks.

[4]For example, of the ten most frequently mentioned people in English Wikipedia, seven are U.S. Presidents, two are prominent figures in Christianity, and the only woman is the British monarch, Queen Victoria.

URL Domain	# Docs	% of Total Docs
bbc.co.uk	116K	1.50%
theguardian.com	115K	1.50%
washingtonpost.com	89K	1.20%
nytimes.com	88K	1.10%
reuters.com	79K	1.10%
huffingtonpost.com	72K	0.96%
cnn.com	70K	0.93%
cbc.ca	67K	0.89%
dailymail.co.uk	58K	0.77%
go.com	48K	0.63%

Table 2.2: The most popular top-level URL domains in OpenWebText. Mainstream news forms the overwhelming majority of content in the dataset. Overall, just 1% of the top-level URL domains in OpenWebText contribute 75% of the total documents in the corpus.

Books3 is deserving of a more thorough investigation, but preliminary analyses reveal that the most prevalent authors in the corpus are American and British writers, especially of romance, mystery, and children's books (e.g., L. Ron Hubbard, Danielle Steel, etc.). This pattern should be considered against the background of the American book publishing industry, which has been widely criticized as homogeneous [Lee & Low Books, 2020].[5]

News and Other Popular Internet Content Radford et al. [2019a] scrape text from the websites featured in popular Reddit submissions (i.e., those that received at least three upvotes) to construct the training data for GPT-2. As the original corpus is unavailable, we analyze its open-source replica, OpenWebText [Gokaslan and Cohen, 2019a].

We do not expect the corpus to represent a wide range of language variation. Reddit users are mostly male, younger, and lean liberal, which influences the types of content shared on the platform.[6] Viral media on the Internet assume similar characteristics; they tend to elicit awe, anger, or anxiety [Berger and Milkman, 2012], validate group identities [Gaudette et al., 2021], and disseminate from users with authority [Weismueller et al., 2022].

Indeed, we find that 1% of the 311K unique top-level domains in OpenWebText contribute 75% of

[5]This 2020 study found that Black people comprise only 5% of the industry, and books by men tend to generate disproportionately more attention than those by women.

[6]As of 2016, 71% of Reddit users are male, 59% are between ages 18–29, and 43% identify as liberal (vs. 19% conservative): https://pewrsr.ch/3FLbNL7

documents in the corpus (Table 2.2). The most common websites in OpenWebText are internationally circulating British and American news outlets (e.g., *BBC*, *The New York Times*, *The Washington Post*, *The Guardian*), blogging platforms (e.g., *Tumblr*, *Blogspot*, or *Medium*), sports content (e.g., *ESPN*, *SBNation*), and tech news (e.g., *TechCrunch*, *Wired*). As expected, these links tend to appear on the most highly trafficked subreddits (e.g., */r/politics*, */r/worldnews*, */r/news*).

These data are likely dominated by formal writing styles. Among news organizations, the adherence to slowly evolving style guides expresses specific linguistic standards [Froke et al., 2020] and even geopolitical interests [Vultee, 2012], which encourage rules about language use that can reinforce gender norms and racial hierarchies [DiNicola, 1994; Bien-Aimé, 2016].

In general, a relatively homogeneous set of authors writes the majority of newswire [Grieco, 2018]. Researchers find a striking lack of diversity in newsrooms and newspaper leadership.[7] This may be compounded by the economic hardships aspiring journalists must incur,[8] which act as a filter for who can afford to be employed in newsrooms.

Summary Authors from specific, relatively powerful social positions produce a disproportionate amount of text in the core data sources of existing language models. These text sources favor privileged segments of the English-speaking population, including men, white populations, communities of higher socio-economic status, and those harboring American and Western European historical, geopolitical, and cultural perspectives. By contrast, these corpora tend to be less inclusive of the voices of women and members of marginalized groups. Alternative perspectives, including those of people from rural areas, non-dominant gender, sexual, or racial identities, and counter-hegemonic vantage points, are less likely to be included, and thus less likely to influence models trained on this data.

Although formal, streamlined content like news or Wikipedia articles may seem like desirable sources for high quality content, not all writing styles or substantive topics that might be relevant to language technologies and their user communities are represented in the resulting corpora. When deployed, many of the technologies using language models trained on these data will face language that—despite being less formal, professional,

[7] As of 2018, racial minorities make up 37% of the U.S. population, but only 17% of staff and 13% of leadership in U.S. newsrooms [Arana, 2018].

[8] In 2020, median salary for U.S. news analysis, reporters, and journalists was $35,950, a slight decrease from 2012 after adjusting for inflation: https://pewrsr.ch/3qCO75v

or carefully edited—is no less high quality and is essential to the communicative lives of the people who use it.

2.4 Measuring the Language Ideology of the GPT-3 Quality Filter

Empirically evaluating the full distribution of authors in the data sources from §2.3 is difficult, due to their size, as well as their lack of metadata about each document's authors. We instead curate a new dataset of U.S. high school newspaper articles that varies both topically and along demographic variables that can be resolved using ZIP codes. Although we do not directly consider individual authors of these articles, this dataset is useful, in that it can be associated with extensive metadata at the level of individual newspapers. We then analyze the behavior of a (replicated) quality filter on text from this dataset and discuss its implications.

2.4.1 U.S. School News

Background Many U.S. schools produce a newspaper to give students journalism experience, to report on local news, to comment on national or global events, and to publish school-related material (e.g., announcements, campus life, student interviews, sports or honor rolls; Gibson, 1961). The substantive content of school newspapers varies considerably, possibly due to their local audiences. Because a school's access to resources is shaped by local income levels [Betts et al., 2000] and tied to student achievement [Greenwald et al., 1996], we expect schools in wealthier areas (relative to poorer areas) to produce newspaper content that is more similar to the formal, professional texts that a quality filter is likely to classify as high quality.

Collection We collect articles from English-language U.S. school newspapers that used a common Wordpress template.[9] After retrieving 2483 schools who use this template, we scrape 1.95M articles from their respective newspaper sites. We retrieve article categories by extracting them from the article URL slugs. We then resolve each school to its ZIP code using the Google Maps Place API.[10] We restrict our dataset to articles from U.S. high schools. We only consider articles from 2010–2019, remove pages under the *video*, *photo*, or *multimedia* categories, and remove schools that have less than 100 articles (which tend to contain

[9] SNOsites.com

[10] https://developers.google.com/maps/documentation/places/web-service/search-find-place?hl=en

scraping errors). The final corpus includes 910K articles, from 1410 schools, located in 1329 ZIP codes (552 counties) dispersed across all U.S. states (plus D.C.).

Limitations Our corpus is neither a random nor a representative sample of U.S. school newspapers. Instead, it represents schools that had sufficient Internet access, that elected to use a particular website template, and that maintain websites with retrievable archived content. The lack of representation in school newspaper leadership positions may influence which students contribute content to school newspapers [Chen et al., 2021a]. Educators also likely shape some articles, at least in part (though we expect them to be similarly affected by resource constraints). Finally, much of the content in these articles is specific to student concerns (e.g., sports, school events, campus culture, etc.), and the writing is, by definition, amateur. Nevertheless, because the corpus captures a wide range of content and geographical areas, it allows us to evaluate how a quality filter handles real-world language variation, within a particular domain.

Using text from school newspapers introduces privacy concerns, especially since authors and subjects are minors. We therefore use this data only for evaluation purposes, and do **not** train (or release) any models on this data, or any raw text from the corpus. We do, however, release a Datasheet [Gebru et al., 2021] which documents the dataset's general characteristics and curation procedure, which can be found here: `https://bit.ly/3rLrmwV`.

2.4.2 The GPT-3 Quality Filter

To investigate how quality correlates with various attributes of a newspaper, we re-implement the Brown et al., 2020 quality filter based on the description provided in the paper. The filter is a binary logistic regression classifier trained (using n-gram features) to distinguish between reference corpora (Books3, Wikipedia, and OpenWebText) and a random sample of Common Crawl.

We replicate the filter as closely as possible using `scikit-learn` [Pedregosa et al., 2011]. To create the training data for the classifier, we sample 80M whitespace-separated tokens of OpenWebText, Wikipedia, and Books3 each for the positive class, and 240M whitespace-separated tokens of a September 2019 Common Crawl snapshot for the negative class. We download the Common Crawl snapshot using code provided by Wenzek et al. [2020]. We perform a 100-trial random hyperparameter search, fixing only the hashing vectorizer and basic whitespace tokenization, following the implementation in Brown et al. [2020]. See the

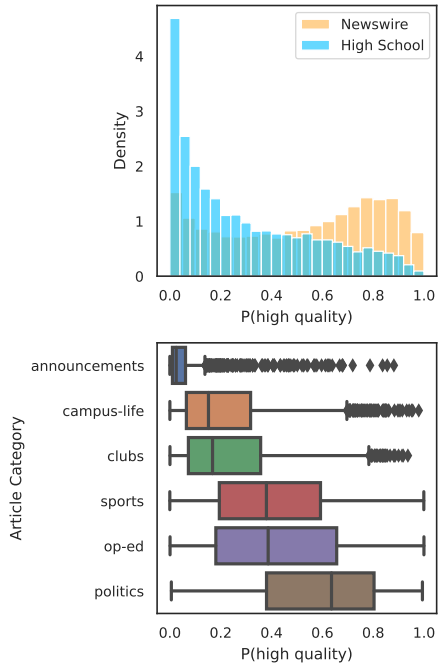

Figure 2.1: Scraped school articles tend to be considered lower quality by the GPT-3 quality filter than general newswire (histogram built from 10K random documents from each domain). This finding is consistent across a variety of categories, and more significant for certain ones (e.g., school announcements).

search space and final hyperparameters of our replicated filter in §A.1.1. Our final classifier gets 90.4% F_1 (91.7% accuracy) on a set of 60M test tokens (30M held-out tokens from each class, or 72K documents from the negative class, and 33K from the positive class). We release code for training the quality filter and a demo of the trained filter.[11,12] We apply the quality filter to the U.S. SCHOOL NEWS data, computing a quality score per document, which we denote $P(\text{high quality})$.

2.4.3 Document-Level Analysis

We first explore document-level preferences of the filter. The GPT-3 quality filter is more likely to classify high school newspaper articles as low quality, compared to general newswire (Figure 2.1).[13] This is unsurprising, since the training data for the GPT-3 quality filter included texts by professional journalists. Table 2.3 shows a random sample of text from the dataset with high and low quality scores, illustrating differences in style and formality.

[11] https://github.com/kernelmachine/quality-filter
[12] https://huggingface.co/spaces/ssgrn/gpt3-quality-filter
[13] Here, the general newswire are articles from popular online news sources; see §2.5 for data details.

Category: Student-Life P**(high quality) = 0.001**
As our seniors count down their final days until graduation, we will be featuring them each day. [REDACTED], what are your plans after graduation? To attend [REDACTED] in the fall and get my basics. Then attend the [REDACTED] program. What is your favorite high school memory? My crazy, obnoxious and silly 5th hour English with [REDACTED]. What advice do you have for underclassmen? Pay attention, stay awake (I suggest lots of coffee), and turn in your dang work! You can do it, keep your head up because you are almost there!
Category: News P**(high quality) = 0.99**
On Monday, September 3rd, Colin Kaepernick, the American football star who started the "take a knee" national anthem protest against police brutality and racial inequality, was named the new face of Nike's "Just Do It" 30th-anniversary campaign. Shortly after, social media exploded with both positive and negative feedback from people all over the United States. As football season ramps back up, this advertisement and the message behind it keeps the NFL Anthem kneeling protest in the spotlight.

Table 2.3: Examples of high school news paper articles from U.S. SCHOOL NEWS. Many of the articles in student-life category, and similar, rated lower quality have very different styles from documents rated high quality.

More notably, controlling for article category (e.g., opinion pieces), we find that the GPT-3 quality filter has topical and stylistic preferences (discovered through exploratory data analysis). For topical features, we train an topic model (via Latent Dirichlet Allocation; Blei et al. 2003) over opinion pieces with 10 topics using `scikit-learn`. We also consider whether documents contain first, second, or third person pronouns, and the length of the document. We then combine these features in a regression model to assess the effect of particular attributes on the quality score of a document, while controlling for others.

The results of our regression are displayed in Table 2.4. We find that certain topics have quite large effect sizes (see Figure 2.2 for the distribution of quality scores per topic). For example, documents entirely about Trump and the presidential election have quality scores 35 percentage points higher, on average, whereas documents about sports are 25 percentage points higher (relative to the omitted topic about food). Stylistically, the presence of first or second pronouns in a document decreases quality score by 5 percentage points, while

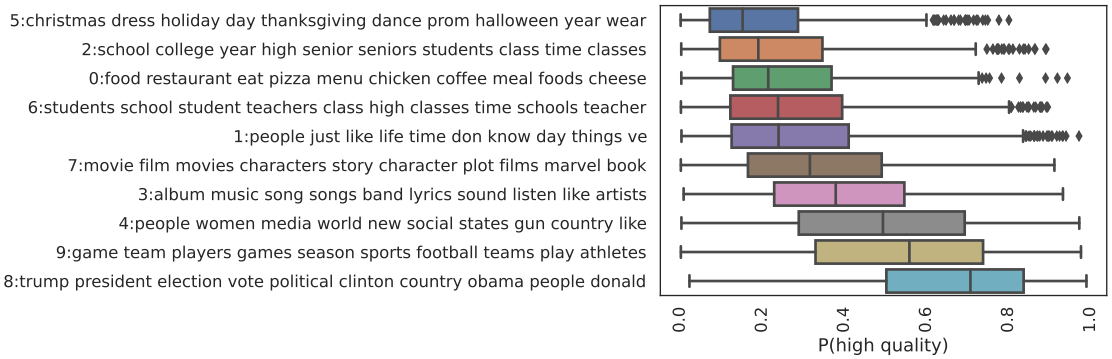

Figure 2.2: The GPT-3 quality filter prefers topics that are more prevalent in Wikipedia or newswire. Here, we consider 10K opinion pieces in U.S. SCHOOL NEWS.

Dependent variable: $P(\text{high quality})$
Number of observations: 10K opinion articles

Feature	**Coefficient**
Intercept	0.471^{***}
Topic 5 (*christmas, dress, holiday*)	-0.056^{***}
Topic 2 (*school, college, year*)	-0.037^{***}
Topic 6 (*student, school, class*)	-0.004
Topic 1 (*people, just, like*)	0.003
Topic 7 (*movie, film, movies*)	0.062^{***}
Topic 3 (*music, album, song*)	0.113^{***}
Topic 4 (*people, women, media*)	0.197^{***}
Topic 9 (*game, team, players*)	0.246^{***}
Topic 8 (*Trump, president, election*)	0.346^{***}
Presence of first/second person pronoun	-0.054^{***}
Presence of third person pronoun	0.024
$\log_2$(Number of tokens)	0.088^{***}
R^2	0.336
adj. R^2	0.336

Table 2.4: Regression of the quality score of an opinion piece in the U.S. SCHOOL NEWS dataset, on document features. We observe that political and sports-related topics, the lack of first and second person pronouns, and longer document lengths are associated with higher quality scores. We omit Topic 0 (*food*, *restaurant*, *eat*) to avoid a saturated model. $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

a doubling of the number of tokens in a document increases the quality score by 9 percentage points.

2.4.4 Demographic Analysis

Next, we examine whether the GPT-3 quality filter prefers language from certain demographic groups over others. We first check raw correlations between average quality scores (per newspaper) and features of

Feature	Description	Level	Source
Is Charter	Is the school a charter school?	School	NCES database
Is Private	Is the school a private school?	School	NCES database
Is Magnet	Is the school a magnet school?	School	NCES database
% Black Students	% students who identify as Black	School	NCES database
% Asian Students	% students who identify as Asian	School	NCES database
% Mixed Students	% students who identify as Mixed race	School	NCES database
% Hispanic Students	% students who identify as Hispanic	School	NCES database
Student:Teacher	Student-teacher ratio	School	NCES database
School Size	Total number of students	School	NCES database
Median Home Value	Median home value	ZIP code	Census
% Adults $\geq$ Bachelor Deg.	% adults ($\geq$ 25 years old) with at least a bachelor's degree	ZIP code	Census
% Rural	Percent of a county population living in a rural area	County	Census
% 2016 GOP Vote	Republican vote share in the 2016 presidential election	County	MIT Election Lab

Table 2.5: Description of features we include in our demographic analyses. These features were drawn from the National Center of Education Statistics database, the Census, or the MIT Election Lab data repository.

interest. As in §2.4.3, we then combine the features in a regression model.

Demographic Features As we note in §5.2.1, we expect *a priori* that content from schools located in wealthier, more educated, and urban areas of the U.S. will tend to have higher quality scores, relative to poorer, less educated, rural areas. Therefore, we consider demographic features that correspond to class, rural/urban divides, and school resources.

For each school, we retrieve 2017–2018 school-level demographic data from the National Center for Education Statistics (NCES).[14] These include the number of students, student:teacher ratio, and indicators for charter, private, and magnet schools. We also retrieve the latest ZIP code- and county-level demographic data from the 2020 U.S. Census.[15] To measure the wealth of the corresponding ZIP code, we use median home values, and for educational attainment we use the percentage of college-educated adults. We also use Census data on the percent of rural population by county. Finally, we consider local political leanings, operationalized by GOP vote share in the 2016 Presidential election, using county-level data from the MIT election lab.[16] We display full descriptions of features in our demographic analysis in Table 2.5.

[14] https://nces.ed.gov/ccd/elsi/tablegenerator.aspx
[15] https://data.census.gov/cedsci/
[16] https://electionlab.mit.edu/data

Correlation Analysis To inform the variables we include in our regressions, we explore correlations between variables of interest and the average quality score of a school newspaper. Our analyses in Figure 2.3 suggest that our initial hypotheses hold: schools in wealthier, urban, and more educated ZIP codes, as well as those in Democrat-leaning counties, tend to have higher quality scores.

Data Preprocessing Here, we use schools as the unit of analysis, and consider average quality score assigned to the school's articles as the dependent variable. We only include those schools that could be matched to the NCES database, dropping schools which are missing school size, as well as those located in ZIP codes with $1M or greater median home value, due to a census artifact.[17] Missing values for other features are imputed with the median value of that feature for the corresponding ZIP code, or (if necessary) county or state. For regressions, we log-transform school size, student:teacher ratio, and home values, using raw values for other features, to preserve interpretability. Our regression dataset includes 968 high schools, in 926 ZIP codes across 354 counties. All linear regressions are implemented with the `statsmodels` API.[18] We release this anonymous dataset to support reproducibility.[19]

Regression Analysis Because the variables identified above are correlated with each other, we use regression to estimate the effect of certain factors while controlling for others, with results shown in Table 2.6. Overall, home values, parental education, school size, public school status, and urban locations all show significant positive associations with quality scores. Thus, even controlling for financial resources, parental education, and other factors, articles from rural schools are still scored as significantly lower quality than those from urban schools.

Nevertheless, the effects, considered individually, are relatively modest. A 14 percentage point increase in percent urban population or a 17 percentage point increase in parental education (percent of adults with college degrees) correspond to a 1 percentage point increase in average quality score, as does a doubling of home values, or a quadrupling of school size. Average quality scores associated with public schools are 1.5 percentage points higher than private schools, controlling for other factors. Coefficients for charter schools, magnet schools, and student:teacher ratio are all sensible, though none are significant. Altogether,

[17] The census data we use imposes an artificial upper bound on housing prices over $1M.
[18] https://www.statsmodels.org/stable/index.html
[19] https://github.com/kernelmachine/quality-filter

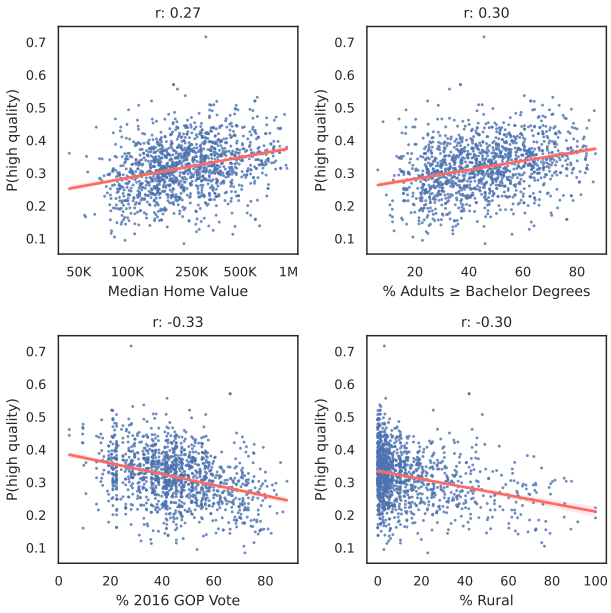

Figure 2.3: Correlations of demographic features of a school's ZIP code or county with its average P(high quality). These results suggest that quality filter scores are positively correlated with median home value and educational attainment.

Dependent variable: P(high quality)
Observations: 968 schools

Feature	**Coefficient**
Intercept	0.076
% Rural	-0.069^{***}
% Adults $\geq$ Bachelor Deg.	0.059^{**}
$\log_2$(Median Home Value)	0.010^{*}
$\log_2$(Number of students)	0.006^{*}
$\log_2$(Student:Teacher ratio)	-0.007
Is Public	0.015^{*}
Is Magnet	0.013
Is Charter	0.033
R^2	0.140
adj. R^2	0.133

Table 2.6: Regression of the average P(high quality) of a school in the U.S. SCHOOL NEWS dataset, on demographic variables. We observe that larger schools in educated, urban, and wealthy areas of the U.S tend to be scored higher by the GPT-3 quality filter. $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

the combined effects of all these factors account for large differences in quality scores between wealthy, urban, educated locations, and poorer, rural, and less educated parts of the country.

Summary and Limitations This analysis reveals an unintended consequence of the GPT-3 quality filter: by attempting to exclude text that is less like mainstream news and Wikipedia, the filter reinforces a language ideology that text from authors of wealthy, urban, and educated backgrounds are more valuable for inclusion in language model training data. These implicit preferences align with the attributes of authors that dominate the corpora from §2.3, which the filter considers to be high quality.

Although most of the above findings are robust to alternate model specifications, the model ultimately only accounts for a relatively small amount of variance in quality scores. In addition, most of our features are taken from a single a point in time, and do not account for changing demographics over the period 2010–2019. Data errors could also arise due to how datasets were aligned (based on school name and ZIP code). These findings may not generalize to other domains (e.g., social media), and inclusion of additional features could affect these findings.

2.4.5 Additional Regressions

Here we include regressions results from two models with additional covariates.

We first consider race as a possible omitted variable, given the extent of school segregation in the U.S. [Reardon and Owens, 2014]. NCES data provides the distribution of students by race for each school, using a particular set of racial categories, which comes with obvious limitations. Nevertheless, we use the raw percentage scores provided as additional covariates in this model as a validity check. We exclude the Native and Pacific Islander categories, due to imbalanced data and geographic concentration, as well as the white category, to avoid a saturated model.

As shown in Table 2.7, the findings are nearly identical to the results in the main paper, with the exception that home values are no longer significant. The only racial category that shows a significant effect is Asian. However, we note a positive correlation between percentage of Asian students and median home values (Pearson r =0.32, $p < 0.001$), suggesting that the variable for percentage of Asian students may be partially absorbing the effect of our measure of wealth.

Table 2.8 shows the results for an alternate model which includes % GOP vote share in the 2016 election. Once again, the results are very similar to the results in the main paper, although there is a strong (and significant) negative association between GOP vote share and quality scores, whereas the measures of home

Dependent variable: $P(\text{high quality})$
Observations: 968 schools

Feature	**Coefficient**
Intercept	0.134
% Rural	-0.073^{***}
% Adults $\geq$ Bachelor Deg.	0.049^{*}
$\log_2$(Median Home Value)	0.007
$\log_2$(Number of students)	0.005^{*}
$\log_2$(Student:Teacher ratio)	-0.008
Is Public	0.020^{*}
Is Magnet	0.013
Is Charter	0.035^{*}
% Asian Students	0.081^{**}
% Mixed Students	0.051
% Black Students	-0.009
% Hispanic Students	-0.020
R^2	0.152
adj. R^2	0.142

Table 2.7: Regression of the average $P(\text{high quality})$ of a school in the U.S. SCHOOL NEWS dataset, on demographic variables. As in the main paper, larger schools in educated and urban areas of the U.S tend to be scored higher by the GPT-3 quality filter. Asian is the only categorical race variable which shows a significant association (using data and categories taken directly from NCES). The association with home values is no longer significant, plausibly explained by a correlation between a higher proportion of Asian students and higher median home values. $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

values and percent rural are no longer significant.

The results for this model exemplify the difficulty of working with highly correlated variables. Given the strong association between GOP voters and rural areas, GOP vote share serves as an effective proxy for other variables of interest. However, because the results of the 2016 Presidential election were likely somewhat idiosyncratic, and because we find wealth and geography to be a more plausible explanation for differences in student writing than political preferences among their parents, we opt for the model without GOP vote share in the main paper.

2.5 Alignment with Other Notions of Quality

The GPT-3 quality filter purports to judge the quality of text. Humans, on the other hand, frequently judge the quality of text without the use of automated systems. In this section, we consider three forms of human evaluations: institutional awards to select books, fact-checkers' designated factuality of news outlets, and

Dependent variable: $P(\text{high quality})$
Observations: 968 schools

Feature	Coefficient
Intercept	0.248^{**}
% Rural	-0.021
% Adults $\geq$ Bachelor Deg.	0.067^{**}
$\log_2$(Median Home Value)	0.003
$\log_2$(Number of students)	0.006^{**}
$\log_2$(Student:Teacher ratio)	-0.007
Is Public	0.017^{*}
Is Magnet	0.009
Is Charter	0.027
% GOP vote share	-0.114^{***}
R^2	0.164
adj. R^2	0.157

Table 2.8: Regression of the average $P(\text{high quality})$ of a school in the U.S. SCHOOL NEWS dataset, on demographic variables, including % 2016 GOP Vote. We observe that including the political leaning of the county tends to wash out other variables, likely because partisan voting correlates heavily with other effects, like the urban/rural divide [Scala and Johnson, 2017]. The only other covariates that stay significant are school size, parental education, and public (as opposed to private) schools. $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

standardized test essays evaluated by human graders. How well does the behavior of the GPT-3 quality filter map onto these other notions of quality?

2.5.1 Data

Factually (Un)reliable News To analyze the correspondence between the GPT-3 quality filter and news factuality, we use the list provided by Baly et al. [2018] to identify a set of popular news sources from a broad range of factuality ratings and political leanings.[20] Using `Newspaper3k`,[21] we scrape and score 9.9K and 7.7K articles from high and low factuality news outlets, respectively.

Essay Exams Next, to analyze the correspondence between the GPT-3 quality filter and essay scores, we collect and score 12.1K participant essays from the *Test Of English as a Foreign Language* (TOEFL) exam, a widely used English language proficiency test [Blanchard et al., 2013]. The TOEFL exam responses include official scores from exam readers, as well as each essay's prompt.

[20] Baly et al. [2018] release a dataset of factual reliability and political leanings across news sources by scraping `NewsMediaBiasFactCheck.org`.

[21] `https://newspaper.readthedocs.io/en/latest/`

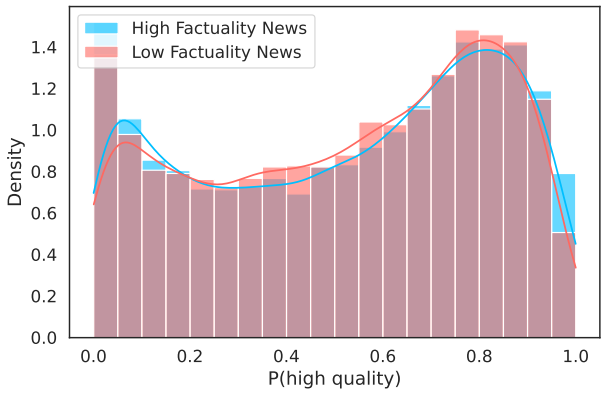

Figure 2.4: Distribution of quality scores for high and low factuality news sources. There is no difference in quality scores between articles written by news sources of high and low factual reliability.

Award-Winning Literature Finally, to analyze the correspondence between the GPT-3 quality filter and literary awards, we select and score books from Books3 and the Gutenberg corpus [Brooke et al., 2015] that have won a Pulitzer Prize in various categories. We collected these data by scraping the publicly available list of recipients.[22]

2.5.2 Results

If the filter aligns with news factuality, we would expect that articles from factually reliable sources would be rated as higher quality than those from factually unreliable ones. However, we find no difference in the quality distribution between articles from high and low factuality news sources ($p = 0.085$, two-way Kolmogorov-Smirnov test; Figure 2.4).

Turning to the TOEFL exam responses, we would expect that if the filter agrees with essay scores, higher scoring essays would receive higher quality scores. While essay scores are weakly correlated with quality scores (Pearson $r = 0.12$, $p < 0.001$), Table 2.9 demonstrates that the essay's prompt is far more predictive of the essay's quality designation. For example, essays responding to a prompt (*P4*) which asks participants to describe *"...whether advertisements make products seem much better than they really are"* are much less likely to be filtered than all other prompts, including *P6*, which asks participants to describe *"...whether it is best to travel in a group"*. The latter prompt tends to invoke personal experiences in the responses.

Finally, if the filter aligns with literary awards, we would expect that most Pulitzer-Prize winning books

[22] https://www.pulitzer.org/prize-winners-categories

Dependent variable: $P(\text{high quality})$
Observations: 12.1K TOEFL exams

Feature	Coefficient
Intercept	0.0631^{***}
Low score	-0.0414
High score	0.0339
Prompt 7	-0.0283^{***}
Prompt 6	-0.0204^{***}
Prompt 2	0.0068^{***}
Prompt 8	0.0346^{***}
Prompt 3	0.0880^{***}
Prompt 5	0.1470^{***}
Prompt 4	0.6745^{***}
R^2	0.712
adj. R^2	0.711

Table 2.9: Regression of the quality of a TOEFL exam essay on its assigned score and prompt. While we observe some relationship between the score an essay receives and its quality score, the essay prompts themselves have significantly higher effect sizes. The highest quality essays come from Prompt 4, which asks participants to discuss products and advertisements. $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

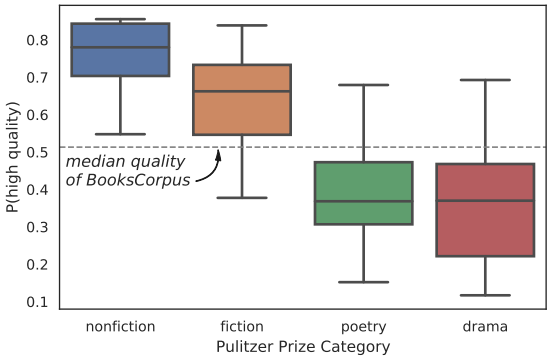

Figure 2.5: Quality scores for different genres of books that have won a Pulitzer Prize. Among works that have won a Pulitzer Prize, the quality filter tends to favor nonfiction and longer fictional forms, disfavoring poetry and dramatic plays.

would achieve high quality scores. On the contrary, quality scores vary heavily based on the genre (Figure 2.5). Poetry and drama are less favored by the filter relative to non-fiction, fiction, and even fan fiction (from the BookCorpus; Zhu et al. 2015a).

Summary Our analysis demonstrates that the GPT-3 quality filter conflicts with other standards of text quality. Of course, even the alternative standards we compare here are subject to their own language ideologies. Readers are more likely to trust news as factual if its political position aligns with their own [Mitchell et al.,

2018]. English-language teaching pedagogies are rooted in ideologies about well-spokenness [Vanegas et al., 2016]. Literary awards favor white and male authors.[23] In general, any designation of text as high quality is subjective and influenced by sociopolitical context.

2.6 Conclusion

The above sections have demonstrated that automated filtering of text to build language modeling corpora may lead to counterintuitive or undesirable exclusion of sources. Because of the variety of use cases for language models and the broad range of text that could be appropriate for certain tasks, we suggest that there is no simple, universal standard for what should be considered high quality text. Indeed, there is a long history of privileging some people's spoken language as better or more "correct" than others. Researchers and practitioners of NLP who are aware of this history have the option to be intentional in their design of systems that, however implicitly, risk excluding the language of underprivileged identities or communities.

Some amount of selection in building corpora is inevitable. It is not possible to collect a uniform random sample of all written utterances. However, our findings suggest that current selection methods are, for many purposes, flawed. Future work into alternative filtering criteria could be paired with investigations into the unintended consequences of their assumptions.

We do not believe that there is likely to be a single solution to this challenge. Indeed, the text that is best suited for training a model may depend on the application of that model. At a minimum, however, the NLP community could more carefully consider and clearly document the criteria by which text is being selected for inclusion. NLP practitioners could also be explicit about the reasons for using certain sources, even if those reasons are related to availability or empirical performance. A collection of tests could also be deployed (and improved over time), to give a clear understanding of the implications of different choices of filters.

More generally, we echo calls in the literature for more thoughtful and inclusive data collection [Jo and Gebru, 2020; Bender et al., 2021a; Tanweer et al., 2021]. This could include, but is not limited to a) intentionally curating data from people and viewpoints that are not otherwise well represented; b) including a greater diversity of genres; c) more nuanced or intentional exclusion criteria; d) more thorough interrogation

[23] A 2016 study by the Columbia Journalism Review found that since 1918, 84% of Pulitzer Prizes had been awarded to white authors, and 84% to male authors: https://www.cjr.org/analysis/100_years_of_data.php.

of what text is being excluded; e) developing standard checks for prominent biases in inclusion; f) abandoning the notion of a general-purpose corpus.

2.7 Limitations

Our work assumes the intention of data curators and model designers when curating the data, which is of course not completely clear. As such, we cannot be entirely sure that the outcomes of quality filters are *unintended* consequences. We hope that future language modeling studies provide criteria for the inclusion and exclusion of certain text in the pretraining data, which will allow us to make more justified assumptions about the underlying intentions around the dataset.

Our U.S. SCHOOL NEWS dataset comes with many limitations, as described in §5.2.1. For example, the dataset contains sampling biases (e.g., it depends on use of a specific publication template), and the ZIP codes and counties are not uniformly spread across U.S. states. In general, our dataset likely captures neither the least resourced schools (which may not have access to online resources) in the United States, nor the wealthiest ones (who may have their own publication platforms). However, we speculate that an expanded corpus, which included writings from these schools, would demonstrate a continuation of trends we report in this paper.

While the text in our dataset varies considerably along topical, stylistic, and demographic variables, it is a niche domain; the text is a specific genre meant for local student consumption, its authors are U.S. students, and it thus primarily represents U.S.-centric cultural and political perspectives. We acknowledge that we also perpetuate some of the biases we identify, especially by working with English language text from the United States. We hope future work will extend this study of language ideologies to multilingual settings, other textual domains, and different sets of authors.

With respect to demographic variables, we merge census demographics with school-level data via ZIP codes or counties, which are imperfect identifiers of a school, since ZIP codes (and counties) may include multiple schools of varying resource levels. Moreover, tracking demographic variables and other author metadata, if deployed at scale, implies a certain level of invasive surveillance [Brayne, 2017]. Future work may explore how to maintain the rights of authors as data subjects and producers while mapping demographic representation in large corpora.

Finally, we did not seek consent from authors to scrape their articles. The ethical and legal norms around scraping public-facing web data, especially those produced by minors, are still in flux [Fiesler et al., 2020], and may not align with user perceptions of what constitutes fair use of online communications [Williams et al., 2017]. For these reasons, we do not release the corpus of school newspaper articles, and only use it for analysis and evaluation. We only make available a dataset of demographic variables and quality scores per school, to support reproducibility.

2.8 Related Work

Language Ideologies Language ideologies have been widely explored in the sociolinguistics literature [Gal and Irvine, 1995; Rosa and Flores, 2017; Craft et al., 2020, *inter alia*]. An ideology that promotes the inherent correctness, clarity, and objectivity of certain language varieties over others is a mechanism for linguistic discrimination [Craft et al., 2020; Gal, 2016; MacSwan, 2020; Rickford and King, 2016]. A salient example of such discrimination is the stigmatization of second-language speakers of English [Lindemann, 2005].

Language ideologies have an important, but often unacknowledged, influence on the development of NLP technologies [Blodgett et al., 2020]. For example, an ideology that distinguishes between *standard* and *non-standard* language variations surfaces in text normalization tasks [van der Goot et al., 2021], which tend to strip documents of pragmatic nuance [Baldwin and Chai, 2011] and social signals [Nguyen et al., 2021]. Language on the Internet has been historically treated as a noisy variant of English, even though lexical variation on the Internet is highly communicative of social signals [Eisenstein, 2013], and varies considerably along demographic variables [Eisenstein et al., 2014a] and community membership [Lucy and Bamman, 2021b]. Language ideologies also surface in tools for toxicity detection; for example, the classification behavior of the PERSPECTIVE API (a popular hate speech detector) aligns with the attitudes of conservative, white, female annotators, who tend to perceive African-American dialects as more toxic [Sap et al., 2021]. In this work, we examine the language ideology encoded in a widely used quality filter for text data selection.

Critiques of *Laissez-Faire* Data Collection We provide empirical evidence that *laissez-faire* data collection (i.e., filtering large web data sources) leads to data homogeneity [Bender et al., 2021a]. As an alternative to

laissez-faire collection, Jo and Gebru [2020] recommend drawing on institutional archival practices. However, we note that language ideologies are also prevalent (and may not be explicit) in institutional archives, which, for example, have preferred colonial voices over colonized ones when documenting historical events [Trouillot, 1995; Decker, 2013b].

Other Quality Filters Other definitions of text quality are used to create pretraining datasets, some of which do not rely on the datasets from §2.3. However, all techniques adopt language ideologies of what constitutes high quality text. *Bad-word* filtering, which removes documents that contain certain stop-words, disproportionately excludes language about and by minority groups [Dodge et al., 2021]. Filtering Internet content for popularity [Radford et al., 2019a] leads to data homogeneity based on the characteristics of viral media and the composition of userbases in online forums (§2.3). Even lightweight filters [Aghajanyan et al., 2021a; Rae et al., 2021a] put more emphasis on features like document length over factuality when determining what makes a document high quality. Any filtering method requires transparent justification and recognition of tradeoffs.

Downstream Behavior The behavior of language systems aligns with what we would expect from a language ideology that favors training data written by a narrow, powerful sector of society. For example, dialogue agents perform significantly worse when engaging in conversations about race [Schlesinger et al., 2018] and with minority dialects of English [Mengesha et al., 2021]. GPT-3 frequently resorts to use of stereotypes when minority groups are mentioned in its prompt [Abid et al., 2021; Blodgett, 2021]. GPT-3 is also prone to producing hate speech [Gehman et al., 2020b] and misinformation [McGuffie and Newhouse, 2020], which we would expect if its quality filter fails to distinguish the factual reliability of news sources in its training data (§2.5). Concurrent to this work, Gao [2021] show that aggressive data filtering with the GPT-3 quality filter degrades downstream task performance. A closer analysis of how the language ideologies in data selection lead to certain model behaviors is a rich area for future work.

2.9 Conclusion

Using a new dataset of U.S. school newspapers, we find that the conventional, automated valuation of Wikipedia, newswire, BOOKCORPUS, and popular Internet content as reference for high quality text implicitly favors content written by authors from larger schools in wealthier, educated, urban areas of the United States. Adopting this language ideology for text data selection leads to implicit, yet systematic and as-yet undocumented inequalities in terms of whose language is more likely to be included in training corpora. Although no single action will solve this complicated issue, data curators and researchers could be more intentional about curating text from underrepresented authors and groups, gathering sources from multiple genres and writing styles, and documenting their curation procedures and possible sources of exclusion.

Chapter 3

Adapting Language Models with Continued Pretraining

The absence of a general purpose corpus implies that to make language models that are broadly useful in the community, we must develop techniques to *adapt* them to downstream use cases after the original pretraining phase. In this chapter, we present a simple yet effective technique to adapt language models through *continued pretraining*. While some studies have shown the benefit of continued pretraining on domain-specific unlabeled data [e.g., Lee et al., 2019], these studies only consider a single domain at a time and use a language model that is pretrained on a smaller and less diverse corpus than the most recent language models. Moreover, it is not known how the benefit of continued pretraining may vary with factors like the amount of available labeled task data, or the proximity of the target domain to the original pretraining corpus (see Figure 3.1).

We address whether continued pretraining on target domains is beneficial for one high-performing model, RoBERTa [Liu et al., 2019b]. We consider four domains (biomedical and computer science publications, news, and reviews; §3.2) and eight classification tasks (two in each domain). For targets that are not already in-domain for RoBERTa, our experiments show that continued pretraining on the domain (which we refer to as *domain-adaptive pretraining* or **DAPT**) consistently improves performance on tasks from the target domain, in both high- and low-resource settings.

Though domains could be defined around provenance labels (e.g., genres or forums), it is also possible to

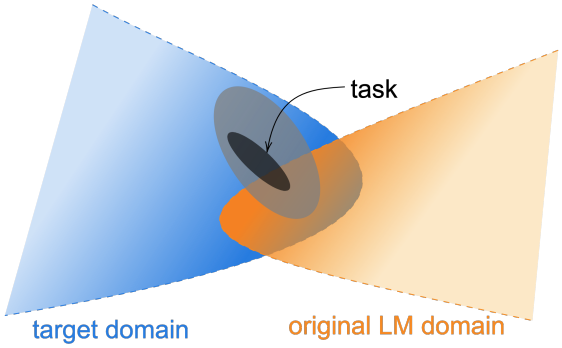

Figure 3.1: An illustration of data distributions. Task data is comprised of an observable task distribution, usually non-randomly sampled from a wider distribution (light grey ellipsis) within an even larger target domain, which is not necessarily one of the domains included in the original LM pretraining domain – though overlap is possible. We explore the benefits of continued pretraining on data from the task distribution and the domain distribution.

induce a domain from a given corpus used for a task, such as the one used in supervised training of a model. This raises the question of whether pretraining on a corpus more directly tied to the *task* can further improve performance. We study how domain-adaptive pretraining compares to *task-adaptive pretraining*, or **TAPT**, on a smaller but directly task-relevant corpus: the unlabeled task dataset (§3.3), drawn from the *task distribution*. Task-adaptive pretraining has been shown effective Howard and Ruder [2018], but is not typically used with the most recent models. We find that TAPT provides a large performance boost for RoBERTa, with or without domain-adaptive pretraining.

Finally, we show that the benefits from task-adaptive pretraining increase when we have additional unlabeled data from the task distribution that has been *manually curated* by task designers or annotators. Inspired by this success, we propose ways to automatically select additional task-relevant unlabeled text, and show how this improves performance in certain low-resource cases (§3.4). On all tasks, our results using adaptive pretraining techniques are competitive with the state of the art.

In summary, our contributions include:

- A thorough analysis of domain- and task-adaptive pretraining across four domains and eight tasks, spanning low- and high-resource settings;
- An investigation into the transferability of adapted LMs across domains and tasks; and

Domain	Pretraining Corpus	# Tokens	Size	$\mathcal{L}_{\text{ROB.}}$	$\mathcal{L}_{\text{DAPT}}$
BIOMED	2.68M full-text papers from S2ORC [Lo et al., 2020b]	7.55B	47GB	1.32	0.99
CS	2.22M full-text papers from S2ORC [Lo et al., 2020b]	8.10B	48GB	1.63	1.34
NEWS	11.90M articles from REALNEWS Zellers et al. [2019a]	6.66B	39GB	1.08	1.16
REVIEWS	24.75M AMAZON reviews [He and McAuley, 2016]	2.11B	11GB	2.10	1.93
ROBERTA (baseline)	see §3.1	N/A	160GB	‡1.19	-

Table 3.1: List of the domain-specific unlabeled datasets. In columns 5 and 6, we report ROBERTA's masked LM loss on 50K randomly sampled held-out documents from each domain before ($\mathcal{L}_{\text{ROB.}}$) and after ($\mathcal{L}_{\text{DAPT}}$) DAPT (lower implies a better fit on the sample). ‡ indicates that the masked LM loss is estimated on data sampled from sources *similar* to ROBERTA's pretraining corpus.

- A study highlighting the importance of pretraining on human-curated datasets, and a simple data selection strategy to automatically approach this performance.

Our code as well as pretrained models for multiple domains and tasks are publicly available.[1] This chapter is based on the work from:

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8342–8360, Online. Association for Computational Linguistics.

3.1 Background: Pretraining

Learning for most NLP research systems since 2018 consists of training in two stages. First, a neural language model (LM), often with millions of parameters, is trained on large unlabeled corpora. The word (or wordpiece; Wu et al. 2016) representations learned in the *pretrained* model are then reused in supervised training for a downstream task, with optional updates (*fine-tuning*) of the representations and network from the first stage.

One such pretrained LM is ROBERTA Liu et al. [2019b], which uses the same transformer-based architecture Vaswani et al. [2017] as its predecessor, BERT Devlin et al. [2019b]. It is trained with a masked

[1] https://github.com/allenai/dont-stop-pretraining

language modeling objective (i.e., cross-entropy loss on predicting randomly masked tokens). The unlabeled pretraining corpus for RoBERTa contains over 160 GB of uncompressed raw text from different English-language corpora. RoBERTa was trained on data from BookCorpus Zhu et al. [2015b],[1] Wikipedia,[2] a portion of the CCNews dataset Nagel [2016],[3] OpenWebText corpus of Web content extracted from URLs shared on Reddit Gokaslan and Cohen [2019b],[4] and a subset of CommonCrawl that it is said to resemble the "story-like" style of Winograd schemas [STORIES; Trinh and Le, 2018].[5] RoBERTa attains better performance on an assortment of tasks than its predecessors, making it our baseline of choice.

Although RoBERTa's pretraining corpus is derived from multiple sources, it has not yet been established if these sources are diverse enough to generalize to most of the variation in the English language. In other words, we would like to understand what is out of RoBERTa's domain. Towards this end, we explore further adaptation by continued pretraining of this large LM into two categories of unlabeled data: (i) large corpora of domain-specific text (§3.2), and (ii) available unlabeled data associated with a given task (§3.3).

3.2 Domain-Adaptive Pretraining

Our approach to domain-adaptive pretraining (DAPT) is straightforward—we continue pretraining RoBERTa on a large corpus of unlabeled domain-specific text. The four domains we focus on are biomedical (BioMed) papers, computer science (CS) papers, newstext from RealNews, and Amazon reviews. We choose these domains because they have been popular in previous work, and datasets for text classification are available in each. Table 3.1 lists the specifics of the unlabeled datasets in all four domains, as well as RoBERTa's training corpus.[6]

3.2.1 Analyzing Domain Similarity

Before performing DAPT, we attempt to quantify the similarity of the target domain to RoBERTa's pretraining domain. We consider domain vocabularies containing the top 10K most frequent unigrams (excluding

[1] https://github.com/soskek/bookcorpus
[2] https://github.com/google-research/bert
[3] https://github.com/fhamborg/news-please
[4] https://github.com/jcpeterson/openwebtext
[5] https://github.com/tensorflow/models/tree/master/research/lm_commonsense
[6] For BioMed and CS, we used an internal version of S2ORC that contains papers that cannot be released due to copyright restrictions.

Figure 3.2: Vocabulary overlap (%) between domains. PT denotes a sample from sources similar to RoBERTa's pretraining corpus. Vocabularies for each domain are created by considering the top 10K most frequent words (excluding stopwords) in documents sampled from each domain.

stopwords) in comparably sized random samples of held-out documents in each domain's corpus. We use 50K held-out documents for each domain other than REVIEWS, and 150K held-out documents in REVIEWS, since they are much shorter. We also sample 50K documents from sources similar to RoBERTa's pretraining corpus (i.e., BOOKCORPUS, STORIES, WIKIPEDIA, and REALNEWS) to construct the pretraining domain vocabulary, since the original pretraining corpus is not released. Figure 3.2 shows the vocabulary overlap across these samples. We observe that RoBERTa's pretraining domain has strong vocabulary overlap with NEWS and REVIEWS, while CS and BIOMED are far more dissimilar to the other domains. This simple analysis suggests the degree of benefit to be expected by adaptation of RoBERTa to different domains—the more dissimilar the domain, the higher the potential for DAPT.

3.2.2 Experiments

Our LM adaptation follows the settings prescribed for training RoBERTa. We train RoBERTa on each domain for 12.5K steps, which amounts to single pass on each domain dataset, on a v3-8 TPU; see other details in Appendix §A.2.3. This second phase of pretraining results in four domain-adapted LMs, one for each domain. We present the masked LM loss of RoBERTa on each domain before and after DAPT in Table 3.1. We observe that masked LM loss decreases in all domains except NEWS after DAPT, where we observe a marginal increase. We discuss cross-domain masked LM loss in Appendix §A.2.4.

Domain	Task	Label Type	Train (Lab.)	Train (Unl.)	Dev.	Test	Classes
BIOMED	CHEMPROT	relation classification	4169	-	2427	3469	13
	†RCT	abstract sent. roles	18040	-	30212	30135	5
CS	ACL-ARC	citation intent	1688	-	114	139	6
	SCIERC	relation classification	3219	-	455	974	7
NEWS	HYPERPARTISAN	partisanship	515	5000	65	65	2
	†AGNEWS	topic	115000	-	5000	7600	4
REVIEWS	†HELPFULNESS	review helpfulness	115251	-	5000	25000	2
	†IMDB	review sentiment	20000	50000	5000	25000	2

Table 3.2: Specifications of the various target task datasets. † indicates high-resource settings. Sources: CHEMPROT Kringelum et al. [2016], RCT Dernoncourt and Lee [2017], ACL-ARC Jurgens et al. [2018], SCIERC Luan et al. [2018], HYPERPARTISAN Kiesel et al. [2019], AGNEWS Zhang et al. [2015], HELPFULNESS McAuley et al. [2015], IMDB Maas et al. [2011a].

Under each domain, we consider two text classification tasks, as shown in Table 3.2. Our tasks represent both high- and low-resource ($\leq$ 5K labeled training examples, and no additional unlabeled data) settings. For HYPERPARTISAN, we use the data splits from Beltagy et al. [2020]. For RCT, we represent all sentences in one long sequence for simultaneous prediction.

Baseline As our baseline, we use an off-the-shelf ROBERTA-base model and perform supervised fine-tuning of its parameters for each classification task. On average, ROBERTA is not drastically behind the state of the art at the time of this project (details in Appendix §A.2.2), and serves as a good baseline since it provides a single LM to adapt to different domains.

Classification Architecture Following standard practice [Devlin et al., 2019b] we pass the final layer `[CLS]` token representation to a task-specific feedforward layer for prediction.

Results Test results are shown under the DAPT column of Table 3.3. We observe that DAPT improves over ROBERTA in all domains. For BIOMED, CS, and REVIEWS, we see consistent improvements over ROBERTA, demonstrating the benefit of DAPT when the target domain is more distant from ROBERTA's source domain. The pattern is consistent across high- and low- resource settings. Although DAPT does not increase performance on AGNEWS, the benefit we observe in HYPERPARTISAN suggests that DAPT may be useful even for tasks that align more closely with ROBERTA's source domain.

Dom.	Task	RoBa.	DAPT	¬DAPT
BM	ChemProt	$81.9_{1.0}$	$\mathbf{84.2}_{0.2}$	$79.4_{1.3}$
	†RCT	$87.2_{0.1}$	$\mathbf{87.6}_{0.1}$	$86.9_{0.1}$
CS	ACL-ARC	$63.0_{5.8}$	$\mathbf{75.4}_{2.5}$	$66.4_{4.1}$
	SciERC	$77.3_{1.9}$	$\mathbf{80.8}_{1.5}$	$79.2_{0.9}$
News	Hyp.	$86.6_{0.9}$	$\mathbf{88.2}_{5.9}$	$76.4_{4.9}$
	†AGNews	$\mathbf{93.9}_{0.2}$	$\mathbf{93.9}_{0.2}$	$93.5_{0.2}$
Rev.	†Helpful.	$65.1_{3.4}$	$\mathbf{66.5}_{1.4}$	$65.1_{2.8}$
	†IMDB	$95.0_{0.2}$	$\mathbf{95.4}_{0.2}$	$94.1_{0.4}$

Table 3.3: Comparison of RoBERTa (RoBa.) and DAPT to adaptation to an *irrelevant* domain (¬DAPT). Reported results are test macro-F_1, except for ChemProt and RCT, for which we report micro-F_1, following Beltagy et al. [2019]. We report averages across five random seeds, with standard deviations as subscripts. † indicates high-resource settings. Best task performance is boldfaced. See §3.2.3 for our choice of irrelevant domains.

3.2.3 Domain Relevance for DAPT

Additionally, we compare DAPT against a setting where for each task, we adapt the LM to a domain **outside** the domain of interest. This controls for the case in which the improvements over RoBERTa might be attributed simply to exposure to more data, regardless of the domain. In this setting, for News, we use a CS LM; for Reviews, a BioMed LM; for CS, a News LM; for BioMed, a Reviews LM. We use the vocabulary overlap statistics in Figure 3.2 to guide these choices.

Our results are shown in Table 3.3, where the last column (¬DAPT) corresponds to this setting. For each task, DAPT significantly outperforms adapting to an irrelevant domain, suggesting the importance of pretraining on domain-relevant data. Furthermore, we generally observe that ¬DAPT results in worse performance than even RoBERTa on end-tasks. Taken together, these results indicate that in most settings, exposure to more data without considering domain relevance is detrimental to end-task performance. However, there are two tasks (SciERC and ACL-ARC) in which ¬DAPT marginally *improves* performance over RoBERTa. This may suggest that in some cases, continued pretraining on any additional data is useful, as noted in Baevski et al. [2019].

IMDB review	REALNEWS article
"The Shop Around the Corner" is one of the great films from director Ernst Lubitsch. In addition to the talents of James Stewart and Margaret Sullavan, it's filled with a terrific cast of top character actors such as Frank Morgan and Felix Bressart. [...] The makers of "You've Got Mail" claim their film to be a remake, but that's just nothing but a lot of inflated self praise. Anyway, if you have an affection for romantic comedies of the 1940's, you'll find "The Shop Around the Corner" to be nothing short of wonderful. Just as good with repeat viewings.	[...] Three great festive films... The Shop Around the Corner (1940) Delightful Comedy by Ernst Lubitsch stars James Stewart and Margaret Sullavan falling in love at Christmas. Remade as You've Got Mail. [...]
HELPFULNESS review	**REALNEWS article**
Simply the Best! I've owned countless Droids and iPhones, but this one destroys them all. Samsung really nailed it with this one, extremely fast, very pocketable, gorgeous display, exceptional battery life, good audio quality, perfect GPS & WiFi performance, transparent status bar, battery percentage, ability to turn off soft key lights, superb camera for a smartphone and more! [...]	We're living in a world with a new Samsung. [...] more on battery life later [...] Exposure is usually spot on and focusing is very fast. [...] The design, display, camera and performance are all best in class, and the phone feels smaller than it looks. [...]

Table 3.4: Examples that illustrate how some domains might have overlaps with others, leading to unexpected positive transfer. We highlight expressions in the reviews that are also found in the REALNEWS articles.

3.2.4 Domain Overlap

Our analysis of DAPT is based on prior intuitions about how task data is assigned to specific domains. For instance, to perform DAPT for HELPFULNESS, we only adapt to AMAZON reviews, but not to any REALNEWS articles. However, the gradations in Figure 3.2 suggest that the boundaries between domains are in some sense fuzzy; for example, 40% of unigrams are shared between REVIEWS and NEWS. As further indication of this overlap, we also qualitatively identify documents that overlap cross-domain: in Table 3.4, we showcase reviews and REALNEWS articles that are similar to these reviews. In fact, we find that adapting ROBERTA to NEWS not as harmful to its performance on REVIEWS tasks (DAPT on NEWS achieves $65.5_{2.3}$ on HELPFULNESS and $95.0_{0.1}$ on IMDB).

Although this analysis is by no means comprehensive, it indicates that the factors that give rise to observable domain differences are likely not mutually exclusive. It is possible that pretraining beyond conventional domain boundaries could result in more effective DAPT; we leave this investigation to future work. In general, the provenance of data, including the processes by which corpora are curated, must be kept in mind when designing pretraining procedures and creating new benchmarks that test out-of-domain generalization abilities.

Domain	Task	RoBERTa	Additional Pretraining Phases		
			DAPT	TAPT	DAPT + TAPT
BioMed	ChemProt	$81.9_{1.0}$	$84.2_{0.2}$	$82.6_{0.4}$	$\mathbf{84.4}_{0.4}$
	†RCT	$87.2_{0.1}$	$87.6_{0.1}$	$87.7_{0.1}$	$\mathbf{87.8}_{0.1}$
CS	ACL-ARC	$63.0_{5.8}$	$75.4_{2.5}$	$67.4_{1.8}$	$\mathbf{75.6}_{3.8}$
	SciERC	$77.3_{1.9}$	$80.8_{1.5}$	$79.3_{1.5}$	$\mathbf{81.3}_{1.8}$
News	HyperPartisan	$86.6_{0.9}$	$88.2_{5.9}$	$\mathbf{90.4}_{5.2}$	$90.0_{6.6}$
	†AGNews	$93.9_{0.2}$	$93.9_{0.2}$	$94.5_{0.1}$	$\mathbf{94.6}_{0.1}$
Reviews	†Helpfulness	$65.1_{3.4}$	$66.5_{1.4}$	$68.5_{1.9}$	$\mathbf{68.7}_{1.8}$
	†IMDB	$95.0_{0.2}$	$95.4_{0.1}$	$95.5_{0.1}$	$\mathbf{95.6}_{0.1}$

Table 3.5: Results on different phases of adaptive pretraining compared to the baseline RoBERTa (col. 1). Our approaches are DAPT (col. 2, §3.2), TAPT (col. 3, §3.3), and a combination of both (col. 4). Reported results follow the same format as Table 3.3. State-of-the-art results we can compare to: ChemProt (84.6), RCT (92.9), ACL-ARC (71.0), SciERC (81.8), HyperPartisan (94.8), AGNews (95.5), IMDB (96.2); references in §A.2.2.

3.3 Task-Adaptive Pretraining

Datasets curated to capture specific tasks of interest tend to cover only a subset of the text available within the broader domain. For example, the ChemProt dataset for extracting relations between chemicals and proteins focuses on abstracts of recently-published, high-impact articles from hand-selected PubMed categories Krallinger et al. [2017, 2015]. We hypothesize that such cases where the task data is a narrowly-defined subset of the broader domain, pretraining on the task dataset itself or data relevant to the task may be helpful.

Task-adaptive pretraining (TAPT) refers to pretraining on the unlabeled training set for a given task; prior work has shown its effectiveness [e.g. Howard and Ruder, 2018]. Compared to domain-adaptive pretraining (DAPT; §3.2), the task-adaptive approach strikes a different trade-off: it uses a far smaller pretraining corpus, but one that is much more task-relevant (under the assumption that the training set represents aspects of the task well). This makes TAPT much less expensive to run than DAPT, and as we show in our experiments, the performance of TAPT is often competitive with that of DAPT.

BIOMED	RCT	CHEMPROT
TAPT	$87.7_{0.1}$	$82.6_{0.5}$
Transfer-TAPT	$87.1_{0.4}$ (↓0.6)	$80.4_{0.6}$ (↓2.2)

CS	ACL-ARC	SCIERC
TAPT	$67.4_{1.8}$	$79.3_{1.5}$
Transfer-TAPT	$64.1_{2.7}$ (↓3.3)	$79.1_{2.5}$ (↓0.2)

NEWS	HYPERPARTISAN	AGNEWS
TAPT	$89.9_{9.5}$	$94.5_{0.1}$
Transfer-TAPT	$82.2_{7.7}$ (↓7.7)	$93.9_{0.2}$ (↓0.6)

REVIEWS	HELPFULNESS	IMDB
TAPT	$68.5_{1.9}$	$95.7_{0.1}$
Transfer-TAPT	$65.0_{2.6}$ (↓3.5)	$95.0_{0.1}$ (↓0.7)

Table 3.6: Though TAPT is effective (Table 3.5), it is harmful when applied *across* tasks. These findings illustrate differences in task distributions within a domain.

3.3.1 Experiments

Similar to DAPT, task-adaptive pretraining consists of a second phase of pretraining ROBERTA, but only on the available task-specific training data. In contrast to DAPT, which we train for 12.5K steps, we perform TAPT for 100 epochs. We artificially augment each dataset by randomly masking different words (using the masking probability of 0.15) across epochs. As in our DAPT experiments, we pass the final layer `[CLS]` token representation to a task-specific feedforward layer for classification.

Our results are shown in the TAPT column of Table 3.5. TAPT consistently improves the ROBERTA baseline for all tasks across domains. Even on the news domain, which was part of ROBERTA pretraining corpus, TAPT improves over ROBERTA, showcasing the advantage of task adaptation. Particularly remarkable are the relative differences between TAPT and DAPT. DAPT is more resource intensive (see Table 3.9 in §3.4.3), but TAPT manages to match its performance in some of the tasks, such as SCIERC. In RCT, HYPERPARTISAN, AGNEWS, HELPFULNESS, and IMDB, the results even exceed those of DAPT, highlighting the efficacy of this cheaper adaptation technique.

Combined DAPT and TAPT We investigate the effect of using both adaptation techniques together. We begin with ROBERTA and apply DAPT then TAPT under this setting. The three phases of pretraining add up to make this the most computationally expensive of all our settings (see Table 3.9). As expected, combined domain- and task-adaptive pretraining achieves the best performance on all tasks (Table 3.5).[7]

Overall, our results show that DAPT followed by TAPT achieves the best of both worlds of domain and task awareness, yielding the best performance. While we speculate that TAPT followed by DAPT would be

[7]Results on HYPERPARTISAN match those of TAPT, within a standard deviation arising from the five seeds.

Pretraining	BIOMED RCT-500	NEWS HYP.	REVIEWS IMDB †
TAPT	$79.8_{1.4}$	$90.4_{5.2}$	$95.5_{0.1}$
DAPT + TAPT	$83.0_{0.3}$	$90.0_{6.6}$	$95.6_{0.1}$
Curated-TAPT	$83.4_{0.3}$	$89.9_{9.5}$	$95.7_{0.1}$
DAPT + Curated-TAPT	$\mathbf{83.8}_{0.5}$	$\mathbf{92.1}_{3.6}$	$\mathbf{95.8}_{0.1}$

Table 3.7: Augmenting training data with additional curated data improves performance. Mean test set macro-F_1 (for HYP. and IMDB) and micro-F_1 (for RCT-500), with Curated-TAPT across five random seeds, with standard deviations as subscripts. † indicates high-resource settings.

susceptible to catastrophic forgetting of the task-relevant corpus Yogatama et al. [2019], alternate methods of combining the procedures may result in better downstream performance. Future work may explore pretraining with a more sophisticated curriculum of domain and task distributions.

Cross-Task Transfer We complete the comparison between DAPT and TAPT by exploring whether adapting to one task transfers to other tasks in the same domain. For instance, we further pretrain the LM using the RCT unlabeled data, fine-tune it with the CHEMPROT labeled data, and observe the effect. We refer to this setting as Transfer-TAPT. Our results for tasks in all four domains are shown in Table 3.6. We see that TAPT optimizes for single task performance, to the detriment of cross-task transfer. These results demonstrate that data distributions of tasks within a given domain might differ. Further, this could also explain why adapting only to a broad domain is not sufficient, and why TAPT after DAPT is effective.

3.4 Augmenting Training Data for Task-Adaptive Pretraining

In §3.3, we continued pretraining the LM for task adaptation using only the training data for a supervised task. Inspired by the success of TAPT, we next investigate another setting where a larger pool of unlabeled data from the task distribution exists, typically curated by humans.

We explore two scenarios. First, for three tasks (RCT, HYPERPARTISAN, and IMDB) we use this larger pool of unlabeled data from an available human-curated corpus (§3.4.1). Next, we explore *retrieving* related unlabeled data for TAPT, from a large unlabeled in-domain corpus, for tasks where extra human-curated data is unavailable (§3.4.2).

3.4.1 Human Curated-TAPT

Dataset creation often involves collection of a large unlabeled corpus from known sources. This corpus is then downsampled to collect annotations, based on the annotation budget. The larger unlabeled corpus is thus expected to have a similar distribution to the task's training data. Moreover, it is usually available. We explore the role of such corpora in task-adaptive pretraining.

Data We simulate a low-resource setting RCT-500, by downsampling the training data of the RCT dataset to 500 examples (out of 180K available), and treat the rest of the training data as unlabeled. The HYPERPARTISAN shared task Kiesel et al. [2019] has two tracks: low- and high-resource. We use 5K documents from the high-resource setting as Curated-TAPT unlabeled data and the original low-resource training documents for task fine-tuning. For IMDB, we use the extra unlabeled data manually curated by task annotators, drawn from the same distribution as the labeled data Maas et al. [2011a].

Results We compare Curated-TAPT to TAPT and DAPT + TAPT in Table 3.7. Curated-TAPT further improves our prior results from §3.3 across all three datasets. Applying Curated-TAPT after adapting to the domain results in the largest boost in performance on all tasks; in HYPERPARTISAN, DAPT + Curated-TAPT is within standard deviation of Curated-TAPT. Moreover, curated-TAPT achieves 95% of the performance of DAPT + TAPT with the fully labeled RCT corpus (Table 3.5) with only 0.3% of the labeled data. These results suggest that curating large amounts of data from the task distribution is extremely beneficial to end-task performance. We recommend that task designers release a large pool of unlabeled task data for their tasks to aid model adaptation through pretraining.

3.4.2 Automated Data Selection for TAPT

Consider a low-resource scenario without access to large amounts of unlabeled data to adequately benefit from TAPT, as well as absence of computational resources necessary for DAPT (see Table 3.9 for details of computational requirements for different pretraining phases). We propose simple unsupervised methods to retrieve unlabeled text that aligns with the task distribution, from a large in-domain corpus. Our approach finds task-relevant data from the domain by embedding text from both the task and domain in a shared space,

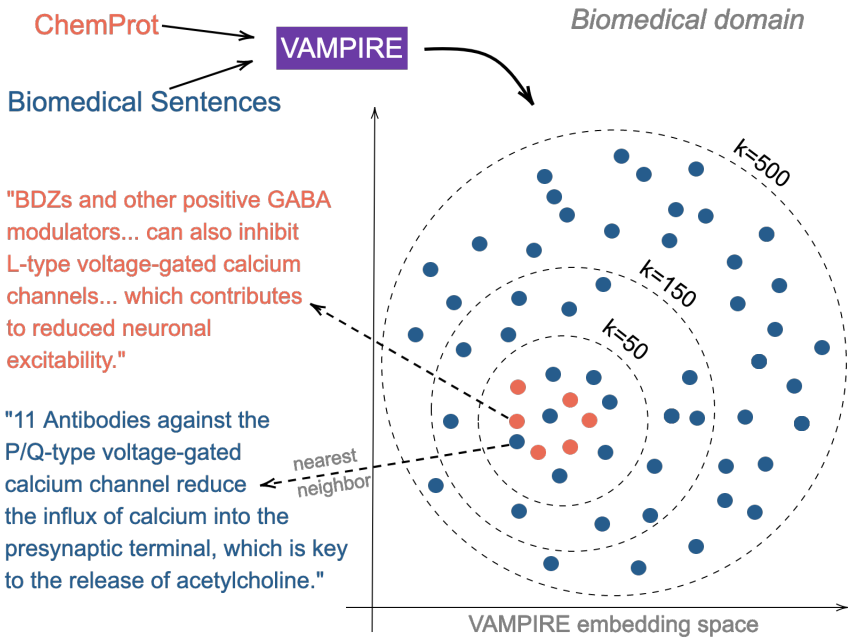

Figure 3.3: An illustration of automated data selection (§3.4.2). We map unlabeled CHEMPROT and 1M BIOMED sentences to a shared vector space using the VAMPIRE model trained on these sentences. Then, for each CHEMPROT sentence, we identify k nearest neighbors, from the BIOMED domain.

then selects candidates from the domain based on queries using the task data. Importantly, the embedding method must be lightweight enough to embed possibly millions of sentences in a reasonable time.

Given these constraints, we employ VAMPIRE (Gururangan et al., 2019; Figure 3.3), a lightweight bag-of-words language model. We pretrain VAMPIRE on a large deduplicated[8] sample of the domain (1M sentences) to obtain embeddings of the text from both the task and domain sample. We then select k candidates of each task sentence from the domain sample, in embeddings space. Candidates are selected (i) via nearest neighbors selection (kNN-TAPT)[9], or (ii) randomly (RAND-TAPT). We continue pretraining ROBERTA on this augmented corpus with both the task data (as in TAPT) as well as the selected candidate pool.

Results Results in Table 3.8 show that kNN-TAPT outperforms TAPT for all cases. RAND-TAPT is generally worse than kNN-TAPT, but within a standard deviation arising from 5 seeds for RCT and ACL-ARC. As we increase k, kNN-TAPT performance steadily increases, and approaches that of DAPT. Future work might

[8]We deduplicated this set to limit computation, since different sentences can share neighbors.
[9]We use a flat search index with cosine similarity between embeddings with the FAISS Johnson et al. [2019] library.

Pretraining	BIOMED		CS
	CHEMPROT	RCT-500	ACL-ARC
ROBERTA	$81.9_{1.0}$	$79.3_{0.6}$	$63.0_{5.8}$
TAPT	$82.6_{0.4}$	$79.8_{1.4}$	$67.4_{1.8}$
RAND-TAPT	$81.9_{0.6}$	$80.6_{0.4}$	$69.7_{3.4}$
50NN-TAPT	$83.3_{0.7}$	$80.8_{0.6}$	$70.7_{2.8}$
150NN-TAPT	$83.2_{0.6}$	$81.2_{0.8}$	$73.3_{2.7}$
500NN-TAPT	$83.3_{0.7}$	$81.7_{0.4}$	$\mathbf{75.5}_{1.9}$
DAPT	$\mathbf{84.2}_{0.2}$	$\mathbf{82.5}_{0.5}$	$75.4_{2.5}$

Table 3.8: Data selection performance improvement with kNN-TAPT. Mean test set micro-F_1 (for CHEMPROT and RCT) and macro-F_1 (for ACL-ARC), across five random seeds, with standard deviations as subscripts, comparing RAND-TAPT (with 50 candidates) and kNN-TAPT selection. Neighbors of the task data are selected from the domain data.

consider a closer study of kNN-TAPT, more sophisticated data selection methods, and the tradeoff between the diversity and task relevance of selected examples.

3.4.3 Computational Requirements

The computational requirements for all our adaptation techniques on RCT-500 in the BIOMED domain in Table 3.9. TAPT is nearly 60 times faster to train than DAPT on a single v3-8 TPU and storage requirements for DAPT on this task are 5.8M times that of TAPT. Our best setting of DAPT + TAPT amounts to three phases of pretraining, and at first glance appears to be very expensive. However, once the LM has been adapted to a broad domain, it can be reused for multiple tasks within that domain, with only a single additional TAPT phase per task. While Curated-TAPT tends to achieve the best cost-benefit ratio in this comparison, one must also take into account the cost of curating large in-domain data. Automatic methods such as kNN-TAPT are much cheaper than DAPT.

3.5 Related Work

Transfer learning for domain adaptation Prior work has shown the benefit of continued pretraining in domain [Alsentzer et al., 2019; Chakrabarty et al., 2019; Lee et al., 2019].[10] We have contributed further

[10] In contrast, Peters et al. [2019] find that the Jensen-Shannon divergence on term distributions between BERT's pretraining corpora and each MULTINLI domain Williams et al. [2018] does not predict its performance, though this might be an isolated finding specific to the MultiNLI dataset.

Pretraining	Steps	Docs.	Storage	F_1
RoBERTa	-	-	-	$79.3_{0.6}$
TAPT	0.2K	500	80KB	$79.8_{1.4}$
50NN-TAPT	1.1K	24K	3MB	$80.8_{0.6}$
150NN-TAPT	3.2K	66K	8MB	$81.2_{0.8}$
500NN-TAPT	9.0K	185K	24MB	$81.7_{0.4}$
Curated-TAPT	8.8K	180K	27MB	$\mathbf{83.4}_{0.3}$
DAPT	12.5K	25M	47GB	$82.5_{0.5}$
DAPT + TAPT	12.6K	25M	47GB	$83.0_{0.3}$

Table 3.9: Computational requirements for adapting to the RCT-500 task. Here, we compare DAPT (§3.2) and the various TAPT modifications described in §3.3 and §3.4.

investigation of the effects of a shift between a large, diverse pretraining corpus and target domain on task performance. Other studies [e.g., Huang et al., 2019] have trained language models (LMs) in their domain of interest, from scratch. In contrast, our work explores multiple domains, and is arguably more cost effective, since we continue pretraining an already powerful LM.

Task-adaptive pretraining Continued pretraining of a LM on the unlabeled data of a given task (TAPT) has been show to be beneficial for end-task performance [e.g. in Howard and Ruder, 2018; Phang et al., 2018; Sun et al., 2019a]. In the presence of *domain shift* between train and test data distributions of the same task, domain-adaptive pretraining (DAPT) is sometimes used to describe what we term TAPT Logeswaran et al. [2019]; Han and Eisenstein [2019]. Related approaches include language modeling as an auxiliary objective to task classifier fine-tuning [Chronopoulou et al., 2019; Radford et al., 2018a] or consider simple syntactic structure of the input while adapting to task-specific data Swayamdipta et al. [2019]. We compare DAPT and TAPT as well as their interplay with respect to dataset size for continued pretraining (hence, expense of more rounds of pretraining), relevance to a data sample of a given task, and transferability to other tasks and datasets. See Table A.2 in Appendix §A.2.1 for a summary of multi-phase pretraining strategies from related work.

Data selection for transfer learning Selecting data for transfer learning has been explored in NLP [Moore and Lewis, 2010; Ruder and Plank, 2017; Zhang et al., 2019, among others]. Dai et al. [2019] focus on identifying the most suitable corpus to pretrain a LM from scratch, for a single task: NER, whereas we select relevant *examples* for various tasks in §3.4.2. Concurrent to our work, Aharoni and Goldberg [2020a] propose

	Training Data		
	Domain (Unlabeled)	Task (Unlabeled)	Task (Labeled)
RoBERTa			✓
DAPT	✓		✓
TAPT		✓	✓
DAPT + TAPT	✓	✓	✓
kNN-TAPT	(Subset)	✓	✓
Curated-TAPT		(Extra)	✓

Table 3.10: Summary of strategies for multi-phase pretraining explored in this chapter.

data selection methods for NMT based on cosine similarity in embedding space, using DistilBERT Sanh et al. [2019] for efficiency. In contrast, we use VAMPIRE, and focus on augmenting TAPT data for text classification tasks. Khandelwal et al. [2020] introduced kNN-LMs that allows easy domain adaptation of pretrained LMs by simply adding a datastore per domain and no further training; an alternative to integrate domain information in an LM. Our study of human-curated data §3.4.1 is related to *focused crawling* Chakrabarti et al. [1999] for collection of suitable data, especially with LM reliance Remus and Biemann [2016].

What is a domain? Despite the popularity of domain adaptation techniques, most research and practice seems to use an intuitive understanding of domains. A small body of work has attempted to address this question [Lee, 2001; Eisenstein et al., 2014a; van der Wees et al., 2015; Plank, 2016a; Ruder et al., 2016, among others]. For instance, Aharoni and Goldberg [2020a] define domains by implicit clusters of sentence representations in pretrained LMs. Our results show that DAPT and TAPT complement each other, which suggests a spectra of domains defined around tasks at various levels of granularity (e.g., Amazon reviews for a specific product, all Amazon reviews, all reviews on the web, the web).

3.6 Conclusion

We investigate several variations for adapting pretrained LMs to domains and tasks within those domains, summarized in Table 3.10. Our experiments reveal that even a model of hundreds of millions of parameters struggles to encode the complexity of a single textual domain, let alone all of language. We show that

pretraining the model towards a specific task or small corpus can provide significant benefits. Our findings suggest it may be valuable to complement work on ever-larger LMs with parallel efforts to identify and use domain- and task-relevant corpora to specialize models.

While our results demonstrate how these approaches can improve RoBERTa, a powerful LM, the approaches we studied are general enough to be applied to any pretrained LM. Since the publication of this work, language models have been adapted with continued pretraining to great success for programming assistance [Rozière et al., 2023; Chen et al., 2021b], instruction following [Zhang et al., 2023], and biomedicine [Chen et al., 2023]. Our work points to numerous future directions, such as better data selection for TAPT, efficient adaptation large pretrained language models to distant domains, and building reusable language models after adaptation.

3.7 Limitations

Our definition of domains in this work dependent on heuristically determined labels (e.g., biomedicine or computer science). While there are relatively clear delineations between these domains in terms of their applications, it is possible that heuristically defined domains do not result in the optimal segmentation of data for downstream tasks. We explore a more data-driven definition of domain in Chapter 5. Furthermore, the methods we present in this work are susceptible to catastrophic forgetting, since all parameters are finetuned to narrow data distributions. In 4, we will explore methods of addressing this limitation.

Chapter 4

Incoporating Language Variation with Conditional Computation

The usefulness of multi-stage adaptive pretraining suggests that incorporating language variation into language model training algorithms can significantly improve their performance. However, as discussed in Chapter 1, the initial pretraining phase still assumes data homogeneity and does not condition on language variation: all parameters are updated to minimize the loss on all of the data. We refer to this approach as *dense training*. Dense training leaves variation in the data to be implicitly discovered [Aharoni and Goldberg, 2020b], assuming that models will be able to fit all language domains equally well.

While dense training is convenient, and densely trained LMs achieve impressive results [Brown et al., 2020], the approach has drawbacks with respect to generalization, efficiency, and flexibility. Even if training data is sourced from many domains, dense training can in practice emphasize subsets of the data in proportion to their ease of access [Oren et al., 2019a; Fan et al., 2020], limiting generalization to less prevalent domains. Updating all parameters of the network gets substantially more expensive as model size grows [Strubell et al., 2019], making fine-tuning or domain-adaptive pretraining harder to perform with smaller computational budgets. It is also difficult to adapt to new domains without forgetting the original data [McCloskey and Cohen, 1989; Aghajanyan et al., 2021b] or restrict access to certain domains the LM has been exposed to during training (e.g., those that contain hate speech; Bender et al. 2021b), leading to risks of unwanted behavior [Gehman et al., 2020a].

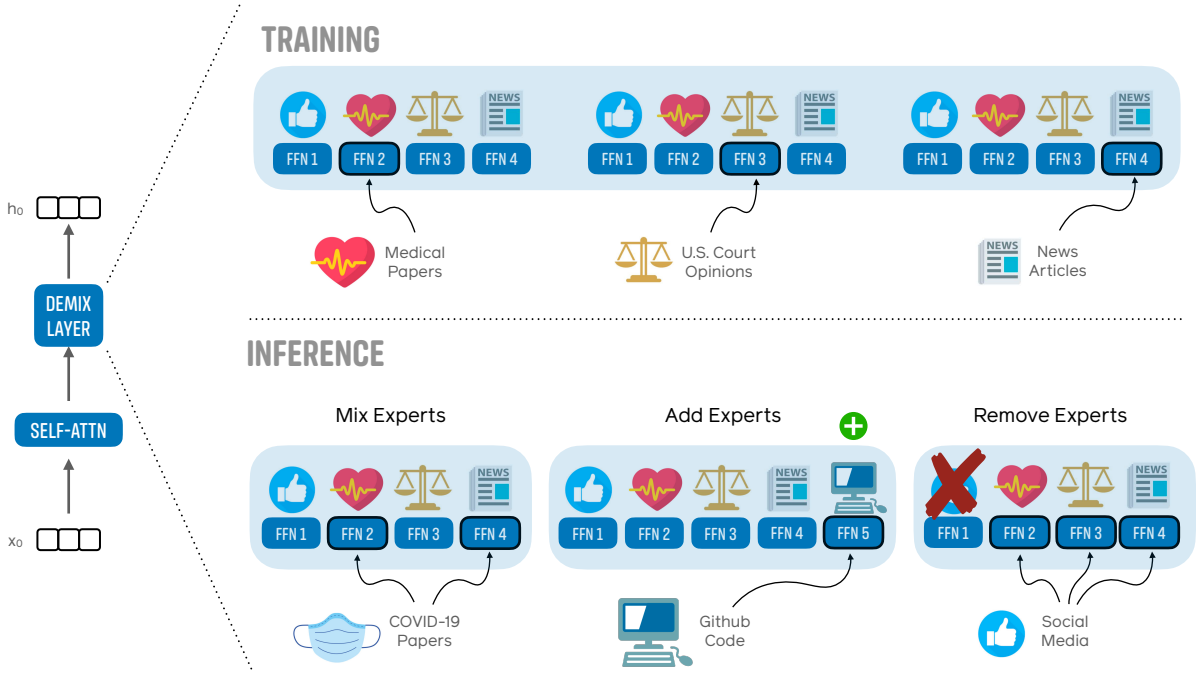

Figure 4.1: Illustration of a DEMIX layer in a single transformer block. During training, expert feedforward networks are conditionally activated based on the domain (here, document provenance) of the input sequence (i.e., scientific papers or court opinions). At inference time, the language model has new modular functions: domain experts can be mixed to handle heterogeneous domains, added to adapt to novel domains, or removed to "forget" unwanted domains. Image attribution: news icon from emojipedia.org; all other icons from istockphoto.com.

To address these limitations of dense training, we argue that LMs should be designed with *modularity*. We propose a modular LM that has components specialized to distinct domains in the training data, and can be customized at inference-time by mixing, adding, or removing these separated components as needed. This design principle emphasizes the ability to rapidly adapt the LM after training, a need that has been broadly advocated for language systems [Dinan et al., 2021; Lazaridou et al., 2021b].

We introduce modularity into an LM with a new domain expert (DEMIX) layer that explicitly conditions the LM on the domain of the input text (when it is known), or estimates the input domain during inference (when it is not known). A DEMIX layer is a drop-in substitute for a feedforward layer in a transformer LM (e.g., GPT-3), creating a specialized version of the layer (or *expert*) per domain (see Figure 4.1; §4.2).[1] We

[1]This is an example of conditional computation [Fedus et al., 2022b; Lepikhin et al., 2021; Lewis et al., 2021a; Roller et al., 2021], which follow prior literature on mixture of experts Jacobs et al. [1991a]; Shazeer et al. [2017]. Unlike dense training, conditional computation activates different parameters for different inputs. Instead of learning how to route data to experts, the DEMIX layer routing mechanism follows from a natural, observable segmentation of the data.

find that replacing every feedforward layer in the transformer with a DEMIX layer offers new affordances for modularity, addressing the challenges above, while improving performance in both training domains and novel test-time domains.

Although the concept of a domain lacks a rigorous definition in NLP, we use coarse provenance categories (e.g., whether a document is a medical research paper or a Reddit post) as a conditioning variable when training an LM with DEMIX layers (§4.1). Training on data from eight different domains, we find that DEMIX layers consistently improve in-domain performance (§4.3). However, because these categories may not be an optimal segmentation of the training data, or may lack coverage of test-time domains, naively selecting a single domain expert at test time can hurt generalization. Instead, we introduce a parameter-free probabilistic approach to dynamically estimate a *weighted mixture* of domains during inference (§4.4). Mixing experts improves DEMIX performance not only on *novel* test-time domains, but also on test data from the *training* domains, which may themselves be heterogeneous. Our results suggest that introducing modularity into an LM need not come at a cost to generalization performance.

Because DEMIX forces experts to specialize to domains, the overall model can be (partially) disentangled after training. Beyond mixing, we can add (§4.5) or remove (§4.6) domain experts, resulting in predictable changes in model behavior at inference time: adding experts allows for model adaptation without updating all parameters (hence avoiding forgetting), and removing experts allows for simulating the removal of training domains without additional training. Overall, DEMIX layers demonstrate benefits of explicitly conditioning on textual domains during language modeling, and our results suggest that these benefits persist at scale. Our code is publicly available.[2] This chapter is based on the work from:

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. 2022. DEMix Layers: Disentangling Domains for Modular Language Modeling. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5557–5576, Seattle, United States. Association for Computational Linguistics.

[2] http://github.com/kernelmachine/demix

	Domain	Corpus	# Train (Eval.) Tokens
TRAINING	1B	30M NewsWire sentences [Chelba et al., 2014b]	700M (10M)
	CS	1.89M full-text CS papers from S2ORC [Lo et al., 2020a]	4.5B (10M)
	LEGAL	2.22M U.S. court opinions, 1658 to 2018 [Caselaw Access Project]	10.5B (10M)
	BIOMED	3.2M full-text medical papers from S2ORC [Lo et al., 2020a]	9.5B (10M)
	WEBTEXT†	8M Web documents [Gokaslan and Cohen, 2019b]	6.5B (10M)
	REALNEWS†	35M articles from REALNEWS Zellers et al. [2019b]	15B (10M)
	REDDIT	Reddit comments from pushshift.io [Baumgartner et al., 2020]	25B (10M)
	REVIEWS†	30M Amazon product reviews [Ni et al., 2019]	2.1B (10M)
		Total	**73.8B (80M)**

	Domain	Corpus	# Train (Eval.) Tokens
NOVEL	ACL PAPERS	1.5K NLP papers from ACL [Dasigi et al., 2021]	1M (1M)
	BREAKING NEWS†	20K latest articles from 400 English news sites [Baly et al., 2018]	11M (1M)
	CONTRACTS†	500 commercial legal contracts [Hendrycks et al., 2021]	1.5M (1M)
	CORD-19	400K excerpts from COVID-19 research papers [Wang et al., 2020]	60M (10M)
	GITHUB	230K public Github repository contents [Github Archive Project]	200M (10M)
	GUTENBERG	3.2M copyright-expired books [Project Gutenberg]	3B (10M)
	TWEETS†	1M English tweets from 2013-2018	8M (1M)
	YELP REVIEWS†	6M Yelp restaurant reviews [Yelp Reviews]	600M (10M)

Table 4.1: Domains that make up our multi-domain training corpus, including the size of our training and evaluation (i.e. validation and test) data, in whitespace-separated tokens. † indicates datasets that we (partially) anonymize (§4.1). REDDIT was extracted and obtained by a third party and made available on `pushshift.io`, and was anonymized by Xu et al. [2021]; we use their version.

4.1 Multi-Domain Corpus

We center this study around a large, multi-domain corpus we constructed with explicit provenance metadata (Table 4.1). While other multi-domain corpora [Koh et al., 2021; Gao et al., 2020] cover many more domains and tasks, the corpus we introduce contains substantial metadata-tagged text for language modeling, as well as datasets with friendly licensing to support reproducibility.

4.1.1 Document Provenance as a Domain Label

While a growing body of work has attempted to address the structure and composition of language domains [Eisenstein et al., 2014b; Plank, 2016b; Aharoni and Goldberg, 2020b], fundamentally what a domain is remains a matter of debate. In this work, we focus on the *provenance* of a document, operationalized coarsely by the dataset we used to access it, which approximates a social process that produced it. Defining domains this way is easy and intuitive, conveys a great deal about the variation in a document's language, and aligns

with common practice in NLP research. However, other accounts of variation in language [e.g., Lucy and Bamman, 2021a], and richer notions of relationships among domains [e.g., hierarchies; Gururangan et al., 2020], may be studied in future work.

4.1.2 Corpus Description

The multi-domain corpus we use in this study consists of two parts. The first is a collection of **training** domains: text from eight domains of largely English text, listed at the top of Table 4.1, each of which vary in complexity and coverage and has been the subject of study in NLP.[3]

The second part is a collection of **novel** domains: text from eight domains also of largely English text, listed at the bottom of Table 4.1, which may or may not align with the training domains. The novel domains allow us to measure how models generalize to a more challenging data distribution shift, where domain boundaries may be less clear.

For most domains, we use the associated sources, listed in Table 4.1, without modification. For TWEETS, we use the Twitter Academic API. For GUTENBERG, we use the scraping tool provided in https://github.com/aparrish/gutenberg-dammit. For BREAKING NEWS, we identify a list of factually reliable English news sources, using the list curated by Baly et al. [2018]. Specifically, we filter on "high" factuality in the data provided in this repository: https://github.com/ramybaly/News-Media-Reliability. We then use Newspaper3K (https://newspaper.readthedocs.io/en/latest/) to scrape the latest 1000 articles from each site. After dropping duplicates, we arrive at about 20K articles from 400 news sources. We provide downloading links and general instructions at https://github.com/kernelmachine/demix-data/blob/main/DOWNLOAD_DATA.md.

To support future work with the data, we also release a standard API to download and preprocess it into a format compatible with Fairseq [Ott et al., 2019a].[4] We replace user identifiable information (e.g., email addresses, user handles, social security numbers, credit card numbers, phone numbers) with dummy tokens.[5]

[3]The metadata for each document includes at least its provenance, and in some cases more information (e.g., URLs, publication venue, or legal jurisdiction). Future work might explore more fine-grained notions of domain.

[4]https://github.com/kernelmachine/demix-data

[5]While it is difficult to anonymize data perfectly, especially at scale, we use a suite of regexes to identify commonly occurring identifiable information on the Internet.

4.2 DEMIX Layer

4.2.1 Background: Mixture-of-Experts Transformers

The transformer architecture is comprised of interleaved multi-head self-attention, layer-norms, and feedforward networks [Vaswani et al., 2017]. Each of these layers produces a vector representation for each of the input tokens. Our focus is on the feedforward component:

$$\mathbf{h}_{t,\ell} = \text{FFN}(\mathbf{h}_{t,\ell-1}), \tag{4.1}$$

where $\mathbf{h}_{t,\ell}$ is the vector for the tth token produced by layer ℓ.

Shazeer et al. [2017] propose a formulation of one or more feedforward layers as an ensemble of n experts $\text{FFN}_1, \ldots, \text{FFN}_n$, assigned weights respectively by functions $g_1, \ldots, g_n$:

$$\text{FFN}(\mathbf{h}_{t,\ell-1}) = \sum_{j=1}^{n} g_j(\mathbf{h}_{t,\ell-1}) \cdot \text{FFN}_j(\mathbf{h}_{t,\ell-1}) \tag{4.2}$$

The g function routes tokens to different experts, usually each a separate instance of the original feedforward network. If g routes to a single expert, then the computational cost (in floating-point operations; FLOPs) will be same as the original feedforward network, even though it has slightly more than n times as many parameters.

4.2.2 DEMIX Routing

Previous approaches *learn* the weighting functions g at a token-level, and either assign at most one [Fedus et al., 2022b] or two [Lepikhin et al., 2021] experts per token. This necessitates load balancing and other techniques to encourage the model to use all experts instead of relying on just a few [Fedus et al., 2022b; Lewis et al., 2021a].

We instead use domain metadata provided with training documents to route data to experts at the *document* (i.e., sequence) level. During training, every token in the same sequence is assigned to the same expert based on the domain label.

Let $\mathcal{D}$ denote the set of domain labels (i.e., the eight labels in Table 4.1). If we index the experts by $\mathcal{D}$ and $d \in \mathcal{D}$ is the domain label for the current training instance, then

$$g_j(\mathbf{h}_{t,\ell}) = \begin{cases} 1 & \text{if } j = d \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

While we assume that each *training* document is associated with a single domain label, we relax this requirement at inference time (§4.4), which improves model performance in mixed and unknown domain scenarios.

4.2.3 DEMIX Architecture

Our design results in one expert in a DEMIX layer per domain (i.e., eight experts for eight training domains in our multi-domain corpus).

We replace *every* feedforward layer in the transformer with a DEMIX layer, in contrast to previous work [Fedus et al., 2022b; Lepikhin et al., 2021] that interleaves shared and expert layers. Preliminary experiments showed that interleaving led to worse in-domain performance with DEMIX layers. We hypothesize that shared layers may serve as a bottleneck to find shared features between domains, and may impact performance adversely when training domains are highly different from one another.[6] Future work might perform careful comparisons of different architectural choices.

In this study, each expert FFN_j is a two-layer MLP with the same dimensions as the original FFN layer of the transformer. As with other conditional computation models [Fedus et al., 2022b; Lepikhin et al., 2021], this means that the effective number of parameters in the overall DEMIX LM increases (Table 4.2). While this incurs memory costs, the computational budget we consider in this study centers around runtime costs. DEMIX layers decrease the runtime costs of training the LM.

4.2.4 DEMIX Training

DEMIX layers increase the total parameters of the LM while also reducing GPU latency costs during training, effectively reducing runtime costs of training the LM.

[6] Indeed, preliminary experiments suggest that interleaving expert layers causes large performance hits in the most distinct domains, i.e., those with lower vocabulary overlap with other domains in the corpus.

DENSE training (also referred to as *data-parallel*) is usually implemented by copying model parameters to every GPU, feeding a different mini-batch of shuffled data to each GPU, computing a stochastic gradient for each mini-batch, and updating all parameters synchronously with the average stochastic gradient from across all GPUs.

To train an LM with DEMIX layers, we instead partition the GPUs among the domains, so that each GPU is assigned a single domain (along with its corresponding expert). During training, we fill a mini-batch with k sequences, where each sequence represents data from a particular domain, and we send each mini-batch to its dedicated domain expert. We use larger batch sizes by performing data-parallel training between expert parameters on GPUs assigned to the same domain; we assign n/8 GPUs to each domain (Table 4.2). To reduce overfitting, we ensure that each of these n/8 GPUs is assigned to different shards of their domain's training data.

We compare the training efficiency of DENSE and DEMIX models up to 1.3B parameters per GPU in Table 4.2. Compared to DENSE LMs, DEMIX layers achieve the same or slightly higher throughput (measured in TFLOPs/GPU) for the same total FLOPs per update, despite adding significantly more parameters.

DEMIX achieves higher throughput because we only synchronize expert parameters allocated to the same domain.[7] As we increase model size, this results in a reduction of latency costs between GPUs, and hence, faster training; instead of synchronizing parameters over n GPUs, we perform eight synchronizations over n/8 GPUs.[8]

In this work, we assume that there is sufficient data for each training domain that each expert can be exposed to the same amount of data, and load balancing between experts is not necessary. Future work may consider how varying the amount of data per domain influences absolute and relative performance across domains, especially in the long tail of rare domains.

While the total number of parameters of DEMIX LMs are substantially larger than their DENSE counterparts, since the practical training costs are essentially the same, we compare baselines in all subsequent experiments based on parameters *per GPU*, as we do in Table 4.2.

[7] Shared parameters are synchronized across all GPUs.

[8] While this technique reduces latency costs, the bandwidth costs are the same between DEMIX and DENSE models.

		Parameters per GPU			
		125M	350M	760M	1.3B
DENSE	**GPUs**	32	64	128	128
	Total Experts	0	0	0	0
	GPUs/expert	0	0	0	0
	Total params	125M	350M	760M	1.3B
	TFLOPs/update	556	3279	13,637	23,250
	TFLOPs/GPU	**31**	**37**	45	51
DEMIX	**GPUs**	32	64	128	128
	Total Experts	8	8	8	8
	GPUs/expert	4	8	16	16
	Total params	512M	1.8B	3.8B	7.0B
	TFLOPs/update	556	3279	13,637	23,250
	TFLOPs/GPU	**31**	**37**	**48**	**55**

Table 4.2: Our specifications for training DENSE and DEMIX LMs. All models are trained for about 48 hours on V100 GPUs. DEMIX layers increase the total parameters of the LM while maintaining (or increasing) throughput, measured in TFLOPs/GPU. We use the formula described in Narayanan et al. [2021] to calculate these metrics.

4.3 In-Domain Performance

The first set of experiments in this study considers the impact of replacing the conventional feedforward layers in a transformer LM with DEMIX layers. We run all experiments in this section with the training domains (Table 4.1).

4.3.1 Experimental Setup

Architecture and Input The model architecture is a randomly-initialized LM with the GPT-3 [Brown et al., 2020] architecture implemented in Fairseq [Ott et al., 2019a]. We experiment with multiple architectures (i.e., those of GPT-3 small, medium, large, and XL), at a maximum size of about 1.3B parameters per GPU. We use the GPT-2 [Radford et al., 2019a] vocabulary of 50,264 BPE types, and train with 1,024-token sequences, with cross-document boundaries. Each document has a beginning-of-sentence token prepended to it.

Hyperparameters We set the total number of training steps based on this allocated runtime, set 8% of these steps to be warm-up, and use the Adam optimizer [Kingma and Ba, 2017] with a polynomial learning rate decay. Learning rates are tuned for each model separately over {0.0001, 0.0003, 0.0005}, taking the fastest learning rate that avoids divergence. Each worker processes two sequences of length 1,024, and gradients are

	Parameters per GPU			
	125M	350M	760M	1.3B
DENSE	20.6	16.5	14.5	13.8
DENSE (Balanced)	19.9	15.8	14.3	13.6
+DOMAIN-TOKEN	19.2	15.9	14.3	**13.4**
DEMIX (naive)	18.4	15.5	14.2	13.8
DEMIX (cached; §4.4.4)	**17.8**	**14.7**	**13.9**	**13.4**

Table 4.3: Average of in-domain test-set perplexity across baselines. We discuss the last row in §4.4.4.

accumulated over 8 updates. We clip gradients if their L_2 norm exceeds 0.1. These settings are inspired by Lewis et al. [2021a].

Computational Budget We follow previous work in using runtime as the primary computational budget, which provides a better comparison of the practical costs of training conditional compute and dense models [Lewis et al., 2021a]. We assume a fixed budget of about 48 hours on NVIDIA V100 32GB GPUs. We display the number of GPUs used for each model size in Table 4.2; we chose these GPU budgets because larger models require more compute to train properly [Lewis et al., 2021a; Kaplan et al., 2020], and found these GPU budgets to result in stable training for each model size given mostly fixed hyperparameters.

Evaluation We report test-set perplexities after about 48 hours of training. In all tables, we report each result with respect to a set number of parameters per GPU, as in Table 4.2. As mentioned in §4.2.4, DEMIX LM will have a larger effective size than the DENSE LM at the same increased throughput.

4.3.2 Compared Models

DENSE The first baseline is a DENSE model that treats the data as homogeneous, i.e., it shares all parameters across all domains. Under this setup, the language model parameters are copied across all GPUs, and gradients computed during training are all-reduced across every GPU. There is no explicit conditioning on domain.

DENSE (Balanced) Under this setting, we train densely but ensure that the model is exposed to an equal amount of data from each domain. While there is still no explicit conditioning on domain, the gradient updates that the model makes during training are an average of those computed across all domains represented in a batch.

Domain	1.3B parameters per GPU		
	DENSE	DEMIX (naive)	DEMIX (cached prior; §4.4.4)
1B	11.8	11.5	**11.3**
CS	13.5	12.2	**12.1**
LEGAL	6.8	**6.7**	**6.7**
BIOMED	9.5	9.2	**9.1**
WEBTEXT	**13.8**	14.6	14.3
REALNEWS	**12.5**	13.3	13.1
REDDIT	28.4	30.6	**28.1**
REVIEWS	14.0	12.6	**12.5**
Average	13.8	13.8	**13.4**

Table 4.4: Test-set perplexity by domain, for an LM with 1.3B parameters per GPU. We discuss the last column in §4.4.4.

+DOMAIN-TOKEN This model is trained identically to DENSE (Balanced), but we prepend a token indicating the sequence's domain to every sequence block (during training and test time). A variant of this domain token is explored in some previous studies [Zellers et al., 2019b; Keskar et al., 2019]. This baseline provides domain information to the language model in the form of input supervision. We ignore the domain token when computing perplexity during evaluation.

DEMIX (naive) We replace every feedforward layer in the transformer with a DEMIX layer, as detailed in §4.2. Under this setting, the domain of the test data is known and revealed to the model (e.g., the CS expert is used for CS test data), which we refer to as *naive*. We also ensure that the model is exposed to an equal amount of data from each domain.

4.3.3 Results

Table 4.3 shows test-set perplexities, averaged across the eight training domains. First, we observe that domain balancing is consistently helpful for DENSE training. We find that balancing is especially important in cases in which there is an imbalance of domain prevalence, confirming similar observations from previous studies [Arivazhagan et al., 2019].[9]

Next, we observe that the benefits of additional domain information (i.e, domain tokens or DEMIX

[9]Balancing improves performance on most domains, but hurts performance relative to a DENSE baseline on the REDDIT domain. In the multi-domain corpus, there is far more REDDIT text than anything else; see Table 4.1.

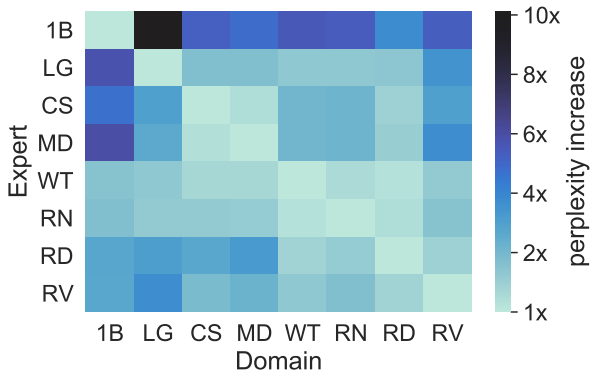

Figure 4.2: Domain experts in DEMIX specialize to their domain. We compute the above heatmap with a DEMIX LM with 1.3B parameters per GPU. Each cell of the heatmap is a ratio between an expert's test perplexity on a domain to that of the expert trained on that domain. The diagonal indicates that each expert has the best performance on its assigned domain. While some experts (e.g., 1B, BIOMED) do not transfer well to most domains in the training corpus, WEBTEXT and REALNEWS experts transfer much better, confirming their heterogeneity. Key: LG → LEGAL, MD → Med, WT → WEBTEXT, RN → REALNEWS, RD → REDDIT, RV → REVIEWS.

layers) are clearest for the smallest model; for larger models, the benefits are smaller but consistent. This result suggests that domain-specific information enables the model to better specialize to different domains in its training data. However, as the model size grows, the DENSE baseline becomes increasingly better at fitting the training domains, catching up to models with additional domain information, in the average case.

4.3.4 Domain Hetereogeneity

A more complete view of the experiments with the largest model is shown in Table 4.4. We see that even at scale, most training domains benefit from DEMIX layers in a naive setting (where the domain label is revealed at test time), but some do not; WEBTEXT, REALNEWS, and REDDIT fare worse than the DENSE baseline. We believe that this variation can be explained by heterogeneity within domains and varying degrees of similarity between them. DENSE training may be advantageous for domains that have a higher degree of overlap with other domains in the corpus (and therefore, benefit from parameter sharing).

To provide further evidence for this explanation, we measure the hetereogeneity of domains in the multi-domain corpus, according to a DEMIX LM. We plot a matrix of the perplexity changes across all domain experts in Figure 4.2, comparing all experts against the expert explicitly trained for each domain. As

the perplexity change tends lower, the corresponding expert has higher affinity to the target domain.

First, we observe that domain experts have the highest affinity to their assigned domain, indicating that they do specialize. We also observe that some experts, e.g., WEBTEXT, REALNEWS, and REDDIT, have relatively high affinities to many domains, suggesting that these domains are hetereogeneous. Separately we observe that an expert's affinity to a domain correlates positively with bigram overlap between the expert domain and target domain (r=0.40, t=3.45, p=0.001). This further suggests that similar domains have more closely aligned domain experts.

These findings suggest that a discrete notion of domain, while usually helpful on average (in our artificially constructed population of eight training domains), is too rigid. In the next section, we introduce new ways of softening Equation 4.3 into a mixture over domain experts, to improve performance on heterogeneous domains.

4.4 Mixing Experts at Inference Time

The previous section establishes that incorporating DEMIX layers improves LM performance on test data from *known* training domains. At inference time, the domain label was revealed to the model and used to select an expert within each DEMIX layer. In practice, however, text may not come with a domain label, may straddle multiple domains, or may not belong to any of the domains constructed at training time; the provenance of the data may even be unknown.

In these cases, rather than a hard choice among experts (Equation 4.3), we propose to treat $g_1, \ldots, g_n$ as mixture coefficients, transforming the domain membership of an input text into a matter of probabilistic belief. Unlike previously proposed mixture-of-experts formulations [Shazeer et al., 2017; Lepikhin et al., 2021], this approach introduces no new parameters and the weights are computed only at test time.[10]

To analyze inference-time behavior in mixed or unknown domain scenarios, we turn to the corpus of novel domains in the multi-domain corpus (Table 4.1). As mentioned in §4.1, these domains have fuzzier boundaries, compared to the training domains.

[10]We choose to explore inference-time mechanisms instead of training mechanisms to mix experts because 1) we want to avoid substantially increasing training costs, i.e., GPU communication between domain experts and 2) we want to maintain the modularity of experts. Exploring mechanisms for training expert mixtures while satisfying these desiderata is a rich area for future work.

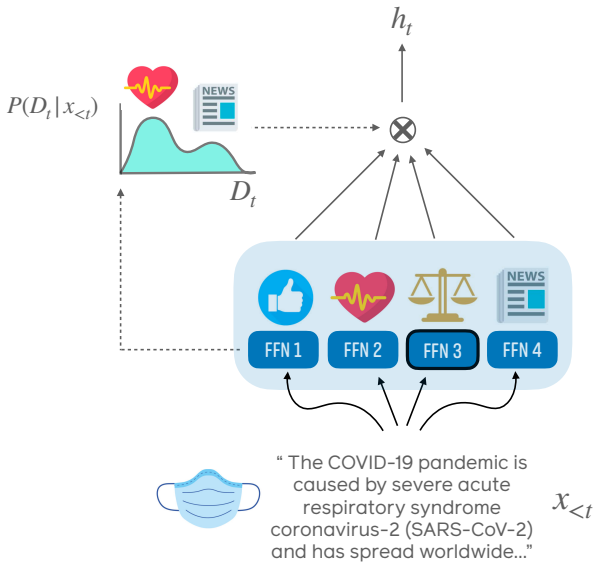

Figure 4.3: Illustration of inference with domain expert mixing. For a given input text $\boldsymbol{x}_{<t}$ from CORD-19, we estimate a posterior domain probabilities $p(D_t \mid \boldsymbol{x}_{<t})$, informed by a prior that is either iteratively updated during inference, or is precomputed and cached on held-out data. In this example, the model assigns highest domain probabilities to the medical and news domains. We use these probabilities in a weighted mixture of expert outputs to compute the hidden representation $\mathbf{h}_t$.

4.4.1 Dynamically Estimating Domain Membership

Consider the probabilistic view of language modeling, where we estimate $p(X_t \mid \boldsymbol{x}_{<t})$. We introduce a domain variable, D_t, alongside each word. We assume that this hidden variable depends on the history, $\boldsymbol{x}_{<t}$, so that:

$$p(X_t \mid \boldsymbol{x}_{<t}) = \sum_{j=1}^{n} p(X_t \mid \boldsymbol{x}_{<t}, D_t = j) \cdot \underbrace{p(D_t = j \mid \boldsymbol{x}_t)}_{g_j} \tag{4.4}$$

This model is reminiscent of class-based n-gram LMs [Brown et al., 1992] and their derivatives [Saul and Pereira, 1997].

We have already designed the DEMIX LM to condition on a domain label, giving a form for $p(X_t \mid \boldsymbol{x}_{<t}, D_t = j)$. The modification is to treat $g_1, \ldots, g_n$ as a posterior probability over domains, calculated at each timestep, given the history so far.

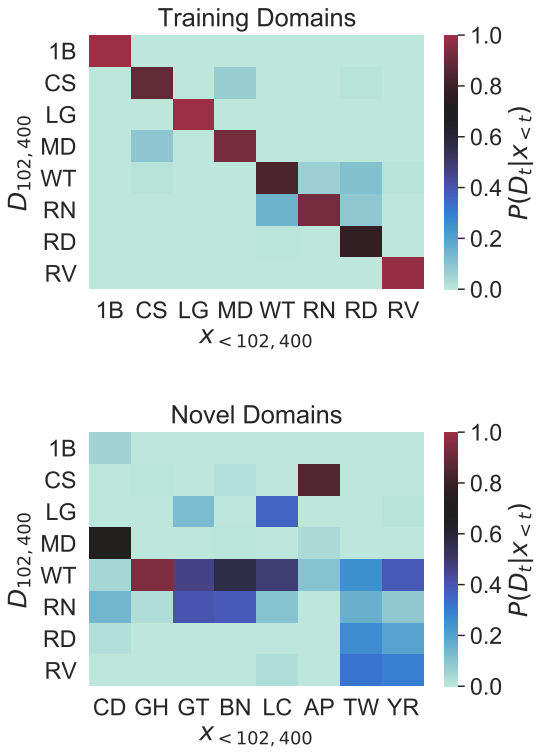

Figure 4.4: Estimates of posteriors $p(D_t \mid \boldsymbol{x}_{<t})$ with a DEMIX LM with 1.3B parameters per GPU, after 100 sequences (i.e., 102,400 tokens) of data in training domains (top heatmap) and new domains (bottom heatmap). Key: LG → LEGAL, MD → Med, WT → WEBTEXT, RN → REALNEWS, RD → REDDIT, RV → REVIEWS, CD → CORD-19, GH → GITHUB, GT → GUTENBERG, BN → BREAKING NEWS, LC → CONTRACTS, AP → ACL PAPERS, TW → TWEETS, YR → YELP REVIEWS.

To do this, we apply Bayes' rule:

$$p(D_t = j \mid \boldsymbol{x}_t) = \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{p(\boldsymbol{x}_{<t})} \tag{4.5}$$

$$= \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{\sum_{j'=1}^{n} p(\boldsymbol{x}_{<t} \mid D_t = j') \cdot p(D_t = j')} \tag{4.6}$$

The conditional probabilities of word sequences given a domain label, as noted above, are already defined by the DEMIX LM. For the prior over domain labels, we consider three alternatives:

Uniform Fix the prior to be uniform across the known domains.

Updating Set the prior at timestep t to be an exponentially-weighted moving average of the posteriors from previous timesteps:

$$p(D_t = j) \propto \sum_{t'=1}^{t-1} \lambda^{t-t'} \cdot p(D_{t'} = j \mid \boldsymbol{x}_{t'}) \tag{4.7}$$

During evaluation, this moving average is calculated over the posterior at the end of each sequence block. The decay factor avoids putting too much weight on calculations made early in the dataset, when posterior calculations are noisier (Appendix §A.3.1). We performed a small grid search over {0.1, 0.3, 0.5, 1.0} to set the value λ, and found that 0.3 worked well for most settings.

Cached If, prior to testing, some data from the test distribution is available, we calculate the posterior over domain labels from that data, and fix the prior to that estimate. Under this setting, we use 100 sequences (i.e., 102,400 tokens) from the development set to estimate the prior, which we found to result in stable posterior probabilities (see Appendix §A.3.1 for more details).

We display an illustration of the mixture technique in Figure 4.3.

4.4.2 Visualizing Domain Membership

In Figure 4.4, we plot the posteriors, calculated using the updating method above after 100 sequences of development data, each from training and novel domains. This evaluation is carried out using the DEMIX LM with 1.3B parameters per GPU from §4.3, with no modifications.

For known domains (top heatmap of Figure 4.4), the correct label has the highest posterior, but these datasets do not appear to be as distinct or mutually exclusive as we assume. For example, Reddit data is estimated to be around 80% REDDIT, 11% WEBTEXT, and 8% REALNEWS. More variation in the estimates is expected and observed for the new domains (bottom heatmap of Figure 4.4). While ACL PAPERS is mostly associated with the CS domain, and BREAKING NEWS mostly with the WEBTEXT and REALNEWS domains, CORD-19 is spread across BIOMED, REALNEWS, and 1B; YELP REVIEWS across REVIEWS, WEBTEXT, and REDDIT. The alignment of multiple domains like GITHUB and CONTRACTS primarily to WEBTEXT suggests the benefit of including a relatively heterogeneous domain in training.

	Parameters per GPU			
	125M	350M	760M	1.3B
DENSE	25.9	21.4	18.4	17.8
DENSE (B)	25.3	19.6	18.3	17.1
+DOMAIN-TOKEN	24.8	20.4	18.4	18.0
DEMIX (naive)	28.8	23.8	21.8	21.1
DEMIX (average)	27.2	22.4	21.5	20.1
DEMIX (uniform)	24.5	20.5	19.6	18.7
DEMIX (updating)	21.9	18.7	17.6	17.1
DEMIX (cached)	**21.4**	**18.3**	**17.4**	**17.0**

Table 4.5: Average perplexity on domains unseen during training. Mixing domain experts with a prior estimated using a small amount of data in the target domain outperforms all other baselines.

4.4.3 Experimental Setup

We experiment with the corpus of novel domains (Table 4.1) to test out-of-distribution performance. We evaluate the three mixture treatments of DEMIX layers (i.e., uniform, updating, and cached priors) against five baselines. Note that no new models are trained for this experiment beyond those used in §4.3.

DENSE and DENSE (Balanced) These are the basic baselines trained as in §4.3; there is no explicit reasoning about domain.

+DOMAIN-TOKEN Here test data is evaluated using each domain label token, and we choose the lowest among these perplexity values per test set.

DEMIX (naive) Similar to +DOMAIN-TOKEN, we evaluate the data separately with each of the eight experts, and report the lowest among these perplexity values per test set.

DEMIX (average) At every timestep, we take a simple average of the eight experts' predictions.

4.4.4 Results

Novel Domain Performance Results averaged across the eight novel domains are summarized in Table 4.5. Ensembling DEMIX experts outperforms DENSE baselines and using experts individually (i.e., the "naive" baseline), and caching a prior prior to evaluation results in the best average performance. While +DOMAIN-

TOKEN is competitive with naively using DEMIX layers in-domain (Table 4.3), it consistently underperforms DEMIX with a weighted mixture on the novel domains. We observe that ensembling DEMIX experts with a cached prior allows smaller models to match or outperform much larger DENSE models. We also find that weighted ensembling outperforms simple averaging, confirming the importance of sparsity in the expert mixture.

Examining per-domain performance, we find that DEMIX LMs with a cached prior either outperform DENSE baselines or closely match them. The largest improvement against DENSE baselines comes from the TWEETS domain, which are on average 67% better across all model sizes. This domain is heterogeneous according to the DEMIX model (Figure 4.4), confirming the importance of mixing experts for heterogeneous domains. These results demonstrate that conditioning the LM on domains during training need not come at a large cost to generalization to new domains, and in many cases can provide large boosts in performance over DENSE baselines.

In-Domain Performance We can also apply the expert mixture variant of inference (using a cached prior) to the training domains. We find that doing so is beneficial; see the last line of Table 4.3.

We see improvements in performance across all domains for every scale, though the largest improvements seem to come from hetereogeneous domains (across all model sizes, REDDIT improves on average 10.7%, WEBTEXT 2.4%, REALNEWS 1.9%), again confirming that our intuition that domain metadata may not perfectly align with the most effective domain boundaries.

4.5 Adaptive Pretraining with New Experts

As discussed in Chapter 3, domain-adaptive, continued pretraining[11] of a language model (DAPT) is a way to use unannotated, in-domain text to improve task performance. However, for a large model, DAPT with DENSE training (which we refer to as DENSE-DAPT) is expensive and may not be feasible on some computational budgets. Furthermore, DENSE-DAPT may result in forgetting what was learned during earlier training phases, limiting reusability.

The modular approach of DEMIX LMs allows the model to avoid forgetting training domains and adapt

[11] This approach typically precedes supervised fine-tuning on task data, hence *pre*training.

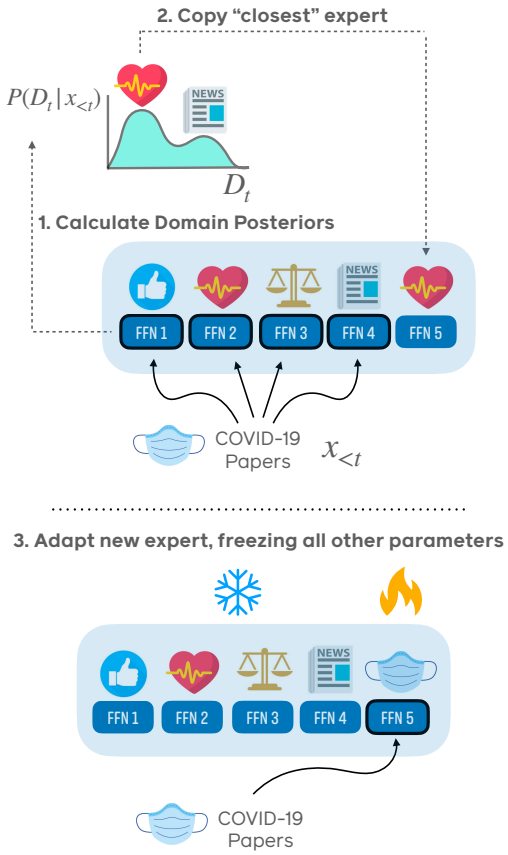

Figure 4.5: Illustration of DEMIX-DAPT. First, we estimate domain posteriors on a held out sample of the target domain (in this case, CORD-19). We then initialize a new expert with the parameters of the most probable expert under the domain posterior distribution. Finally, we adapt the parameters of the newly initialized expert to the target domain, keeping all other parameters in the LM frozen.

cheaply: we can train a new expert and add it to the DEMIX layers of the network without updating the other experts or the shared parameters. Because the original model is not changed, forgetting is impossible. We refer to this method of adaptation as DEMIX-DAPT.[12]

We display an illustration of DEMIX-DAPT in Figure 4.5. We instantiate a new expert in each DEMIX feedforward layer, initialize it with the parameters of the pretrained expert nearest to the new domain. We use the posterior calculations from §4.4 on a held-out sample to choose the most probable expert. We then train the added expert on target data, updating only the new expert parameters. For inference, we use the weighted mixture of domain experts with a cached prior (§4.4).

[12]Our proposed technique is reminiscent of *Progressive Neural Networks* [Rusu et al., 2016].

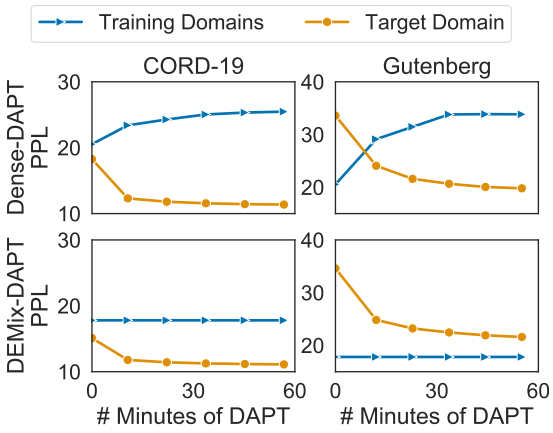

Figure 4.6: Adapting LMs with 125M parameters per GPU to CORD-19 or GUTENBERG. Top row: when performing DENSE-DAPT on a new domain (TARGET), average perplexity on all pretraining domains degrades. Bottom row: DEMIX-DAPT avoids that degradation while achieving close (in the case of GUTENBERG) or better (in the case of CORD-19) performance. The new CORD-19 expert was initialized with the BIOMED expert, and the new GUTENBERG expert was initialized with a WEBTEXT expert.

4.5.1 Experimental Setup

We compare DEMIX-DAPT to DENSE-DAPT on all novel domains. We report final test-set perplexity after adapting to each domain for 1 hour with 8 NVIDIA V100 32GB GPUs, tracking validation perplexity every 10 minutes for early stopping. We adapt to each novel domain with the same hyperparameters as the original phase of training (§4.3), except for a 10x smaller learning rate.

4.5.2 Results

Adding one expert We display examples of DEMIX-DAPT and DENSE-DAPT on a single additional domain in Figure 4.6. We observe that while DENSE-DAPT reduces perplexity on the novel domain, its performance on the training domains progressively worsens, displaying the forgetting effect (we show similar results in larger models in Appendix §A.3.2). In contrast, DEMIX-DAPT reduces perplexity on the novel domain *without* forgetting.

We generally observe that DEMIX-DAPT outperforms DENSE-DAPT for some domains (e.g., CORD-19 and ACL PAPERS), while it closely approaches DENSE-DAPT for others (e.g., GUTENBERG). Overall, the parameters for the additional expert comprise about 10% of the total parameters in the DEMIX model, and DENSE-DAPT involves updating all the parameters of the model towards in the target domain, so we

Domains	# Experts	Parameters per GPU 125M	350M	760M	1.3B
TRAINING	8	17.8	14.7	13.9	**13.4**
	16	**17.7**	**14.6**	**13.7**	**13.4**
NOVEL	8	21.4	18.3	17.4	17.0
	16	**16.0**	**14.0**	**13.5**	**12.5**

Table 4.6: Average perplexity in training and novel domains before and after adding 8 experts adapted to the novel domains (via DEMIX-DAPT). Adding experts reduces perplexity on all domains, even those previously seen.

would expect that DENSE-DAPT outperforms DEMIX-DAPT in some cases. The strong performance of DEMIX-DAPT on domains like CORD-19 and ACL PAPERS suggests that DEMIX-DAPT is especially helpful when the target domain strongly aligns with one of the experts (Figure 4.4).

Adding eight experts With expert mixing (§4.4), newly added experts can be combined with existing ones in the model at test time. To more thoroughly understand the effect of adding more experts to the system, we add all experts adapted to novel domains to the DEMIX model from §4.3. We display the performance of a DEMIX LM with 16 experts (8 experts trained on training domains, 8 additional experts adapted to novel domains) in Table 4.6. We generally observe that DEMIX-DAPT reduces perplexity on all domains for all model sizes, again without forgetting.

Adding the eight additional experts in fact reduces perplexity on *previously seen* domains. For example, across all model sizes, on average, we see an 2.4% reduction on BIOMED, 1.8% reduction on REALNEWS, and 2% reduction on REDDIT. These improvements are small, which is expected given that we only performed DEMIX-DAPT for at most one hour with eight GPUs. Even so, these results suggest that DEMIX layers can enable the LM to incorporate knowledge from novel domains to improve its performance on previously seen domains.

4.6 Language Models with Removable Parts

Current LM pretraining datasets are rife with undesirable content, from hatespeech to extremism [Gehman et al., 2020a; Bender et al., 2021b]. Another consequence of DENSE training is that it is difficult to restrict the model's access to these problematic domains after training, as might be desirable for many user-facing

	125M Parameters per GPU		
Domain	**+EXPERT**	**–EXPERT**	**–DOMAIN**
1B	13.7	25.5	**30.4**
CS	15.7	22.4	**25.4**
LEGAL	8.9	20.9	**22.7**
BIOMED	12.4	18.6	**21.9**
WEBTEXT	20.9	**27.3**	25.4
REALNEWS	18.9	**26.7**	25.0
REDDIT	34.4	47.8	**51.3**
REVIEWS	20.5	39.0	**43.0**
Average	18.2	28.5	**30.6**

Table 4.7: In a 125M parameter model, removing a domain expert (–EXPERT) results in perplexity degradation on the corresponding domain, approaching the performance of an LM that has not been exposed to that domain (–DOMAIN). Here we bold the *worst* performing model for each domain, i.e. the one that gets the *highest* perplexity.

tasks [Xu et al., 2021; Dinan et al., 2021].

DEMIX layers offer new capabilities for lightweight control over the domains in the training data that LMs use to make predictions at inference time. In particular, since DEMIX layer experts specialize to their domain (Figure 4.2), experts that are assigned to domains that are *unwanted* at test-time can be simply disabled and unused.

A key question is whether disabling an expert can simulate a model that has not been exposed to that domain, which we study in this section. However, since the self-attention and input embedding parameters in the DEMIX LM are shared across domains, removing an expert offers no guarantee of having fully forgotten content from the removed domain. Establishing such bounds is an important avenue for future work.

4.6.1 Experimental Setup

To evaluate whether we can simulate models that have not been exposed to a particular domain, we compare three settings:

+EXPERT A DEMIX LM with all experts active.

–EXPERT A DEMIX LM with a domain expert deactivated.

–DOMAIN A DEMIX LM retrained from scratch without a particular domain. We replace the removed domain with GUTENBERG.[13]

We evaluate expert removal (+EXPERT and –EXPERT) with the DEMIX LM with 125M parameters per GPU from §4.3, with no modifications. For all baselines,we evaluate use expert mixing with a cached prior (§4.4).

4.6.2 Results

Removing a domain expert harms model performance on the associated domain, in most cases approaching the performance of a model that has not been exposed to data from that domain (Table 4.7). In some cases (e.g., WEBTEXT and REALNEWS), –EXPERT even underperforms –DOMAIN. This leads us to conjecture that most domain-specific learning happens within the DEMIX layer, despite the fact that other parts of the model are affected by all training domains.

4.7 MoE Comparison

Here we describe empirical comparisons between DEMIX and GSHARD, the token-level mixture of experts transformer proposed by Lepikhin et al. [2021]. As opposed to DEMIX, which uses domain labels to route data to experts, GSHARD learns a token-level routing mechanism during training. Each token in every other layer is sent to two of k experts, and this routing is updated via backpropagation.

As GSHARD is emblematic of an learned routing procedure, we are generally interested if GSHARD naturally learns to specialize experts to domains, whether its experts are modular, and how GSHARD LM generally performs compared to DEMIX and DENSE models on our multi-domain corpus.

Experimental Setup We aim to make minimal changes to the overall architecture of the model, to focus on the differences afforded by token-level routing (vs. DEMIX routing). As such, we keep all architecture and computational budgets the same as our DEMIX and DENSE baselines (we generally display results for the 125M, 350M, and 760M parameter LMs). We only add the GSHARD routing procedure to every other layer, which involves routing each token to the top-2 experts of that layer. This additionally necessitates

[13] Our cluster requires that jobs are allocated with eight GPUs, necessitating eight experts — hence the substitution.

		Parameters per GPU			
		125M	350M	760M	1.3B
DENSE	**GPUs**	32	64	128	128
	Total Experts	0	0	0	0
	GPUs/expert	0	0	0	0
	Total params	125M	350M	760M	1.3B
	TFLOPs/update	556	3279	13,637	23,250
	TFLOPs/GPU	**31**	**37**	45	51
DEMIX	**GPUs**	32	64	128	128
	Total Experts	8	8	8	8
	GPUs/expert	4	8	16	16
	Total params	512M	1.8B	3.8B	7.0B
	TFLOPs/update	556	3279	13,637	23,250
	TFLOPs/GPU	**31**	**37**	**48**	**55**
GSHARD	**GPUs**	32	64	128	128
	Total Experts	32	64	128	128
	GPUs/expert	1	1	1	1
	Total params	1B	6.7B	29.5B	52.5B
	TFLOPs/update	675	4120	17,400	30,000
	TFLOPs/GPU	15	16	19	13

Table 4.8: Our specifications for training DENSE, DEMIX, and GSHARD LMs. All models are trained for about 48 hours on V100 GPUs. DEMIX layers increase the total parameters of the LM while maintaining (or increasing) throughput, measured in TFLOPs/GPU. We use the formula described in Narayanan et al. [2021] to calculate these metrics.

a load balancing loss to prevent only a minority of experts from being used [Lepikhin et al., 2021]. All GSHARD experts are of the same size as our DEMIX experts, i.e., each expert is a two layer MLP with the same dimensions as the original feedforward layer of the transformer.

Model Scale In DEMIX, we always add the same number of experts as the number of training domains (in our case — eight experts), and use extra computation to increase the batch size for each expert. Our GSHARD implementation, on the other hand, allocates one expert per GPU. This means that GSHARD adds many more experts to the system, which results in a substantially larger increase in model size (Table 4.8). Unlike DEMIX, GSHARD results in an increase in FLOP count relative to the DENSE model, due to a variety of additional computation during training, like load balancing and routing to two experts for every token, which DEMIX does not need.

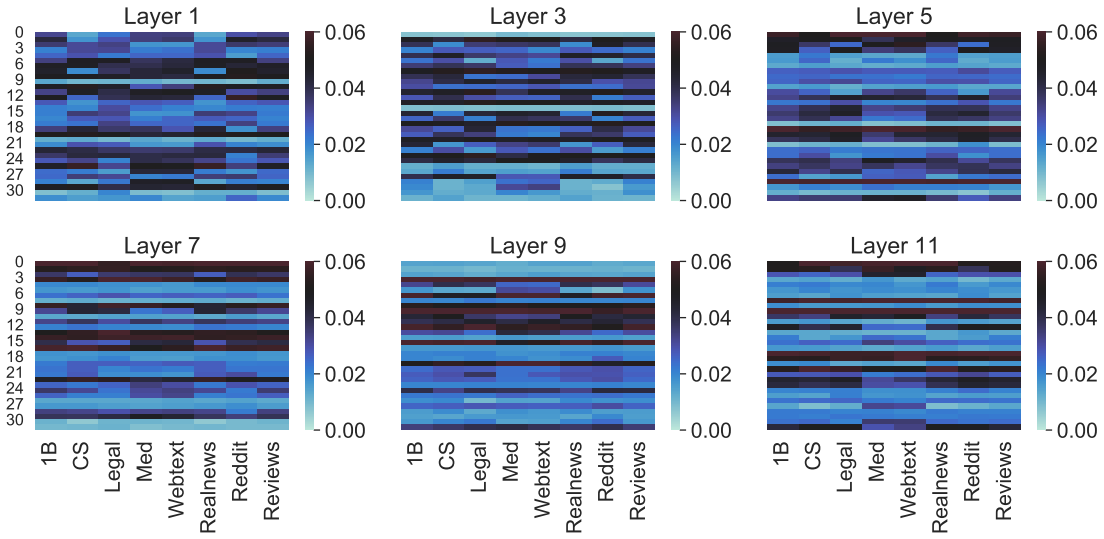

Figure 4.7: Average gating probabilities across domains (x-axis) for each expert (y-axis) in the expert layers of a GSHARD LM with 125M parameters per GPU. We observe high entropy of gating probabilities across experts and domains in each expert layer, with similar results in larger models.

Training efficiency However, unlike DEMIX, which increases model size while maintaining or improving GPU throughput, GSHARD in fact *reduces* GPU throughput during training (Table 4.8). This is due to the necessity of expensive *all-to-all* operations in GSHARD which mediate communication between experts on different GPUs that are activated for different tokens of the same document.[14] These *all-to-all* operations are bottlenecked by the quality of GPU communication channels on the cluster. We also found that additional inefficiencies are introduced via GSHARD's load balancing, since some experts are not used at test time. DEMIX has no load balancing or all-to-all communication. It uses all experts to maximum efficiency, because we simply assign GPUs to domains for our routing protocol.

Evaluation efficiency Another benefit to DEMIX is that its experts specialize to their domain, and only a sparse subset of them are activated at test time. Does token-level routing via GSHARD also result in a modular model? We explore this question by computing the average gating probabilities in the GSHARD router across all experts for all test data in each domain. We generally find that gating probabilities in GSHARD have high entropy across experts regardless of domain, suggesting that the token-level routing procedure does not in fact result in modularity out-of-the-box and all experts are needed for all input texts (Figure 4.7). As we

[14] https://images.nvidia.com/events/sc15/pdfs/NCCL-Woolley.pdf

	Parameters per GPU			
	125M	350M	760M	1.3B
DENSE (balanced)	19.9	15.8	14.3	13.6
DEMIX	**17.8**	**14.7**	**13.9**	13.4
GSHARD	17.2	14.3	14.2	**12.7**

Table 4.9: Average in-domain test-set perplexity across the 8 domains in the training data. We discuss the last row in §4.4.4.

	Parameters per GPU			
	125M	350M	760M	1.3B
DENSE (balanced)	25.9	21.4	18.4	17.8
DEMIX	**21.4**	**18.3**	**17.4**	**17.0**
GSHARD	24.0	19.5	18.9	17.2

Table 4.10: Average test perplexity on novel domains. Mixing domain experts with a prior estimated using a small amount of data in the target domain outperforms all other baselines.

increase computational budget, this issue is exacerbated; we need 128 GPUs to evaluate on the test data for the final model. Whereas with DEMIX, we only need 8 GPUs to compute the domain posterior on a subset of the validation data. Moreover, because the domain posterior is usually sparse, one can use an even smaller number of GPUs for evaluating on test data, loading only those experts with non-zero probabilities.

Model performance As noted earlier, our GShard implementation substantially increases the effective parameter count of the model relative to DEMIX (Table 4.8). While this expansion of model size by GShard translates to better in-domain performance than DEMix for the 32 and 64 GPU settings, we observe the DEMix LMs consistently outperform GShard on the novel domains regardless of computational budget (Table 4.10). Surprisingly, GSHARD underperforms DEMIX even in-domain for the 760M parameter model (Table 4.9), despite being 4x larger in effective parameter count (Table 4.8). This suggests that domain-modularity is an important mechanism to improve model generalization, in addition to model size. We believe there is a rich area of future work to investigate how to combine token- and domain-level routing, to realize the benefits of increasing parameter count while maintaining domain modularity at scale.

Summary Our results suggest that while GSHARD is an effective method for substantially increasing model size under a fixed budget, it comes with large costs to training and evaluation efficiency, does not result

in a modular LM. The lack of modularity also implies that GSHARD suffers from similar downstream issues as DENSE models, e.g., forgetting after adaptation and lack of lightweight controllability, though we leave a close exploration of those phenomena to future work. Overall, DEMIX LMs are substantially simpler and more efficient for training and evaluation, and even outperform GSHARD (especially out of domain) despite being substantially smaller, suggesting the importance of domain modularity as an alternative mechanism to model scaling for improving generalization in LMs.

4.8 Related Work

Incorporating Metadata Document metadata has been commonly used to improve the quality of topic models [Mimno and McCallum, 2012; Ramage et al., 2009; Zhu et al., 2012], and previous works have used metadata for adapting RNN-based language models [Jaech and Ostendorf, 2018] or learning better document representations [Card et al., 2018]. Zellers et al. [2019b] and Keskar et al. [2019] prepend document metadata in the input text (similar to our +DOMAIN-TOKEN setting) while training transformer LMs to provide better inference-time control of text generation.

Inference-time Control DEMIX layers provide a simple mechanism for inference-time control of language model behavior. Previously proposed methods for inference-time control are either expensive to use [Dathathri et al., 2020], or rely on densely trained models [e.g., Keskar et al., 2019]. Liu et al. [2021] use multiple experts for inference-time text generation control. This method may be applied to DEMIX layers to steer text generation with experts trained on different domains.

Multilinguality Related to variation across domains is crosslingual variation. Past work has suggested that multilingual models benefit from language-specific parameters [Fan et al., 2020; Pfeiffer et al., 2020; Chau et al., 2020]. Here, we investigate the effect of incorporating *domain*-specific parameters into the LM. Though the boundaries between languages are (often) more clear than those among domains, DEMIX layers draw inspiration from multilingual research, and future work might explore a compositional approach with both language experts and domain experts.

Continual Learning DEMIX-DAPT is a type of continual learning, in which the model learns incrementally on new data [Chen et al., 2018]. Previously proposed techniques to support continual learning include regularization [Kirkpatrick et al., 2017], meta-learning [Munkhdalai and Yu, 2017], episodic memory modules [Lopez-Paz and Ranzato, 2017; de Masson d'Autume et al., 2019], and data replay [Sun et al., 2019b], all of which may be combined with DEMIX layers. Model expansion techniques to incorporate new reinforcement learning or visual tasks [Rusu et al., 2016; Draelos et al., 2017] is especially related to DEMIX-DAPT. Our results suggest that continual learning in LMs is naturally enabled with modular domain experts; this may be further explored using temporally-relevant domains [Lazaridou et al., 2021b].

LM Adapters Also related to DEMIX-DAPT is the line of work into adapter modules for pretrained LMs [Houlsby et al., 2019; Pfeiffer et al., 2020]. Similar to the setting in which we add experts for new domains, adapter modules involve freezing the pretrained language model and updating a small number of additional parameters that are appended to certain parts of the network. This study confirms previous findings that only a subset of LM parameters need to be fine-tuned to a target dataset [Ben Zaken et al., 2022]. Expert addition may be performed with adapter modules to further improve efficiency.

Multi-Domain Models Multi-domain models have been studied extensively in the context of machine translation, first with statistical systems [Banerjee et al., 2010; Sennrich et al., 2013], and more recently with neural networks [Pham et al., 2021]. Other works have explored multi-domain settings with smaller models and explicit domain labels, using supervision [e.g., Wright and Augenstein, 2020; Guo et al., 2018; Zeng et al., 2018] or dense training [e.g., Maronikolakis and Schütze, 2021]. Previous studies have shown the importance considering domains when adapting LMs [Ramponi and Plank, 2020; Gururangan et al., 2020]. Our study establishes the importance of considering domains when training LMs from scratch.

4.9 Conclusion

We introduce DEMIX layers for language models, which provide modularity at inference time, addressing limitations of dense training by providing a rapidly adaptable system. DEMIX layers experts can be mixed to handle heterogeneous or unseen domains, added to iteratively incorporate new domains, and removed to

restrict unwanted domains.

There are many exciting directions for future work, in addition to those described throughout the paper. They include combining domain and token-level routing, to realize the benefits of modularity while scaling models efficiently. The design of DEMIX layers assumes access to coarse provenance labels (or other metadata) to identify domains in pretraining data; an alternative option is to use unsupervised learning to discover domains in the corpus, which, in concert with domain metadata, may lead to better DEMIX expert assignments. Furthermore, in this work, we study DEMIX layers with a dataset that has a few large domains. In practice, textual domains usually contain many diverse subdomains of varying prevalence. Training DEMIX layers on a dataset with a long tail of domains may require automatic measures to cluster smaller domains, or hierarchical experts that are specialized to progressively narrower data distributions.

4.10 Limitations

While DEMIX offers new opportunities to reduce the influence of unwanted training domains (e.g., those that contain hatespeech) at inference time, shared parameters in the LM may prevent the model from fully forgetting the unwanted domain after expert removal. Therefore, DEMIX LMs may still be prone to producing harmful generations when deployed, and further research is required to understand the bounds on the probability of toxic degeneration after expert removal.

While we partially anonymize our corpus with simple regexes, it is difficult to guarantee that sensitive information is not exposed in large datasets. To protect data authors and subjects, we do not publicly release our models or data, although we provide instructions and code to replicate them to support reproducibility.

Chapter 5

Building a Decentralized Language Model

DEMix enables adaptive and customizable language models through conditional computation, where subsets of parameters in the Transformer are updated with respect to a distinct domain of the training corpus and ensembled during inference. However, the design of DEMix has two primary limitations.

First, DEMix still requires synchronization across all *shared* parameters (e.g., self-attention layers, embedding layers) in the model. When scaled to thousands of GPUs [Touvron et al., 2023], this synchronization incurs an extreme cost and leads to a centralization of model development, exacerbating the data selection issues we outline in Chapter 2. Second, DEMix relies on document metadata to identify domains, and such supervision is not always available [e.g., in large Internet crawls; Raffel et al., 2019; Rae et al., 2021b; Gao et al., 2021]. Moreover, the optimal number of metadata-based domains for a fixed budget is unknown, since metadata domains cannot be easily merged or divided.

In this chapter, we introduce Cluster-Branch-Train-Merge (C-BTM; Figure 5.1), a metadata-free algorithm to train a fully decentralized language model. Our method does not require the massive multi-node synchronization necessary for both dense and DEMix training. We develop a new class of LMs that is instead embarrassingly parallel: different parts of the model are independently trained on different subsets of the data, with no need for multi-node training or inference.

Our new ELMFOREST[1] models consist of a set of **EXPERT LMs** (ELMs), each specialized to a distinct domain in the training corpus, e.g., scientific or legal text. The ELMs are each independently functional LMs

[1] **E**xpert **L**anguage **M**odels **For** **E**fficient **S**parse **T**raining

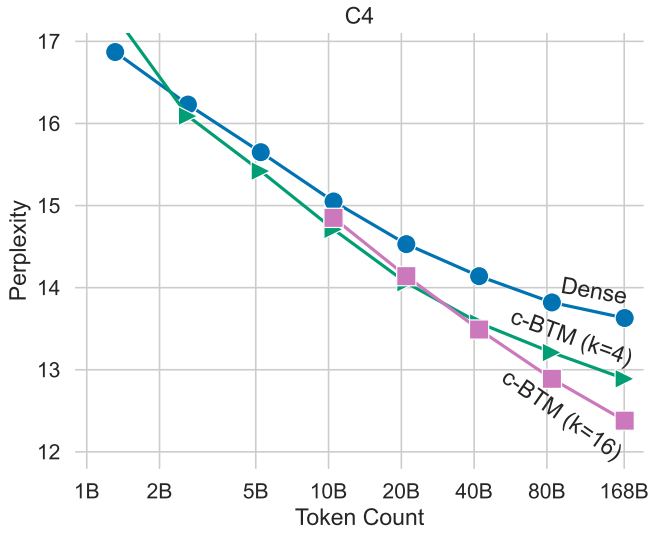

Figure 5.1: We present C-BTM, a new technique to asynchronously scale expert LMs (§5.1). C-BTM splits a corpus into k clusters, trains an expert LM on each cluster, and creates a sparse ensemble during inference. Above, LMs trained with C-BTM (with 4 or 16 clusters) achieve lower validation perplexity than compute-matched dense LMs. These LMs begin with OPT-1.3B [Zhang et al., 2022], and are further trained on C4 [Raffel et al., 2019]. The optimal cluster count for C-BTM, and its performance gains, increase with the size of training data (shown in log-scale).

with *no shared parameters*, unlike DEMix models that only specialize the Transformer feedforward layers. ELMs can be added and removed to the model at any time to update data coverage, ensembled to generalize to new domains.

In the C-BTM algorithm, e first use unsupervised clustering to discover domains in a corpus, and train an ELM on each cluster independently (§5.1.1). At inference time, we *sparsely* activate a subset of the trained ELMs (§5.1.2). We ensemble ELMs by weighting their outputs with the distances between an embedding of the current context and each expert's cluster center. This enables simple and efficient sparse computation [Fedus et al., 2022a] by activating only the top-k experts when predicting each new token.

C-BTM allows for fine-grained control over the number and size of data clusters, since they are automatically learned without being constrained by available metadata. We use this new capability to investigate the scaling properties of C-BTM as a function of the number of experts trained, controlling for a variety of factors (§5.2). Extensive experiments show that training more clusters always results in better validation perplexity than single cluster (i.e., dense) models, and the optimal cluster count increases with the overall compute (§5.3.1). These results are consistent for both 1.3B and 6.7B parameter experts.

With more clusters, we can aggressively parallelize expert training: for example, we train 128 ELMs (168B parameters in total) on 168B tokens of text in aggregate with only 8 GPUs at a time. This enables us to avoid many practical difficulties associated with training large LMs across many nodes simultaneously (§5.3.2). Moreover, the number of parameters at inference time can be kept constant even as the number of experts grows (§5.3.3): using just the top-2 or top-4 experts is comparable to using all experts, while using just the top-1 expert still outperforms the dense model. Training with more clusters is also more effective than training larger dense models: in §5.3.4, we demonstrate that training many 1.3B expert LMs, and sparsifying them to a 5.2B parameter LM by combining the top-4 experts, achieves the same perplexity as a 6.7B dense model, but with only 29% as many training FLOPs. These gains are also reflected in few-shot text classification experiments (§5.4), which show that C-BTM models outperform dense baselines even with heavily sparsified inference.

C-BTM provides a radically simplified sparse modeling approach that eliminates nearly all communication overhead from existing sparse LM schemes. Existing sparse LMs typically route different tokens to specialist parameters [Lepikhin et al., 2021; Fedus et al., 2022b; Clark et al., 2022]. However, they have yet to be widely adopted, perhaps due in part to the communication costs of routing each token in each sparse layer [Artetxe et al., 2021], challenges in learning to specialize experts to tokens [Zhou et al., 2022], and the necessity of additional mechanisms to balance expert utilization [Lewis et al., 2021b]. C-BTM improves over sparse LMs by routing sequences (instead of tokens) using offline balanced clustering (instead of online load balancing) with no shared parameters between experts. We compare directly to a mixture-of-experts model with top-2 routing [Lepikhin et al., 2021] in §5.5.

Our final analysis (§5.6) shows that balanced clustering is key to C-BTM performance; it works as well as expert assignment with gold metadata, and strongly outperforms random and unbalanced clustering baselines. Overall, our findings suggest that C-BTM is an efficient and accessible method to scale large language models into massive datasets.

These results provide compelling evidence for the promise of scaling large language models with many smaller, independently trained ELMs. We envision that this work lays the foundation for democratized model development at inclusive compute budgets — that groups with different resource constraints and research interests may combine efforts to build open-sourced, community-authored large language models, comprised

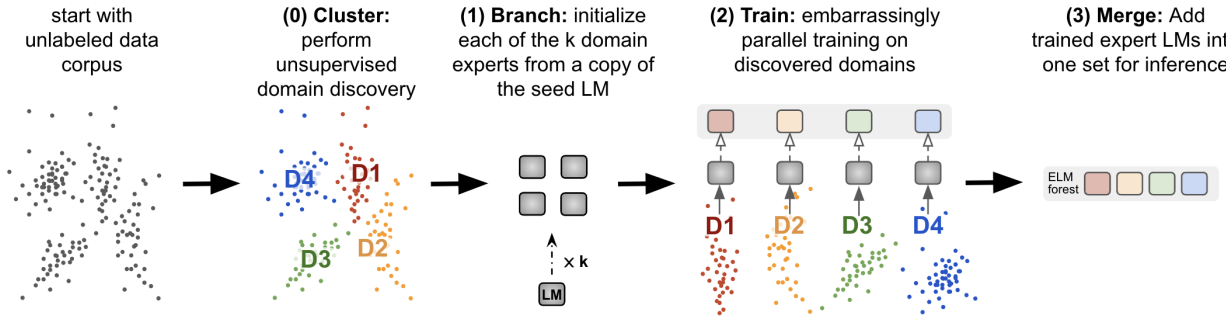

Figure 5.2: C-BTM training process (§5.1.1). C-BTM begins with unsupervised domain discovery using k-means clustering. We then initialize expert language models (ELMs) with a seed language model (e.g., OPT; Zhang et al. 2022) and train an ELM on each cluster. The resulting experts are added to a larger collection for sparse inference.

of continually-evolving repositories of ELMs.

We release our code and models publicly.[2] This chapter is based on work from:

Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A. Smith, Luke Zettlemoyer. 2023. Scaling Expert Language Models with Unsupervised Domain Discovery. ArXiv, abs/2303.14177.

5.1 C-BTM

We introduce C-BTM, a method for embarrassingly parallel training that specializes expert language models to domains discovered through clustering instead of metadata. C-BTM enables scaling to arbitrary numbers of domains and compute budgets on any corpus. In this section, we outline C-BTM training (Figure 5.2) and inference (Figure 5.3).

5.1.1 Training

Step 0: Cluster To segment our corpus, we employ k-means clustering, enforcing balanced clusters. ELMs trained without this constraint perform worse (§5.6.2).[3]

Consider the iterative, hard expectation-maximization view of k-means clustering. In the expectation step, each document embedding is assigned to a cluster center based on its Euclidean distance to each center. In the

[2] https://github.com/kernelmachine/cbtm

[3] Other techniques to improve clusters, e.g. k-means++ [Arthur and Vassilvitskii, 2007], can be used to improve performance.

maximization step, each cluster center is updated to be the mean embedding of the current set of documents assigned to it. To balance the clusters, we formulate the expectation step as a balanced linear assignment problem [Malinen and Fränti, 2014]. Given D document embeddings with representations $\{w_1, \ldots, w_D\}$ and K cluster centers with representations $\{h_1, \ldots, h_K\}$, we assign each document d to a cluster with the assignment index $a_d \in \{0, \ldots, K\}$:

$$\min_{a_1,\ldots,a_D} \sum_{d=1}^{D} dist(h_{a_d}, w_d) \text{ s.t. } \forall k, \sum_{d=1}^{D} \mathbb{1}_{a_d=k} = \frac{D}{K} \tag{5.1}$$

where $dist$ is the Euclidean distance. Many algorithms exist to solve this problem; we follow Lewis et al. [2021b] and use the auction algorithm [Bertsekas, 1992]. We only use balancing when estimating the cluster centers; we use greedy inference when predicting clusters.

In our experiments, we use a simple tf-idf embedding function, which is highly efficient at scale and leads to interpretable clusters.[4] We only use a single shard of each corpus to train our clustering model. Any new document, once embedded, can be efficiently mapped to its nearest cluster(s) without additional training. Any embedding function can be used, though the choice of embedder may apply different assumptions of what constitutes a textual domain and come with efficiency trade-offs.[5] Comparing to other embedding or clustering methods is an interesting area for future work, and could likely improve performance.

Step 1: Branch (from seed LM) To begin training experts on each of the k clusters from Step 0, we first *branch* from (i.e., make k copies of) a *seed* LM. Seed LMs are critical for the overall functionality of ELMs, and ELMs perform best when the seed LM has been trained with a diverse corpus [Li et al., 2022]. In our experiments, we use an OPT LM [Zhang et al., 2022] as our seed.[6]

Step 2: Train We assign each ELM to a single cluster, and train on each cluster with the log likelihood objective.

[4] In initial experiments, tf-idf outperformed other scalable text embeddings, like hash embeddings [Svenstrup et al., 2017].

[5] tf-idf assumes that domains are lexically-driven, which may not correspond with other notions of domain.

[6] Li et al. 2022 find that dedicating more compute to branching (rather than seed training) leads to better in-domain performance, and the choice of seed LM has a strong effect on the modularity of the resulting ELMs. Future work may explore the effect of different seed LMs on C-BTM performance.

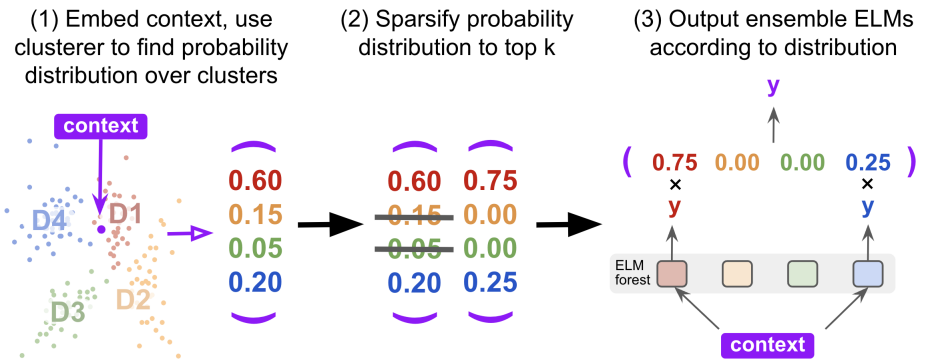

Figure 5.3: C-BTM inference process (§5.1.2). At inference time, we embed each incoming context and estimate a probability distribution over clusters, by calculating the distance between the embedded context and each cluster center. We use this probability distribution, optionally sparsified to use only the top-k experts, to weight an output ensemble of the ELMs.

Step 3: Merge After training on the assigned domain, we add the new ELM into a larger collection for inference.

In this work, we focus on a single iteration of C-BTM for simplicity. Future work may explore branching from already trained experts in multiple iterations.

5.1.2 Inference

At inference time, we use a sparse ensemble of the outputs of ELMs for incoming test contexts (Figure 5.3). Formally, consider that the language model provides, at each timestep, $p(X_t \mid \boldsymbol{x}_{<t})$. We introduce a domain variable D, alongside each sequence. Then the next-step conditional distribution on the history $\boldsymbol{x}_{<t}$ is:

$$p(X_t \mid \boldsymbol{x}_{<t}) = \sum_{j=1}^{k} p(X_t \mid \boldsymbol{x}_{<t}, D = j) \cdot \underbrace{p(D = j \mid \boldsymbol{x}_{<t})}_{\text{ensemble weights}} \tag{5.2}$$

With the pretrained embedder and clustering model from Step 0 (§5.1.1), we embed the context $h_{x_{<t}}$ and use the k cluster centers $\{h_{c_0} \ldots h_{c_k}\}$. We set ensemble weights as:

$$p(D = j \mid \boldsymbol{x}_{<t}) \propto \text{topk}[\exp(-dist(h_{x_{<t}}, h_{c_j})^2 / T)] \tag{5.3}$$

Where $dist$ is the Euclidean distance, T is a temperature parameter which sharpens or smoothes the probability distribution over cluster centers, and the top-k function filters for the top-k probabilities and

renormalizes the distribution to sum to 1. This formulation is reminiscent of nearest-neighbor retrieval mechanisms for language models [Khandelwal et al., 2019; Shi et al., 2022].

These ensemble weights are updated for every incoming token, although in separate experiments we observe that we find that cluster assignments (and in effect, ensemble weights) can be fixed for the second half of a document with no drop in performance; this can further speedup inference.

We find that, in practice, the performance of our models is robust to even top-2 or top-4 experts, meaning that the inference costs of the language model are equivalent to a much smaller LM. We perform an empirical study of inference variations in §5.3.3.

5.1.3 Comparing to Dense Training

Dense LMs are typically trained using hundreds or thousands of concurrent GPUs, all of which synchronize gradients each update. For example, OPT-175B [Zhang et al., 2022] was trained on 992 80GB A100 GPUs, and PaLM-540B [Chowdhery et al., 2022] was trained on 6144 TPU v4 chips. C-BTM improves training efficiency by reducing communication overhead, as only GPUs training the same ELM must communicate. Furthermore, the chance of a GPU failure can grow considerably with the number of GPUs. C-BTM improves the resiliency of distributed training, since a single GPU failure only delays training for a single ELM, whereas in dense training, a single GPU failure afflicts training on all other GPUs. C-BTM also makes training large LMs more feasible on shared GPU clusters, since it effectively decomposes training into smaller jobs which can run asynchronously. This makes job scheduling more efficient by reducing the number of GPUs that need to be allocated simultaneously.

5.1.4 Comparing to DEMix (Chapter 4)

Our method addresses several limitations of the training and inference techniques of DEMix.

First, DEMix is limited to training data with metadata which can be used to determine its domains. Typical LM corpora, including C4 [Raffel et al., 2019] and the Pile [Gao et al., 2021], are sourced from the Internet without retaining document provenance at collection time, and are infeasible to label manually. Also, the optimal number of experts for a fixed corpus size, model architecture, and budget remains unknown, and is difficult to explore with metadata-based domains, since they cannot be easily merged or divided. C-BTM

broadens the applicability of DEMix to arbitrary datasets.

Moreover, DEMix inference follows the *cached prior* method, where the ensemble weights are estimated using Bayes' rule on additional held out data, and the prior $P(D = j)$ is estimated with an exponential moving average over sequences of posterior estimates that require forward passes on experts. This estimate is then fixed during test data evaluation.

With C-BTM, we route based only on the current context. Thus, no additional data or forward passes through the experts are needed to estimate ensemble weights, nor do we need to assume that adjacent documents in the test set come from the same distribution.

Finally, similar to dense training, DEMix is afflicted by the synchronization costs , because it still has shared parameters across nodes (e.g., self-attention and embedding layers). With C-BTM, we have no shared parameters across experts specialized to different clusters.

5.1.5 Comparing to Mixture-of-Experts (MoE)

Like MoE models [e.g., Fedus et al., 2022a], C-BTM allows for efficient scaling of large LMs while keeping inference costs manageable. However, C-BTM routes sequences (instead of tokens) using offline balanced clustering (instead of online load balancing) with no shared parameters between experts. This eliminates effectively all complexities associated with balancing expert utilization [Lewis et al., 2021b], avoids expensive all-to-all operations between experts [Artetxe et al., 2021], and naturally leads to interpretable expert specialization to domains of the training corpus. In §5.5, we compare directly to MoE baselines trained with sparse upcycling [Komatsuzaki et al., 2022], which initializes the MoE with a dense checkpoint, mirroring how C-BTM initializes ELMs.

5.2 Experimental Setup

We design a set of experiments to study C-BTM on two large corpora (Figure 5.4) selected to be distinct from the corpus used to train our seed OPT model, and report perplexity on held out data from each corpus.

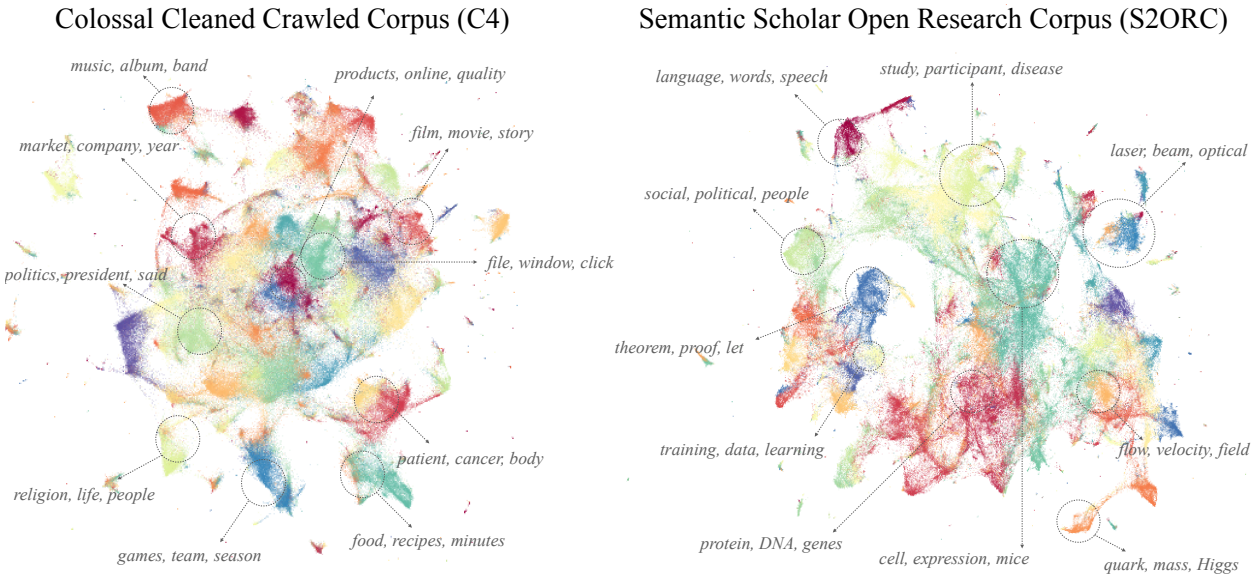

Figure 5.4: We train and evaluate on two large text corpora (§5.2.1). C4 (left; Raffel et al. 2019) and S2ORC (right; Lo et al. 2019) are diverse and contain many different clusters of text, indicated by these UMAP visualizations of 400K random documents in each corpus, colored with 32 automatically discovered and annotated clusters. See §5.6.3 for the description of our clustering and annotation procedure.

5.2.1 Data

C4 [Raffel et al., 2019] C4 is a publicly available distribution of a Common Crawl snapshot on Huggingface datasets.[7] We use the *no blocklist* version (`en.noblocklist`) to train on a dataset that is out of distribution to our seed (OPT) pretraining corpus. C4 consists of 393M documents totaling 220B BPE tokens. We train on up to 168B tokens.

S2ORC [Lo et al., 2019] The Semantic Scholar Research Open Corpus (S2ORC) is a publicly available corpus of full-text academic papers from the Semantic Scholar.[8] The corpus spans 20 fields of study (e.g., Biology, Computer Science, Art), and contains 16M documents, totaling 87B BPE tokens. We train on up to 168B tokens over multiple epochs.[9]

[7] https://huggingface.co/datasets/c4

[8] https://allenai.org/data/s2orc

[9] We do not observe overfitting in any of our experiments, consistent with other studies that train LMs for multiple epochs [Taylor et al., 2022; Muennighoff et al., 2023].

Evaluation data For all experiments, we report language modeling perplexity on 200 randomly-sampled held out documents. Because S2ORC does not come with pre-defined validation data, we create a validation corpus by sampling an equal number of documents from each field of study.

5.2.2 Experimental Setup

Clustering the data We segment each corpus using balanced k-means clustering for $k \in \{2, 4, 8, 16, 32, 64, 128\}$ (§5.1.1). To train the clustering models, we first embed all data with a tf-idf vectorizer using scikit-learn,[10] with minimal assumptions: we only remove stop-words from a fixed lexicon and replace numbers with a dummy token. We then reduce the dimensionality of the resulting embeddings; we perform truncated SVD with 100 dimensions, then normalize the vector by removing its mean and scaling to unit variance, which we observed in initial experiments improved the clustering quality. Finally, these representations are clustered using a custom Pytorch implementation.[11] We present learned clusters and visualizations in Figure 5.4. We use a single shard of each training corpus (384K documents for C4, 155K documents for S2ORC) to train the clustering model and its embedder. No evaluation data is used in this process.

Seed LM As LMs trained on diverse corpora make for better seeds [Li et al., 2022], we use pretrained OPT language models [Zhang et al., 2022] as our seed for all experiments.

Model hyperparameters We use the OPT architecture implemented in Metaseq [Zhang et al., 2022]. We use OPT-1.3B for the initial set of experiments, and replicate our experiments with OPT-6.7B. Following Zhang et al. 2022, we use the GPT-2 vocabulary of 50,257 BPE types [Radford et al., 2019b], and train with 2,048-token sequences, across document boundaries. We prepend a beginning-of-document token to each document. We set dropout to 0.1 for all parameters except those of the embedding layer.

Training hyperparameters For all models, we fix the learning rate to that used during OPT pretraining (2e-4 for 1.3B parameter models; 1.2e-4 for 6.7B parameter models; Zhang et al. 2022) using a linear decay learning rate schedule to zero (with no warmup), which we found to work well for most settings after a grid search of fastest learning rates that avoided divergence. We use a batch size of 8 for each GPU, and train

[10] https://scikit-learn.org/
[11] https://github.com/kernelmachine/balanced-kmeans

with `fp16` and fully-sharded data-parallel [Artetxe et al., 2021]. We train on NVIDIA V100 32GB GPUs. All models are trained with Metaseq [Zhang et al., 2022]. For a given number of clusters k and total GPU budget n, each ELM is allocated n/k GPUs, keeping the total effective number of FLOPs fixed across models exposed to the same number of tokens. See §A.4.2 for more details.

Scaling We train for a total of 10K steps in each run; to expose the model to more tokens, we increase the total GPU budget proportionally, up to 64 GPUs. We simulate larger budgets, up to 1024 GPUs, by increasing gradient accumulation steps with 64 GPUs. This method of scaling increases the model's effective batch size for the same number of steps, and maintains near constant run-times across our many experiments. This experimental setup also means that as the number of clusters increases, the overall set of ELMs is exposed to more data with less simultaneous computation among GPUs.

Other ways of training on more data (e.g., by keeping total batch size fixed and increasing step count) may yield different results. The best batch size and learning rate combinations for training language models are likely specific to a variety factors, including the model size, dataset, and total compute available [Shallue et al., 2018; McCandlish et al., 2018; Yang et al., 2021]. In preliminary experiments, we found that expert models benefit from faster learning rates and larger batch sizes. Given a sufficiently large batch size, experts are robust to a variety of learning rates. Our larger budget experiments might benefit from higher learning rates, but we leave further tuning for future work.

Inference One of the key hyperparameters for inference is the temperature T (Equation 3), which governs the sharpness of the probability distribution over experts for a given context. We find that setting T=0.1 works well for most settings (see §A.4.6 for more details). We also compute the nearest cluster centers for every incoming context, regardless of how stable the cluster assignments already are for a document. However, we find that these assignments can be fixed for the second half of a document with no drop in perplexity; this can further speedup inference. The other important hyperparameter is the top-k value, which sparsifies the probability distribution over experts. For our core experiments in §5.3.1, we set top-k to the total number of experts we have trained for each model. We explore the effect of enforcing sparsity with lower top-k values in §5.3.3.

Baselines In our primary experiments (§5.3), we compare with a strong dense baseline (i.e., our 1-cluster model) following OPT pretraining. We also progressively increase the number of clusters we train with for a fixed number of tokens. In subsequent experiments (§5.5), we compare to MoE language models initialized from a dense checkpoint.

5.2.3 Making Fair Model Comparisons

We follow the recommendations of Dehghani et al. [2021] and report results with multiple cost metrics, and detail our choices here. When comparing model training budgets, we are primarily concerned with the true monetary cost of model training, which is typically billed in direct proportion to GPU-time. Model inference comparisons have two main considerations: monetary cost incurred by the model deployer, again measured in GPU-time, and latency for end-users, or wall-clock time (i.e., how slow a model inference is for an end-user).

We explicitly do *not* compare or match the number of model parameters during training, which has minimal bearing on the cost of model training separately from its influence on GPU-time. The number of training parameters is a particularly misleading cost measure that is unsuitable for sparse models, since they can maintain the FLOPs and inference speed of dense models despite training many more parameters [Dehghani et al., 2021].

Training GPU-time Assuming fixed hardware, GPU-time during model training is determined mostly by FLOPs and inter-machine communications. However, prior work typically only FLOP-matches, ignoring the additional inter-GPU communications incurred by some models (e.g., MoE) that increase training costs. Ideally, our comparisons could directly fix GPU-time. This is challenging in practice, as even identical computations on the same GPU node at different times can vary wildly in speed due factors like temperature, other activity on the node, or the quality of GPU interconnect. To maintain consistency and fairness despite these confounds, our results compare FLOP-matched models with the same training data budget over the same number of updates (§5.3.1), but also report the speed of training for each FLOP-matched model (§5.3.2). This allows us to disentangle and accurately reflect multiple cost metrics of training. Since, in our experiments, models exposed to the same number of tokens incur the same number of FLOPs, we use training data size as a more interpretable measurement of the overall training budget (see §A.4.2 for more details).

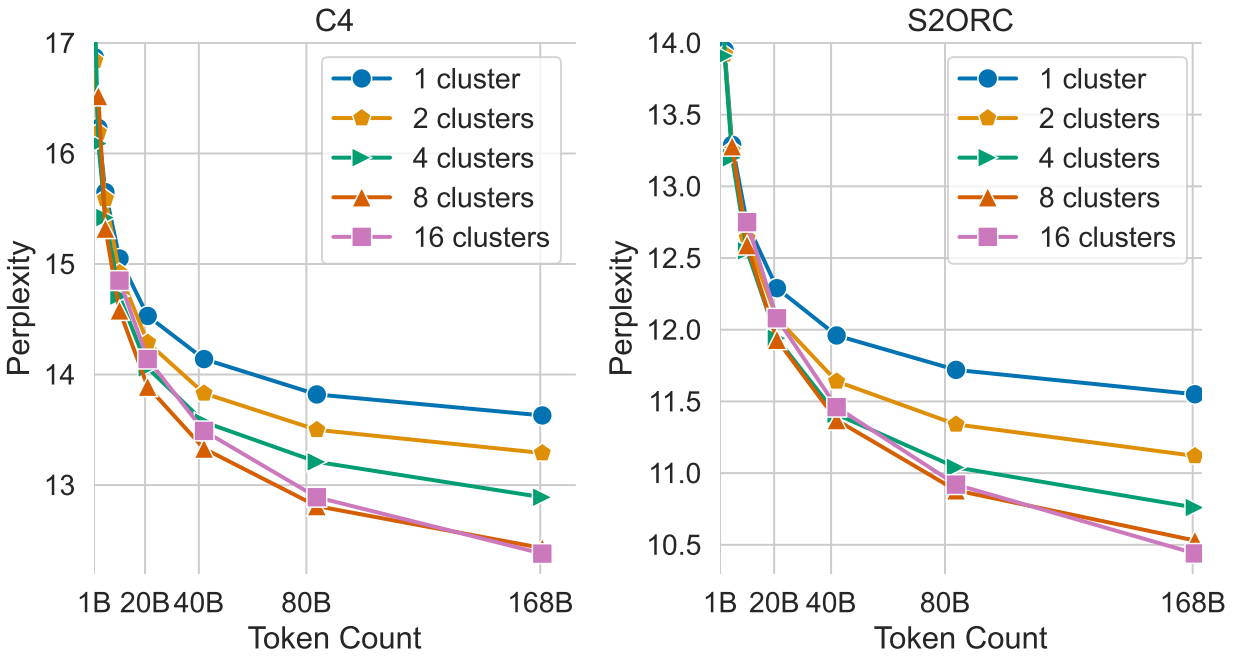

Figure 5.5: Increasing cluster count in C-BTM improves language modeling performance for a fixed compute budget (§5.3.1). Performance of ELMs trained with C-BTM as a function of the total number of tokens trained on, which, in our experiments, equalizes FLOP count. Training with more than one cluster always outperforms the compute-matched, single cluster dense model, and we observe improving performance (and in §5.3.2, faster updates) as we increase the number of clusters.

Inference GPU-time Inference GPU-time is also primarily the result of FLOPs and communication costs. Since communication during inference is minimal, we compare FLOPs via inference parameters (§5.3.3). We do not account for the FLOPs of the C-BTM router, which varies based on the clustering approach, and in our experiments are relatively negligible.

Inference latency FLOPs is not an ideal metric for inference latency of our models, because C-BTM allows for parallel inference across ELMs. This means that if ELMs share the same architecture (e.g., OPT-1.3B), inference latency is always equivalent to that of a single ELM, regardless of the number of experts active. However, inference latency may be quite different between model architectures (e.g., OPT-1.3B and OPT-6.7B); we discuss this further in §5.3.4. As with inference GPU-time, we do not consider the latency of the C-BTM router.

5.3 Language Modeling Results

We begin with a set of experiments in which we train LMs with C-BTM on datasets from §5.2.1. We are interested in measuring how perplexity changes as we increase overall compute. We first compare models against training costs: *total training tokens* (§5.3.1) and *training time* (§5.3.2). Then, in §5.3.3, we compare model performance along an axis of inference costs: the *total parameter count at inference time*. Finally, in §5.3.4 we compare model performance by fixing both training and inference costs. Across all computational budgets, C-BTM provides substantial benefits over dense training, and performance improvements increase as the total compute grows.

5.3.1 Controlling for Total Training Tokens

First, we compare model performance controlling for overall training data size (or equivalently, training FLOPs; §5.2.3). Figure 5.5 shows evaluation perplexity on C4 and S2ORC with up to 16 clusters. Training on more than one cluster always outperforms training with a single cluster (i.e., a dense model). As the amount of training data grows, the gap between our models and the dense one widens, indicating that experts make better use of larger training datasets, possibly due to their increased specialization. These results suggest that as we increase the amount of data available, C-BTM benefits from more clusters.

However, Figure 5.6 shows that there exists an optimal cluster count for each token budget that we consider. Each number of clusters has a budget range in which they are optimal, and the optimum smoothly progresses from smaller to larger cluster counts as we increase the training data size. If we increase the cluster count past the optimum, each expert has an insufficient share of the data, resulting in worse performance.

Nevertheless, we observe that using more clusters than optimal for the highest token budget settings still outperforms the dense model. Since it is cheaper to train with more clusters for a fixed training data size due to parallelism, it may be preferable in some settings to train with a large number of clusters despite their less-than-optimal performance. Based on the trends we observe at this scale, we expect that higher cluster counts would become optimal as we scale the training data size even further.

The consistency of our results on C4 and S2ORC suggests that these general trends may be widely applicable to many datasets. However, the optimal number of clusters for a given computational budget is likely dataset specific. Future work may explore relationships between dataset features and the optimal

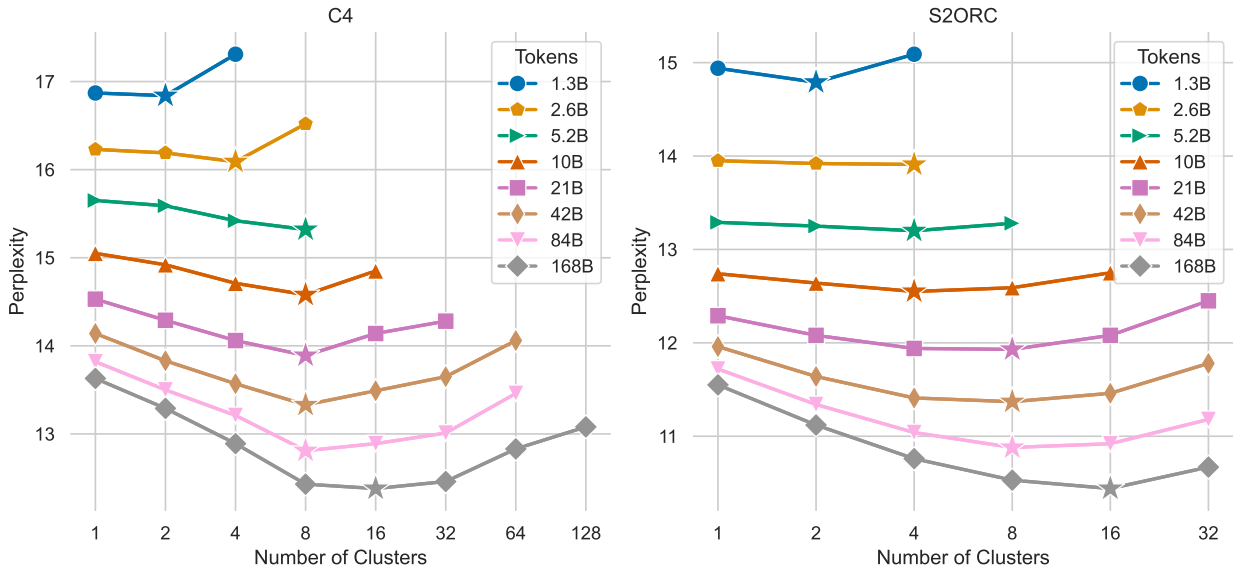

Figure 5.6: There exists an optimal cluster count for each compute budget (§5.3.1). The optimal cluster count increases as one increases the compute budget, but using too many clusters *without* sufficiently increasing compute can degrade performance. For both C4 and S2ORC, 16 clusters gets the best performance at the highest budget (168B tokens), although higher cluster counts still outperform the 1-cluster (dense) model ($x = 1$ in this graph).

cluster count.

These trends are consistent as we increase the size of our experts to 6.7B parameters (Figure 5.7), although the gaps between our baselines reduce, likely due to the substantial increase in pretraining FLOPs for OPT-6.7B.[12]

5.3.2 Comparing Training Time

Now, we turn to comparing our models based on training times. We measure the speed of training each model with the maximum seconds-per-update for each training run.[13] For C-BTM models with more than one cluster, we use the maximum seconds-per-update across all experts. To make our comparisons fair, we only compare the training times of models that have the same effective batch size (§5.2.3). Our results are displayed in Figure 5.8. As we increase the number of clusters and training data size, the update speed for C-BTM *increases*, since models with higher cluster counts use fewer GPUs per expert under a fixed budget,

[12] OPT-6.7B was pretrained for 1.83 ZFLOPs, while the OPT-1.3B was trained for 0.34 ZFLOPS.

[13] Other measures of seconds-per-update (e.g., average, median) tend to be noisy, due to factors such as dataloading and bursty GPU activity.

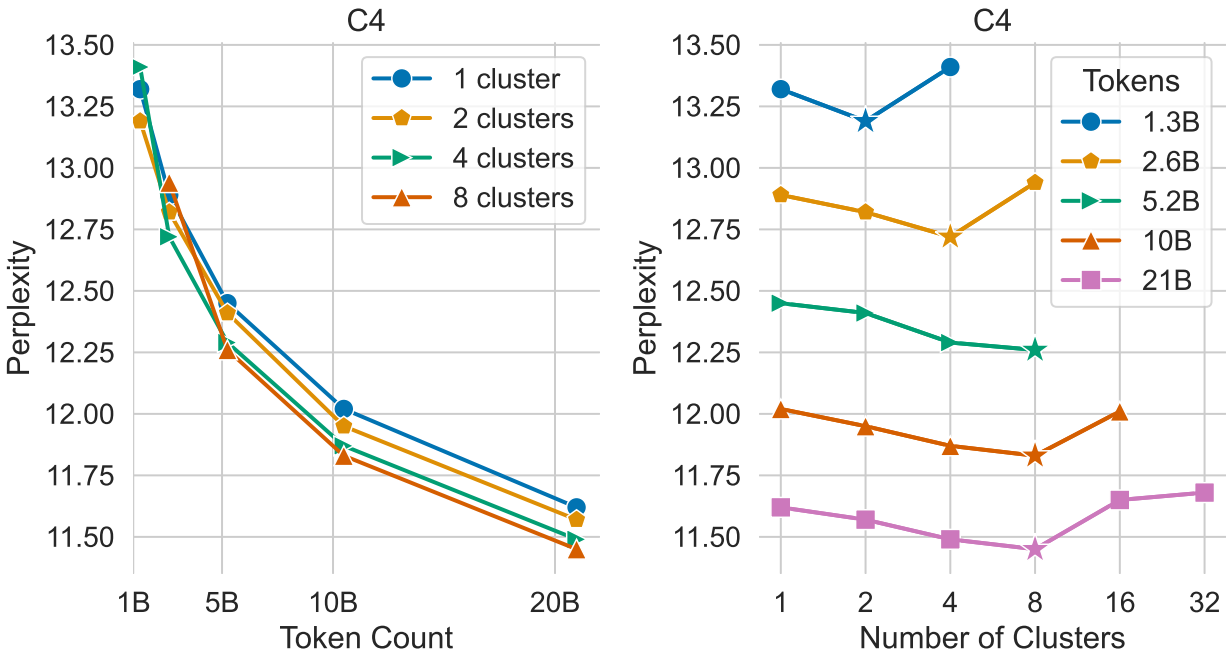

Figure 5.7: Our results are consistent even as we increase expert size to 6.7B parameters (§5.3.1). 8 clusters is optimal at 21B tokens, as it is for the 1.3B parameter ELMs. However, the gaps between these models are smaller, due to the substantial increase in pretraining FLOPs for the OPT-6.7B checkpoint.

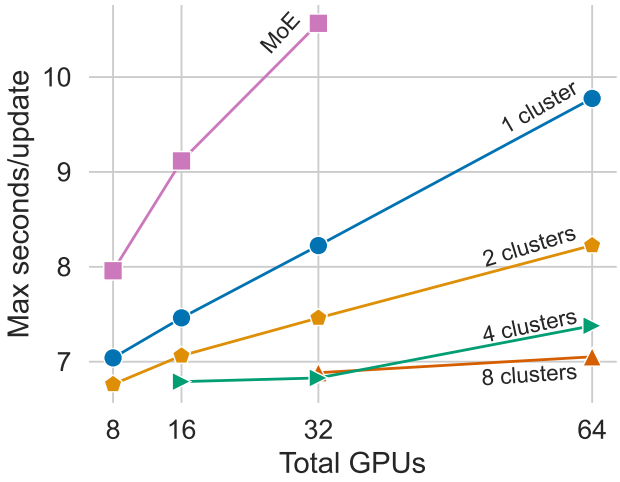

Figure 5.8: Models trained with more clusters have faster updates as we increase the total compute (§5.3.2). We display the maximum seconds-per-update for C-BTM and MoE models with varying GPU counts (across all experts). Under fixed compute, training with more clusters uses fewer GPUs per expert, and C-BTM avoids communication between experts, resulting in faster updates. On the other hand, MoE models are much slower to train, due to extensive communication between experts (§5.5.3), as well as additional FLOPs from top-2 routing [Artetxe et al., 2021].

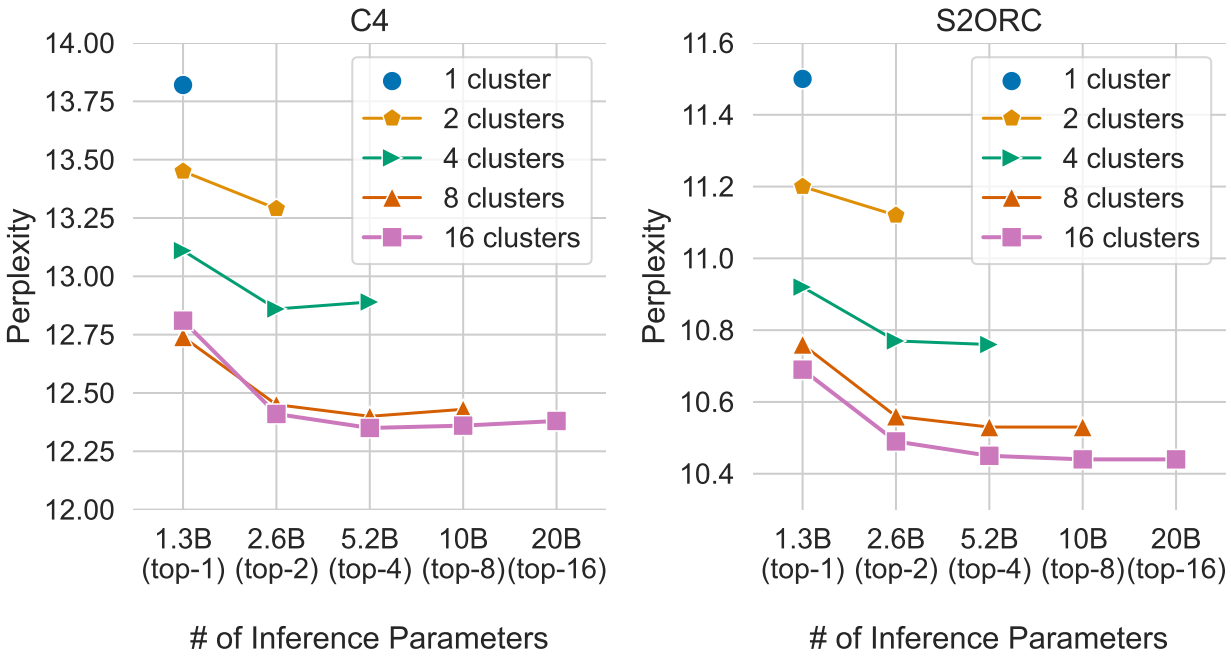

Figure 5.9: Sparse top-k inference performance at 168B token budget (§5.3.3). ELMs perform well even with heavily sparsified inference. Top-1 inference substantially outperforms the densely trained baseline at no additional inference cost, and top-2 and top-4 inference performs comparably to (and is sometimes slightly better than) activating all experts. In our setup, inference parameters are proportional to inference FLOP count (§5.2.3).

and there is no communication between experts. This suggests that C-BTM models with more clusters can be exposed to more data for in the same amount time as dense models.

As discussed in §5.1.3, C-BTM also provides important practical speedups when training large LMs at scale. C-BTM divides large compute budgets among many models, such that we can train on 168B tokens with only 8 GPUs per expert in the 128-cluster setting. On shared multi-node clusters, allocating many smaller jobs incurs shorter cumulative wait times than a single, large synchronous job, since they can make more efficient use of shared resources, and run on short-lived, idle nodes [Wortsman et al., 2022]. Furthermore, large LM training is prone to node failures, gradient spikes, and other unexpected behaviors [Zhang et al., 2022]. With dense models, when one node fails, all nodes must restart due to synchronization. With C-BTM, experts are trained independently; if a node fails, only the corresponding expert needs to be restarted, and all other experts are unaffected.

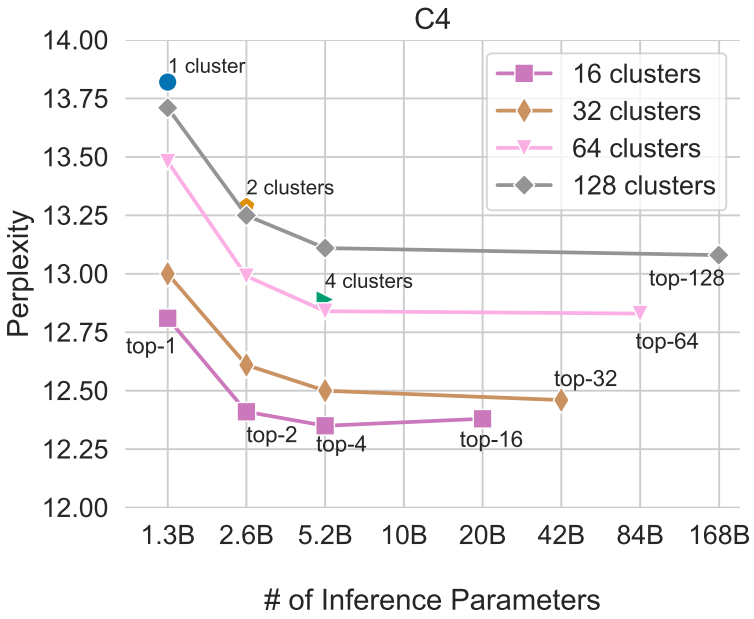

Figure 5.10: Train large, then sparsify (§5.3.3). Despite using more than the optimal cluster count for the 168B token budget, sparsifying the 32, 64, and 128-cluster models with top-1, top-2, or top-4 experts is usually better than training 1-, 2-, or 4-cluster models.

5.3.3 Controlling for Inference Costs via Parameter Count

Comparing models with just training budgets ignores the fact that C-BTM inference GPU-time costs grow as we increase the number of clusters, since we train more parameters. To mitigate these costs, we can use the top-k function (Equation 5.3) to dynamically use a subset of experts for each incoming context during evaluation (§5.1.2). Next, we study the effect of inference parameter count on model performance. We focus on the largest training budget (i.e., 168B tokens) for these experiments.

Results (Figure 5.9) show that despite training many more parameters, training C-BTM with many clusters and then using only the top-1 expert still outperforms the dense model. Further, using the top-2 or top-4 experts yields comparable performance to activating all experts. Sometimes we observe that sparsifying can even slightly *improve* performance over using all experts (for example, see the 16 cluster model for C4 in Figure 5.9). We speculate that having all experts active may introduce interference effects from experts that are specialized to clusters unrelated to test-time contexts.

Our results in Figure 5.10 suggest that sparsifying even larger expert models (i.e., those with more clusters than the optimal for a given token budget) is still highly effective. At the most extreme setting, using the

top-1 expert for the 128 cluster model (using 0.7% of total parameters at inference time for each context) still outperforms the dense model, and the top-4 expert model (3.1% of total parameters) performs comparably to using all experts.

These results suggest that C-BTM results in a highly sparse LM, and that inference costs can be kept constant even as the number of experts grows, though additional experts can be added to further boost performance.

5.3.4 Comparing to a Larger Dense Model

In our final comparison of this section, we consider both training and inference costs together. We compare a 6.7B 1-cluster (dense) model and C-BTM model with 1.3B parameter experts, which uses 16 clusters (optimal in our experiments from §5.3.1) and top-4 inference, resulting in 5.2B inference parameters. This C-BTM model has lower inference cost than the larger 6.7B parameter dense model (§5.2.3). The former uses fewer inference parameters, incurring a smaller inference GPU-time cost, and has lower latency, comparable to that of a single 1.3B-parameter ELM.

We compare the FLOPs used to train each model. Following Artetxe et al. [2021], we build continuous efficiency curves by interpolating between our empirical observations. Specifically, we calculate the speedup between our cluster expert models and dense model by interpolating between the discrete observations of perplexity values for a given empirical number of FLOPs.[14] Our goal is to identify the FLOP count necessary to achieve a particular perplexity value. If ELMs trained with C-BTM achieve the same perplexity as the dense model with half the FLOPs, we conclude that C-BTM achieves a 2× speedup.

Our results are presented in Figure 5.11. A smaller C-BTM model, exposed to 168B tokens of text, can achieve the same perplexity as the larger 6.7B dense model with 3.5× speedup. These speedup estimates are dependent on the amount of pretraining performed on each model. Future work may perform these experiments with larger models and many more ELMs.

[14] See §A.4.3 for details on this interpolation.

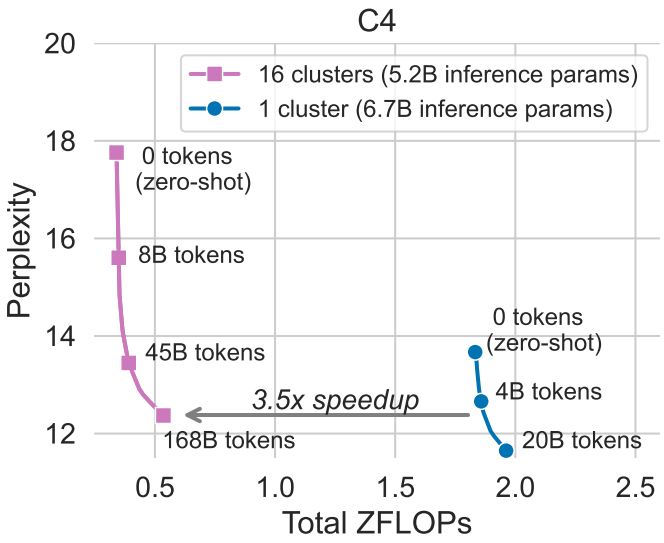

Figure 5.11: Training with C-BTM is substantially more efficient than training a larger dense LM (§5.3.4). We train a 16-cluster C-BTM model and use top-4 inference, resulting in a 5.2B parameter LM, and compare its performance with a 6.7B parameter dense LM. The C-BTM model at 168B tokens achieves the same perplexity on C4 as a 6.7B dense model with 3.5x fewer ZFLOPs. The total ZFLOPs includes the cost of pretraining the seed OPT checkpoints.

5.3.5 Summary

Our results demonstrate that, controlling for a variety of different types of computational budget, C-BTM outperforms dense training in language modeling. Furthermore, we demonstrate that C-BTM results in an effective *sparse* language model, where the top-1, top-2 and top-4 experts from models with at least 8 clusters significantly outperform 1-cluster, 2-cluster and 4-cluster models. These results suggest the possibility of outperforming dense models by increasing margins, while keeping both training and inference costs fixed, as compute and the number of experts grow.

5.4 Downstream Task Results

Do the trends from §5.3 extend to downstream settings? To begin to answer this question, we perform few-shot evaluation on six downstream text classification tasks. We indeed find that models trained with C-BTM outperform their dense counterparts in these settings.

	Few-shot Text Classification Accuracy (%)						
↓ Model (params)	**AGNews** *Topic*	**DBPedia** *Topic*	**SST-2** *Sentiment*	**Amazon** *Sentiment*	**Phrasebank** *Sentiment*	**Twitter** *Hatespeech*	**Average**
Random	25.0	7.10	50.0	50.0	33.3	50.0	35.9
OPT (1.3B)	42.9	57.2	72.8	81.3	72.5	**65.1**	65.3
OPT (6.7B)	51.9	58.9	77.0	83.8	76.4	39.6	64.6
1-cluster (1.3B)	47.4	61.1	80.2	80.7	66.6	60.9	66.2
1-cluster (6.7B)	**68.1**	62.4	80.7	**84.9**	**80.6**	37.4	69.0
16-cluster				*Cluster Routing*			
top-1 (1.3B)	47.1	**62.9**	74.3	79.1	72.9	56.4	65.4
top-4 (5.2B)	49.3	62.3	80.0	81.3	78.7	61.3	68.8
top-16 (20.8B)	50.6	62.0	**84.0**	83.2	78.6	61.7	**69.9**

Table 5.1: C-BTM models outperform dense counterparts on downstream text classification tasks (§5.4.2). We display accuracy of models from §5.3 on six text classification tasks, using eight demonstrations for each example and no additional fine-tuning. We report accuracy averaged over five random seeds. The 1- and 16-cluster models are trained on 168B tokens of C4 (i.e., our highest budget). The 16-cluster model, with top-4 or top-16 inference, always outperforms the 1-cluster model, and top-1 inference usually outperforms the 1-cluster model at no additional inference cost. We include average performance of models across tasks for readability.

5.4.1 Experimental Setup

Tasks We experiment with six text classification tasks, spanning topic, sentiment, and hatespeech classification. Details of the datasets are in Appendix §A.4.7.

Few-shot inference We perform 8-shot evaluations. For each task, we randomly sample 8 examples with their labels from the train set, and prepend them as demonstrations for each test example. For C-BTM models, we estimate ensemble weights for each example by passing both the example and the demonstrations through our pretrained clusterer (§5.1.2). We calculate the probability of each label for the task under the model, and report accuracy by counting the proportion of test examples where the gold label has the highest probability. We report average accuracy over 5 random seeds. We leave careful analysis of C-BTM with varying numbers of demonstrations and few-shot inference techniques to future work.

Baselines We compare the performance of 1- and 16-cluster C-BTM models trained on 168B tokens of C4 (i.e., our highest budget from §5.3). For the 16-cluster model, we also perform top-1 and top-4 inference (§5.3.3). We additionally compare against a random baseline, the original OPT-1.3B and 6.7B models (without any additional training), and the 6.7B parameter 1-cluster model trained on 20B tokens of C4.

5.4.2 Results

Our results in Table 5.1 show that the 16-cluster C-BTM model always outperforms the 1-cluster, 1.3B parameter baseline, sometimes dramatically. This aligns with our language modeling results (§5.3.1). The 1-cluster model achieves lower accuracy than OPT-1.3B on some tasks despite additional training, suggesting that our models may suffer from catastrophic forgetting, since the C4 corpus we use in our experiments is out-of-domain to OPT.

Nevertheless, the 16-cluster model outperforms OPT-1.3B on all tasks other than Twitter. Also, top-1 and top-4 inference matches or exceeds using all experts in some settings, consistent with our language modeling results in §5.3.3. We examine the clusters associated with the most likely experts for each task, and find that their top-terms are relevant to the task's domain (Table A.13 in the appendix). This supports our hypothesis that C-BTM is able to leverage any part of the corpus which is in-domain to the test task, even if the training corpus as a whole might be sufficiently out-of-domain as to have a negative effect on performance.

We then mirror the analysis in §5.3.4, by comparing our 16-cluster models to 6.7B parameter dense models. First, we observe that our 1-cluster 6.7B model outperforms OPT-6.7B on all tasks except Twitter, possibly because this model has had less exposure to C4, and suffers from less catastrophic forgetting. Our 16-cluster model performs comparably to both 6.7B models, and on multiple tasks, our 16-cluster model *outperforms* both 6.7B models, which have been trained with at least 3.5× more compute (§5.3.4). With top-4 inference, C-BTM models activate even fewer parameters than the 6.7B parameter models, yet perform comparably. These results corroborate our findings in §5.3.4 that compared to larger dense models, models trained with C-BTM have more training efficiency and lower inference latency, and result in comparable or better performance.

In separate experiments, we observe that routing examples to experts based on their performance on few shot examples, rather their clusters, results in even better downstream task performance with C-BTM models. This is likely because few-shot performance depends on factors such as example order, label distributions, and the quality of the demonstrations, which not necessarily tied to the domain of the task [Min et al., 2022; Lu et al., 2022]. We analyze this finding further in §A.4.7, and leave more careful development of routing protocols for downstream tasks to future work.

5.4.3 Summary

We demonstrate that, consistent with the language modeling results in §5.3.1, C-BTM improves downstream performance on a variety of few-shot text classification tasks. C-BTM models consistently outperform dense 1-cluster baselines, and usually outperform the original OPT models, despite being trained on an out-of-domain corpus. We also find that top-k activation reduces inference costs with negligible effects on downstream task performance. C-BTM models perform comparably to larger, 6.7B OPT and 1-cluster dense baseline models, despite being trained with 3.5x less compute, and even when activating fewer inference parameters.

5.5 Comparing to Mixture-of-Experts

Finally, we compare C-BTM against an alternative sparse LM, a mixture-of-experts (MoE) which learns a routing between tokens and feedforward experts in the transformer [Lepikhin et al., 2021; Fedus et al., 2022b]. As discussed in §5.1.5, C-BTM is substantially simpler than MoE.

5.5.1 Sparse Upcycling

To mirror C-BTM seed initialization, we initialize our MoE with a dense checkpoint. We use the *sparse upcycling* technique from Komatsuzaki et al. [2022]. Upcycling a dense model into an MoE with k experts entails initializing shared parameters (e.g., attention and embedding layers) and k expert parameters (e.g., every other feedforward layer) from a dense checkpoint, and initializing new parameters for the router. Then the model is simply trained as an MoE. Here, we use top-2, token-level routing [Lepikhin et al., 2021].

5.5.2 Experimental Setup

Hyperparameters We train an MoE with sparse upcycling on C4, starting from OPT-1.3B and using the same general experimental setup detailed in §5.2.2. We follow the settings from Komatsuzaki et al. [2022] as closely as possible. We conducted experiments with 8, 16, 32, 64, and 128 experts for each compute budget. 8 and 16 experts are similar to, but slightly worse than, 32 experts; 64 experts and 128 experts consistently have exploding losses, and the few which successfully train are also similar to but slightly worse

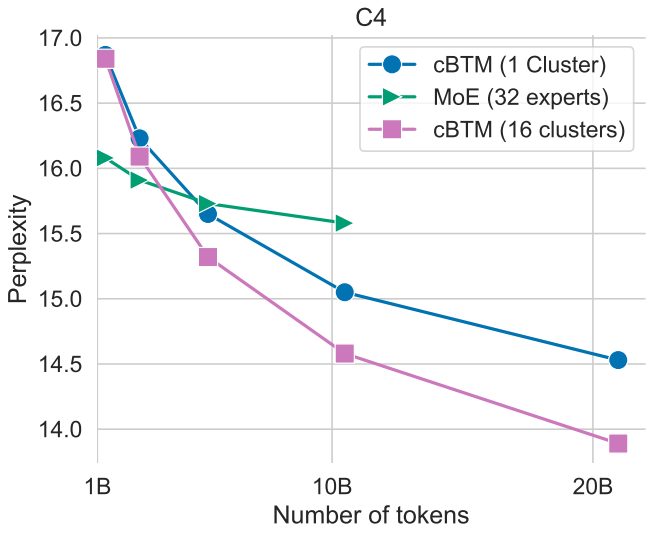

Figure 5.12: MoE underperforms C-BTM (§5.5.3). We compare a 32-expert MoE with top-2 routing [Lepikhin et al., 2021] trained with sparse-upcycling [Komatsuzaki et al., 2022]. While the MoE outperforms C-BTM models with 16 experts at small budgets, it fails at larger budgets, even under-performing the dense model. We speculate this could be due to distribution shifts after pretraining, which might increase the instability of upcycling.

than 32 experts. In general, we find that both large expert count (and higher compute budgets) result in sparse upcycling training instability.

We use 32 experts in our MoE, a capacity factor of 2, and continue training without resetting the optimizer from that used during OPT pretraining. We set all hyperparameters to be the same as our C-BTM models (§5.2.2), except that we use a peak learning rate of 2e-5, which we found to be the highest learning rate that that did not result in divergence after a sweep. We release our code for sparse upcycling, implemented in Fairseq [Ott et al., 2019b], publicly.[15]

Baselines We compare the 32-expert MoE LM to 1-cluster (i.e., dense) and 16-cluster C-BTM models.

5.5.3 Results

The MoE expends more FLOPs than the other models due to the additional feedforward layer at every other transformer block for top-2 routing, as well as the routing projection [Artetxe et al., 2021]. For clarity and consistency, we update-match and separately report GPU-time cost of updates, as in §5.3.1.

[15] https://github.com/kernelmachine/moe-fairseq

We display results in Figure 5.12. MoE substantially underperforms C-BTM with 16 clusters as the compute budget grows. Surprisingly, we observe that with enough compute, MoE underperforms even the dense LM, and that when compute budgets are further increased, losses consistently explode. This suggests that sparse upcycling is highly unstable, possibly due to distribution shifts from pretraining.

In Figure 5.8, we compare the maximum seconds-per-update of the MoE model with that of the C-BTM models. MoE becomes substantially slower as more GPUs are used during training. This is largely due to expensive all-to-all communication that occurs between experts during MoE training, which is necessary to route tokens to experts [Artetxe et al., 2021]. On the other hand, our method does not have any shared parameters between experts. Also, MoE expends more FLOPs during training than the C-BTM models. Finally, MoE still requires synchronous compute to train experts due to shared parameters, so they are also afflicted by the practical difficulties of training dense language models at scale (§5.3.2).

5.5.4 Summary

Our results suggest that language models trained with C-BTM substantially outperform MoEs trained to the same budget. The performance gains of our technique likely are a result of the simplicity of our deterministic routing (based on empirically derived clusters), instabilities associated with sparse upcycling, and other factors.

5.6 Analysis

In §5.3, §5.4, and §5.5, we demonstrate that C-BTM outperforms compute-matched densely trained and MoE baselines. We now study our clustering approach in more detail and describe its effect on overall performance of C-BTM.

5.6.1 Is clustering important?

To assess the importance of the clustering algorithm, we perform C-BTM as above, except that we assign each document to a random cluster, rather than a learned one. This is equivalent to the random ensemble baseline from Li et al. [2022]. Results in Figure 5.13 demonstrate that using random clusters dramatically underperforms both our method and the dense baseline. Therefore, cluster specialization is vital for C-BTM.

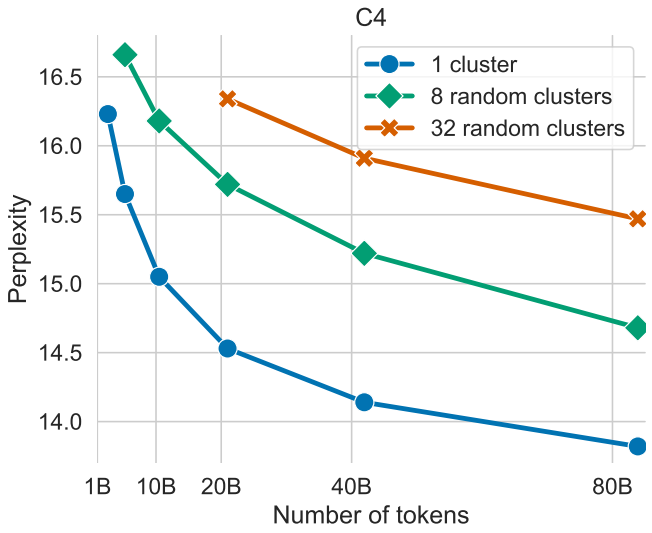

Figure 5.13: Random clusters underperform (§5.6.1). Training experts on random clusters underperforms even the dense, single cluster model, showing the importance of cluster specialization. Random clusters become more harmful as the cluster count grows.

This confirms results from Li et al. [2022], who found that domain specialization of ELMs is critical for performance the ensemble, as well as those from Jang et al. [2023], who show that instruction-specialized ELMs transfer to other tasks with similar instructions.

5.6.2 Is it important to balance the clusters?

Applying a balancing constraint in k-means avoids the degenerate outcome of a long tail in cluster sizes (Chang et al. 2014). Indeed, with 10K documents of held out validation data in C4, we observe that balanced clustering significantly increases the median cluster size, and narrows its range, relative to an unbalanced baseline (§A.4.4). To assess the effect of balancing cluster size on the performance of C-BTM, we perform C-BTM with a k-means clustering model but remove the balancing constraint. For the 8-cluster model, we observe that balancing has little effect. However, for the 32-cluster model (Figure 5.14), unbalanced clustering consistently leads to worse performance. These results suggest that balancing becomes more important as one scales the number of clusters. This is consistent with separate experiments that show that the long tail in cluster sizes becomes a more consistent problem with higher cluster counts, which may lead to undertrained ELMs.

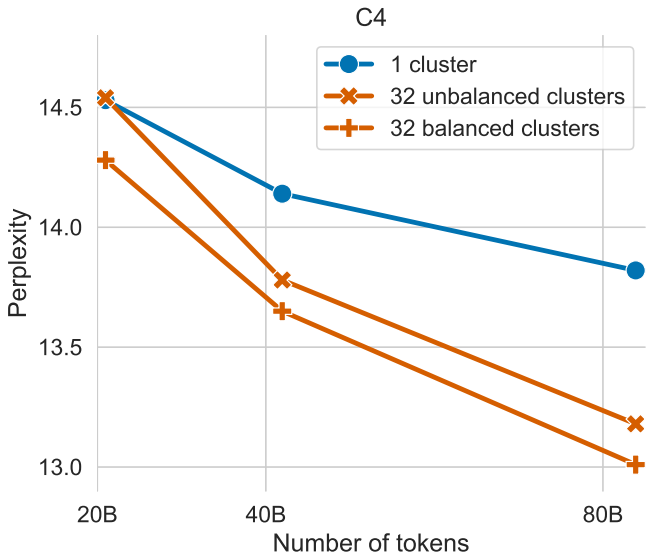

Figure 5.14: Cluster balancing improves performance (§5.6.2). When training on C4 with 32 clusters, balancing consistently improves over the unbalanced version, suggesting that cluster size important aspect to C-BTM.

5.6.3 Are clusters well defined? Do experts specialize?

Since we use tf-idf as our document embedder in C-BTM, we can perform an inverse transform from the cluster centers into the vocabulary space to identify terms that most likely would have been embedded as the cluster center. We display the top five terms per cluster in §A.4.1. We observe that as the number of clusters increases, the top terms across clusters become more specific and varied.

Next, we study whether ELMs trained on these clusters specialize. Using the 32-cluster model trained on 84B tokens of C4, we compute perplexity of all experts across 200 held out documents in each cluster. For each cluster, we then measure the ratio of the perplexity of each expert to the perplexity of the expert trained on that cluster.

We display those ratios in Figure 5.15. We see that all experts perform best on their own cluster. Some experts do not transfer well at all to other clusters, while others do reasonably well. Cross referencing with the cluster term tables in §A.4.1, we see that cluster experts 3 and 5 tend to generalize well and the top terms in these clusters are more generic (with words such as "*just*, *like*, *love*"). The experts specialized to content such as *"site, page, website"* (cluster 0) and *"app, phone, video"* (cluster 29), tend to do poorly on all other

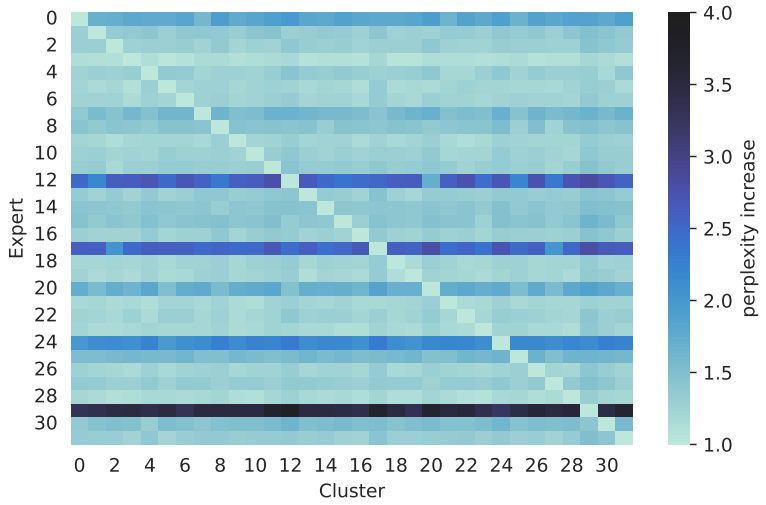

Figure 5.15: Experts specialize to their cluster (§5.6.3). Here, we use the 32-cluster model trained on 84B tokens of C4. Each cell is a ratio between one expert's test perplexity on a cluster to that of the expert trained on that cluster. The diagonal indicates that experts perform best on the cluster they were trained on. Many experts transfer well across clusters, but some do not, likely because they contain content not useful for generalizing to other domains.

clusters.[16] These results suggest that experts specialize to their cluster. We infer that the success of sparse C-BTM inference is a result of expert specialization, and that C-BTM performance gains may be partially due to the sample efficiency of specialized training.

5.6.4 How do discovered clusters and metadata domains compare?

The key motivation for C-BTM is to remove the reliance on metadata to delineate the domains to which ELMs specialize. How well do clusters reflect the dataset segmentation produced by metadata? We use S2ORC to study this question. First, we align the learned clusters from a 32-cluster model with the fields-of-study metadata available from S2ORC [Lo et al., 2019]. Then we visualize the overlap between the metadata and clusters (Figure 5.16). We observe only a partial alignment between metadata and clusters in S2ORC. Documents with some metadata labels (e.g., Enviromental Science, Political Science) are mostly assigned to their own clusters, while documents with other labels (e.g., Computer Science, Physics) are distributed across multiple clusters.

[16]This result imply that cluster experts can be removed to filter out unwanted generations after training, without significantly impacting performance on other content. We leave such exploration to future work.

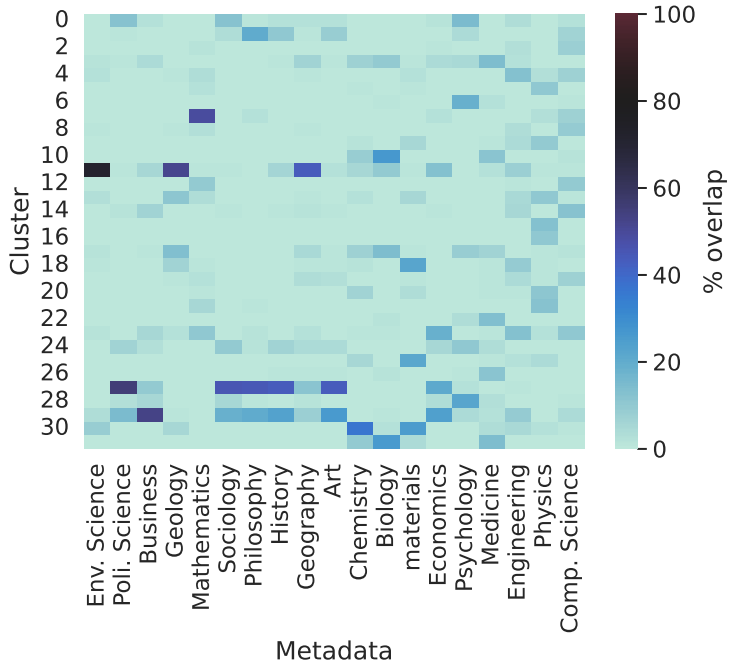

Figure 5.16: Clusters and metadata do not perfectly align (§5.6.4). Each cell in the heatmap is the % overlap between a cluster and a metadata label identifying the field-of-study of a document in S2ORC; high overlap indicates that most documents with the corresponding label get assigned to the corresponding cluster. While documents with certain labels (e.g., Environmental Science, Political Science, Business) get primarily assigned to a single cluster, documents with other labels (e.g., Engineering, Physics, Computer Science) are distributed across multiple clusters.

The partial alignment between metadata and clusters suggests that C-BTM models may not have the same performance as those trained with metadata labels to delineate domains. To investigate this hypothesis further, we perform experiments using a subset of the Pile [Gao et al., 2021] to compare the performance of experts trained with metadata and experts trained with clusters. See §A.4.5 for more details on this corpus. We observe that experts trained with learned clusters perform only slightly better ($\sim$0.1 perplexity points) than those with metadata labels on a held out validation data (Table A.11 in the appendix). Both techniques perform better than training with just a single cluster on the Pile, confirming our results from §5.3.1. These results imply that metadata may not correspond with the most optimal segmentation of the corpus, possibly due to suboptimal balancing. However, using metadata has the advantage of interpretability and simplicity, and metadata can identify domains that are not just lexically driven [e.g., Lucy and Bamman, 2021c; Gururangan et al., 2022]. Future work may explore combining metadata- and cluster-specialized ELMs.

5.6.5 Summary

Our analysis demonstrates that the improvements from C-BTM are not the result of ensembling alone. Various components of our training method, particularly the nature of the learned clusters, play a critical role in C-BTM performance. Improving the representation of domains in a corpus, perhaps using other pretrained representations [Aharoni and Goldberg, 2020b] or more sophisticated clustering algorithms [Ester et al., 1996; Chronopoulou et al., 2022], are likely to improve C-BTM performance.

5.7 Related Work

Sparse Models C-BTM is closely related to sparse models which activate only a subset of parameters [Evci et al., 2020; Mostafa and Wang, 2019; Dettmers and Zettlemoyer, 2019]. C-BTM is inspired by MoE, but is much simpler and more efficient to train. Most MoE methods rely on training token-based routing mechanisms [Lepikhin et al., 2021; Fedus et al., 2022b; Lewis et al., 2021b; Roller et al., 2021], but others rely on task [Kudugunta et al., 2021a] or domain [Gururangan et al., 2022] routing.

Expert Language Models As we note throughout the study, this work is most directly related to BTM [Li et al., 2022]. BTM is in turn partially inspired by prior work on variations of MoE models [Jacobs et al., 1991b], but especially DEMix layers [Gururangan et al., 2022], which replace transformer feedforward layers with metadata-defined domain experts. Jang et al. [2023] train expert language models on instruction-based tasks, while Pfeiffer et al. [2022] train expert language models on different languages.

Cluster Routing Chronopoulou et al. [2022] and Chronopoulou et al. [2023] use hierarchical clustering to identify domains to specialize adapter experts to, and use the adapters in an ensemble or parameter average at inference time. Duan et al. [2021] build ensembles of task-specific models by clustering the training data of supervised tasks. Gross et al. [2017] employ a cluster-based router similar to ours in an image classification setting using ResNets. However, they use a hard routing (or only activate a single expert) in both training and inference, while we use hard routing during training but ensemble experts during inference. Tian et al. [2021] pretrain experts on clusters of vision datasets, but distill experts for inference. Our inference technique is inspired by nearest neighbor retrieval mechanisms in language models [Khandelwal et al., 2019; Shi et al.,

2022].

Communication-efficient training Our study contributes to a line of research into communication-efficient algorithms for training large models. Some previous work proposes ways to train large dense models collaboratively over distributed networks of servers [Borzunov et al., 2022; Yuan et al., 2022]. Other works focus on new forms of data [Gan et al., 2021], model [Ryabinin et al., 2023], and pipeline [Wang et al., 2022a] parallelism to allow for model training on heterogeneous devices that can recover from node failures. Wortsman et al. [2022] propose a communication-efficient method of fine-tuning by training a collection of models with different hyperparameters on individual GPU nodes, and then averaging their parameters after training. Our work uses expert specialization for communication efficient training, and C-BTM can be combined with any of these other techniques to improve training efficiency.

5.8 Conclusion

We introduce C-BTM, a new technique to efficiently train large decentralized LMs. C-BTM splits a corpus into k clusters, trains an expert LM on each cluster, and creates a sparse ensemble during inference. We observe that the optimal number of clusters for C-BTM increases with the amount of data, and using more clusters also allows us to aggressively parallelize training to efficiently scale into massive datasets. Future work could investigate C-BTM in multitask or multilingual settings, the usefulness of multiple iterations of C-BTM on a corpus (perhaps with hierarchical clustering), or the possibility of combining metadata- and cluster-based routing to scale into many heterogeneous datasets in parallel.

5.9 Limitations

The methods presented in this chapter imply that entire language models need to be finetuned towards each domain. As domains become smaller and experts larger, this may become suboptimal. Future work could explore relaxing this assumption with parameter efficient experts that are of different size.

In this chapter, we only explore a single round of C-BTM; future work may explore doing multiple rounds of training, using for example, hierchical clustering to generate data segmentations that are progressively

Task	1st	2nd	3rd
AGNews	*game, team, season*	*said, government, president*	*business, company, market*
DBPedia	*students, school, university*	*said, government, president*	*city, park, hotel*
SST-2	*book, film, life*	*love, family, great*	*music, art, band*
Amazon	*book, film, life*	*just, like, know*	*game, team, season*
Phrasebank	*business, company, market*	*service, customer, products*	*said, government, president*
Twitter	*data, software, download*	*click, website, page*	*just, like, know*

Table 5.2: Top-terms of clusters associated with top-3 experts for each classification task (§5.4.2) By inspection, the highest probability experts are usually quite relevant to task's domain.

finer grained and specialized, which is likely to improve performance [Li et al., 2022]. Additionally, our clustering-based inference assumes that all experts can be loaded in parallel, to avoid latency costs. With more or larger experts, parallel loading may be infeasible. This implies that the clustering inference mechanism may incur additional latency costs, since experts would need to be swapped into the GPU from disk for every new context. Applying additional priors (as we do in §A.4.7) may improve the method's efficiency, since the experts necessary for a particular test set could be fixed.

Chapter 6

Conclusion

Language models are powerful tools for natural language processing, and the impressive effect of scale on their performance cannot be ignored. However, current trends in NLP research ignore underlying variation in language model training data, instead emphasizing the idea that scale yields general purposeness, or that language models can be applied to all language variations universally.

In Chapter 2, we empirically demonstrate that despite their scale, current language model training data implicitly favor certain language ideologies that emphasize text from authors of hegemonic social positions. We argue that it is impossible to build a general purpose corpus due to unintended ideologies that are reflected in the data selection.

This thesis presents a new framework that bakes language variation into language model training, and moves away from the laissez faire treatment of data. The lack of a universal corpus motivates the techniques we present in Chapter 3, to customize language models to new distributions of text that are not well represented in the pretraining data. We present multi-stage adaptation to domains and tasks, showing that specializing models along a spectrum of coarser to finer granularities of data distributions results in the best performance.

The basic technique of continued pretraining leads into techniques presented in Chapter 4, where we conditionally update parameters in the original pretraining phase based on the domain of the incoming document. We then suggest *growing* the model with new experts as the model encounters new domains after the initial pretraining phase, as a way to mitigate catastrophic forgetting. We also suggest *removing* experts as a way to forget particular data distributions with strong guarantees.

The techniques of DEMix inspire our main algorithm in Chapter 5, called C-BTM, which applies conditional compute to *all* parameters of the language model, enabling completely asynchronous training of experts – a truly decentralized language model. We demonstrate the scaling patterns of this approach on massive corpora, showing that the benefits of decentralized training over dense baselines grows with data and compute, suggesting this is a very efficient and accessible way to train large language models.

Overall, our results suggest that avoiding treating data homogeneously is not just desirable from an ethical perspective. It also leads to cheap customization options after training which empower users, efficient scaling into massive corpora, and more guarantees on model behavior. This all comes with better performance over dense baselines across numerous axes, such as loss, downstream performance, and overall efficiency.

Our research in Chapter 2 could be extended to quantitatively understand whose language is included directly in web crawl data. While it is difficult to make such broad assesements due to the lack of metadata in web crawl, certain subsets (such as *About Me* pages) may provide data around identities of authors for such analysis. More tools to improve the data transparency around large language model training data would support the arguments we make in this thesis.

C-BTM may be improved with hierarchical clustering, to make use of finer micro-domain distributions, as we saw in Chapter 3. Relaxing the architectural constraints of C-BTM would likely improve performance as well. We envision expert models that are of different sizes, or even perhaps those that are parameter efficient [Houlsby et al., 2019]. The vocabulary and tokenizers could also be made more flexible and customized per domain.

Finally, further research on deployment of C-BTM models would prove to be fruitful as well. We explore a basic version of ensembling experts at inference time, which might introduce additional latency based on the amount of compute available and number of experts used. Li et al. [2022] suggested parameter averaging as a reasonable way to combine models; more sophisticated methods of model merging [Yadav et al., 2023; Ilharco et al., 2023] and distillation [Kudugunta et al., 2021b] could also be explored.

Bibliography

Abubakar Abid, Maheen Farooqi, and James Zou. 2021. Persistent anti-Muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*.

Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. 2021a. Htlm: Hyper-text pre-training and prompting of language models. *arXiv*, abs/2107.06955.

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021b. Better fine-tuning by reducing representational collapse. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Roee Aharoni and Yoav Goldberg. 2020a. Unsupervised domain clusters in pretrained language models. In *ACL*. To appear.

Roee Aharoni and Yoav Goldberg. 2020b. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763, Online. Association for Computational Linguistics.

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*.

Gabriel Arana. 2018. Decades of failure. *Columbia Journalism Review*.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges.

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, Giri Anantharaman, Xian Li, Shuohui Chen, Halil Akin, Mandeep Baines, Louis Martin, Xing Zhou, Punit Singh Koura, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Mona Diab, Zornitsa Kozareva, and Ves Stoyanov. 2021. Efficient large scale language modeling with mixtures of experts.

David Arthur and Sergei Vassilvitskii. 2007. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA. Society for Industrial and Applied Mathematics.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. In *EMNLP*.

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. 2022. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074.

Tyler Baldwin and Joyce Chai. 2011. Beyond normalization: Pragmatics of word form in text messages. In *Proceedings of 5th International Joint Conference on Natural Language Processing*.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3528–3539, Brussels, Belgium. Association for Computational Linguistics.

David Bamman and Noah A Smith. 2014. Unsupervised discovery of biographical structure from text. *Transactions of the Association for Computational Linguistics*, 2:363–376.

Jack Bandy and Nicholas Vincent. 2021. Addressing "documentation debt" in machine learning: A retrospective datasheet for BookCorpus. In *NeurIPS*.

Pratyush Banerjee, Jinhua Du, Baoli Li, Sudip Naskar, Andy Way, and Josef van Genabith. 2010. Combining multi-domain statistical machine translation models using automatic classifiers. In *Proceedings of the 9th Conference of the Association for Machine Translation in the Americas: Research Papers*, Denver, Colorado, USA. Association for Machine Translation in the Americas.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *EMNLP*.

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

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021a. On the dangers of stochastic parrots: Can language models be too big.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021b. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Jonah Berger and Katherine L. Milkman. 2012. What makes online content viral? *Journal of Marketing Research*, 49(2):192–205.

Dimitri P. Bertsekas. 1992. Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications*, 1:7–66.

Julian R. Betts, Kim S. Reuben, and Anne Danenberg. 2000. *Equal Resources, Equal Outcomes? The Distribution of School Resources and Student Achievement in California*. Public Policy Institute of California.

Douglas Biber. 1988. *Variation across Speech and Writing*. Cambridge University Press.

Steve Bien-Aimé. 2016. AP stylebook normalizes sports as a male space. *Newspaper Research Journal*, 37(1):44–57.

Daniel Blanchard, Joel R. Tetreault, Derrick Higgins, A. Cahill, and Martin Chodorow. 2013. TOEFL11: A corpus of non-native English. *ETS Research Report Series*, 2013:15.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Su Lin Blodgett. 2021. *Sociolinguistically Driven Approaches for Just Natural Language Processing*. Ph.D. thesis, University of Massachusetts Amherst.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

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 EMNLP*.

Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. 2022. Petals: Collaborative inference and fine-tuning of large models.

Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. 2023. Petals: Collaborative inference and fine-tuning of large models.

Sarah Brayne. 2017. Big data surveillance: The case of policing. *American Sociological Review*, 82(5):977–1008.

Julian Brooke, Adam Hammond, and Graeme Hirst. 2015. GutenTag: an NLP-driven tool for digital humanities research in the Project Gutenberg corpus. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–480.

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

Niels Brügger. 2011. Web archiving - between past, present, and future.

Dallas Card, Chenhao Tan, and Noah A. Smith. 2018. Neural models for documents with metadata. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Caselaw Access Project. Caselaw access project.

Soumen Chakrabarti, Martin van den Berg, and Byron Dom. 1999. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. *Comput. Networks*, 31:1623–1640.

Tuhin Chakrabarty, Christopher Hidey, and Kathy McKeown. 2019. IMHO fine-tuning improves claim detection. In *NAACL*.

Xiaojun Chang, Feiping Nie, Zhigang Ma, and Yi Yang. 2014. Balanced k-means and min-cut clustering.

Ethan C. Chau, Lucy H. Lin, and Noah A. Smith. 2020. Parsing with multilingual BERT, a small corpus,

and a small treebank. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1324–1334, Online. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Michael Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014a. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014b. One billion word benchmark for measuring progress in statistical language modeling.

Janice Kai Chen, Ilena Peng, Jasen Lo, Trisha Ahmed, Simon J. Levien, and Devan Karp. 2021a. Voices investigation: Few black, latinx students are editors of top college newspapers. *AAJA Voices*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021b. Evaluating large language models trained on code.

Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, Alexandre Sallinen, Alireza Sakhaeirad, Vinitra Swamy, Igor Krawczuk, Deniz Bayazit, Axel Marmet, Syrielle Montariol, Mary-Anne Hartley, Martin Jaggi, and Antoine Bosselut. 2023. Meditron-70b: Scaling medical pretraining for large language models.

Zhiyuan Chen, Bing Liu, Ronald Brachman, Peter Stone, and Francesca Rossi. 2018. *Lifelong Machine Learning: Second Edition*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. 2019. An embarrassingly simple approach for transfer learning from pretrained language models. In *NAACL*.

Alexandra Chronopoulou, Matthew Peters, and Jesse Dodge. 2022. Efficient hierarchical domain adaptation for pretrained language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1336–1351, Seattle, United States. Association for Computational Linguistics.

Alexandra Chronopoulou, Matthew E. Peters, Alexander M. Fraser, and Jesse Dodge. 2023. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *ArXiv*, abs/2302.07027.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew Johnson, Katie Millican, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. 2022. Unified scaling laws for routed language models.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Arman Cohan, Iz Beltagy, Daniel King, Bhavana Dalvi, and Dan Weld. 2019. Pretrained language models for sequential sentence classification. In *EMNLP*.

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

Justin T. Craft, Kelly E. Wright, Rachel Elizabeth Weissler, and Robin M. Queen. 2020. Language and discrimination: Generating meaning, perceiving identities, and discriminating outcomes. *Annual Review of Linguistics*, 6(1):389–407.

Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2019. Using similarity measures to select pretraining data for NER. In *NAACL*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Stephanie Decker. 2013a. The silence of the archives: business history, post-colonialism and archival ethnography. *Management & Organizational History*, 8(2):155–173.

Stephanie Decker. 2013b. The silence of the archives: business history, post-colonialism and archival ethnography. *Management & Organizational History*, 8(2):155–173.

Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. 2021. The efficiency misnomer.

Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k RCT: a dataset for sequential sentence classification in medical abstracts. In *IJCNLP*.

Jacques Derrida and Eric Prenowitz. 1995. Archive fever: A freudian impression. *Diacritics*, 25(2):9–63.

Tim Dettmers and Luke Zettlemoyer. 2019. Sparse networks from scratch: Faster training without losing performance. *CoRR*, abs/1907.04840.

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

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

Emily Dinan, Gavin Abercrombie, A. Stevie Bergman, Shannon Spruit, Dirk Hovy, Y-Lan Boureau, and Verena Rieser. 2021. Anticipating safety issues in e2e conversational ai: Framework and tooling.

Robert DiNicola. 1994. Teaching journalistic style with the AP stylebook: Beyond fussy rules and dogma of 'correctness'. *The Journalism Educator*, 49(2):64–70.

Jesse Dodge, Maarten Sap, Ana Marasovic, William Agnew, Gabriel Ilharco, Dirk Groeneveld, and Matt Gardner. 2021. Documenting the english colossal clean crawled corpus. *arXiv*, abs/2104.08758.

T. Draelos, N. Miner, Christopher C. Lamb, Jonathan A. Cox, Craig M. Vineyard, Kristofor D. Carlson, William M. Severa, C. James, and J. Aimone. 2017. Neurogenesis deep learning: Extending deep networks to accommodate new classes. *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 526–533.

Zhibin Duan, Hao Zhang, Chaojie Wang, Zhengjue Wang, Bo Chen, and Mingyuan Zhou. 2021. Enslm: Ensemble language model for data diversity by semantic clustering. In *Annual Meeting of the Association for Computational Linguistics*.

Penelope Eckert. 1989. *Jocks and burnouts: Social categories and identity in the high school*. Teachers college press.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

Jacob Eisenstein, Brendan O'connor, Noah A. Smith, and Eric P. Xing. 2014a. Diffusion of lexical change in social media. *PloS ONE*.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2014b. Diffusion of lexical change in social media. *PLoS ONE*, 9(11):e113114.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2943–2952. PMLR.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation.

William Fedus, Jeff Dean, and Barret Zoph. 2022a. A review of sparse expert models in deep learning.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity.

William Fedus, Barret Zoph, and Noam Shazeer. 2022b. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Casey Fiesler, Nathan Beard, and Brian Keegan. 2020. No robots, spiders, or scrapers: Legal and ethical regulation of data collection methods in social media terms of service. In *Proceedings of ICWSM*.

Paula Froke, Anna Jo Bratton, Jeff McMillan, Pia Sarkar, Jerry Schwartz, and Raghuram Vadarevu. 2020. *The Associated Press stylebook 2020-2022*. The Associated Press.

Susan Gal. 2016. *Sociolinguistic differentiation*, page 113–136. Cambridge University Press.

Susan Gal and Judith T. Irvine. 1995. The boundaries of languages and disciplines: How ideologies construct difference. *Social Research*, 62(4):967–1001.

Shaoduo Gan, Xiangru Lian, Rui Wang, Jianbin Chang, Chengjun Liu, Hongmei Shi, Shengzhuo Zhang, Xianghong Li, Tengxu Sun, Jiawei Jiang, Binhang Yuan, Sen Yang, Ji Liu, and Ce Zhang. 2021. Bagua: Scaling up distributed learning with system relaxations.

Leo Gao. 2021. An empirical exploration in quality filtering of text data. *arXiv*, abs/2109.00698.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021. The pile: An 800gb dataset of diverse text for language modeling.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *NLP-OSS*.

Tiana Gaudette, Ryan Scrivens, Garth Davies, and Richard Frank. 2021. Upvoting extremism: Collective identity formation and the extreme right on reddit. *New Media & Society*, 23(12):3491–3508.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2021. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020a. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020b. RealToxici-

tyPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Joyce Still Gibson. 1961. A study of the status of high school newspapers in the virginia public schools. Master's thesis, University of Richmond.

Github Archive Project. Github archive project.

Aaron Gokaslan and Vanya Cohen. 2019a. Openwebtext corpus.

Aaron Gokaslan and Vanya Cohen. 2019b. Openwebtext corpus.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. Book in preparation for MIT Press.

Rob van der Goot, Alan Ramponi, Arkaitz Zubiaga, Barbara Plank, Benjamin Muller, Iñaki San Vicente Roncal, Nikola Ljubešić, Özlem Çetinoğlu, Rahmad Mahendra, Talha Çolakoğlu, Timothy Baldwin, Tommaso Caselli, and Wladimir Sidorenko. 2021. MultiLexNorm: A shared task on multilingual lexical normalization. In *Proceedings of the Seventh Workshop on Noisy User-generated Text*.

Eduardo Graells-Garrido, Mounia Lalmas, and Filippo Menczer. 2015. First women, second sex: Gender bias in Wikipedia. In *Proceedings of the 26th ACM conference on hypertext & social media*.

Rob Greenwald, Larry V. Hedges, and Richard D. Laine. 1996. The effect of school resources on student achievement. *Review of Educational Research*, 66(3):361–396.

Elizabeth Grieco. 2018. Newsroom employees are less diverse than U.S. workers overall. *Pew Research Center*. [online; accessed 2022-01-22].

Sam Gross, Marc'Aurelio Ranzato, and Arthur Szlam. 2017. Hard mixtures of experts for large scale weakly supervised vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6865–6873.

Jiang Guo, Darsh Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4694–4703, Brussels, Belgium. Association for Computational Linguistics.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. 2019. Variational pretraining for semi-supervised text classification. In *ACL*.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. 2022. DEMix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5557–5576, Seattle, United States. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *EMNLP*.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

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.

Dirk Hovy and Diyi Yang. 2021. The importance of modeling social factors of language: Theory and practice. In *Proceedings of NAACL*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Keira Huang. 2013. Wikipedia fails to bridge gender gap. *South China Morning Post*. [online; accessed 2022-01-11].

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. ClinicalBERT: Modeling clinical notes and predicting hospital readmission. arXiv:1904.05342.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic.

Robert Jacobs, Michael Jordan, Steven Nowlan, and Geoffrey Hinton. 1991a. Adaptive mixture of local expert. *Neural Computation*, 3:78–88.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991b. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Aaron Jaech and Mari Ostendorf. 2018. Low-rank rnn adaptation for context-aware language modeling.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2023. Exploring the benefits of training expert language models over instruction tuning.

Eun Seo Jo and Timnit Gebru. 2020. Lessons from archives: Strategies for collecting sociocultural data in machine learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. ACM.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

David Jurgens, Srijan Kumar, Raine Hoover, Daniel A. McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *TACL*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *ICLR*. To appear.

Paresh Kharya and Ali Alvi. 2021. Using deepspeed and megatron to train megatron-turing nlg 530b, the world's largest and most powerful generative language model. [online; accessed 2022-01-20].

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan news detection. In *SemEval*.

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

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. 2021. WILDS: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5637–5664. PMLR.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2022. Sparse upcycling: Training mixture-of-experts from dense checkpoints.

Martin Krallinger, Obdulia Rabal, Saber Ahmad Akhondi, Martín Pérez Pérez, Jésús López Santamaría, Gael Pérez Rodríguez, Georgios Tsatsaronis, Ander Intxaurrondo, José Antonio Baso López, Umesh Nandal, E. M. van Buel, A. Poorna Chandrasekhar, Marleen Rodenburg, Astrid Lægreid, Marius A. Doornenbal, Julen Oyarzábal, Anália Lourenço, and Alfonso Valencia. 2017. Overview of the biocreative vi chemical-protein interaction track. In *Proceedings of the BioCreative VI Workshop*.

Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. 2015. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7(1):S2.

Jens Kringelum, Sonny Kim Kjærulff, Søren Brunak, Ole Lund, Tudor I. Oprea, and Olivier Taboureau. 2016. ChemProt-3.0: a global chemical biology diseases mapping. In *Database*.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. 2021a. Beyond distillation: Task-level mixture-of-experts for efficient inference.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. 2021b. Beyond distillation: Task-level mixture-of-experts for efficient inference.

T Kunny. 1997. A digital dark ages? challenges in the preservation of electronic information of electronic information. In *63rd IFLA Conuncil and General Conference*.

William Labov. 2006. *The Social Stratification of English in New York City*, 2 edition. Cambridge University Press.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomáš Kočiský, Sebastian Ruder, Dani Yogatama, Kris Cao, Susannah Young, and Phil Blunsom. 2021a. Mind the gap: Assessing temporal generalization in neural language models. In *Advances in Neural Information Processing Systems*.

Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Sebastian Ruder, Dani Yogatama, Kris Cao, Tomas Kocisky, Susannah Young, and Phil Blunsom. 2021b. Pitfalls of static language modelling.

David YW Lee. 2001. Genres, registers, text types, domains and styles: Clarifying the concepts and navigating a path through the BNC jungle. *Language Learning & Technology*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Lee & Low Books. 2020. Where is the diversity in publishing? The 2019 diversity baseline survey results. [online; accessed 2021-11-24].

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. 2014. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021a. Base layers: Simplifying training of large, sparse models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6265–6274. PMLR.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021b. Base layers: Simplifying training of large, sparse models.

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

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. 2022. Few-shot learning with multilingual language models.

Stephanie Lindemann. 2005. Who speaks "broken English"? US undergraduates' perceptions of non-native English. *International Journal of Applied Linguistics*, 15(2):187–212.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. Roberta: A robustly optimized bert pretraining approach.

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

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Dan S. Weld. 2019. S2orc: The semantic scholar open research corpus.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020a. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel S. Weld. 2020b. S2ORC: The Semantic Scholar Open Research Corpus. In *ACL*. To appear.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In *ACL*.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *EMNLP*.

Li Lucy and David Bamman. 2021a. Characterizing English variation across social media communities with BERT. *Transactions of the Association for Computational Linguistics*, 9:538–556.

Li Lucy and David Bamman. 2021b. Characterizing English Variation across Social Media Communities with BERT. *Transactions of the Association for Computational Linguistics*, 9:538–556.

Li Lucy and David Bamman. 2021c. Characterizing English variation across social media communities with BERT. *Transactions of the Association for Computational Linguistics*, 9:538–556.

Kelvin Luu, Daniel Khashabi, Suchin Gururangan, Karishma Mandyam, and Noah A. Smith. 2022. Time waits for no one! analysis and challenges of temporal misalignment. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5944–5958, Seattle, United States. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011a. Learning word vectors for sentiment analysis. In *ACL*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011b. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Jeff MacSwan. 2020. Academic english as standard language ideology: A renewed research agenda for asset-based language education. *Language Teaching Research*, 24(1):28–36.

Mikko I. Malinen and Pasi Fränti. 2014. Balanced k-means for clustering. In *International Workshop on Structural and Syntactic Pattern Recognition*.

P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65.

Michael Mandiberg. 2020. Mapping wikipedia. *The Atlantic*. [online; accessed 2021-11-24].

Antonios Maronikolakis and Hinrich Schütze. 2021. Multidomain pretrained language models for green NLP. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 1–8, Kyiv, Ukraine. Association for Computational Linguistics.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning.

Sorin Adam Matei and Brian C. Britt. 2017. *Structural Differentiation in Social Media*. Springer International Publishing.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *ACM SIGIR*.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. 2018. An empirical model of large-batch training.

M. McCloskey and N. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165.

Kris McGuffie and Alex Newhouse. 2020. The radicalization risks of gpt-3 and advanced neural language models.

Zion Mengesha, Courtney Heldreth, Michal Lahav, Juliana Sublewski, and Elyse Tuennerman. 2021. "I don't think these devices are very culturally sensitive."—Impact of automated speech recognition errors on African Americans. *Frontiers in Artificial Intelligence*, 4:169.

Meta-wiki. 2018. Community insights/2018 report/contributors. [online; accessed 2012-11-24].

Albert J Mills and Mills Jean C Helms. 2017. Digging archeology: Postpositivist theory and archival research in case study development. In J Mills Albert, editor, *Insights and Research on the Study of Gender and Intersectionality in International Airline Cultures*, pages 93–109. Emerald Publishing Limited.

David Mimno and Andrew McCallum. 2012. Topic models conditioned on arbitrary features with dirichlet-multinomial regression.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Amy Mitchell, Jeffrey Gottfried, Michael Barthel, and Nami Sumida. 2018. Can Americans tell factual from opinion statements in the news? *Pew Research Center's Journalism Project*. [online; accessed 2022-01-22].

Arindam Mitra, Pratyay Banerjee, Kuntal Kumar Pal, Swaroop Ranjan Mishra, and Chitta Baral. 2020. Exploring ways to incorporate additional knowledge to improve natural language commonsense question answering. arXiv:1909.08855v3.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *ACL*.

Hesham Mostafa and Xin Wang. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4646–4655. PMLR.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. 2023. Scaling data-constrained language models. *arXiv preprint arXiv:2305.16264*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. *Proceedings of machine learning research*, 70:2554–2563.

Sebastian Nagel. 2016. CC-NEWS.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. Scispacy: Fast and robust models for biomedical natural language processing. *Proceedings of the 18th BioNLP Workshop and Shared Task*.

Dong Nguyen, Laura Rosseel, and Jack Grieve. 2021. On learning and representing social meaning in NLP: A sociolinguistic perspective. In *Proceedings of NAACL*.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. 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 188–197, Hong Kong, China. Association for Computational Linguistics.

Jessica Ogden, Susan Halford, and Leslie Carr. 2017. Observing web archives: The case for an ethnographic study of web archiving. In *Proceedings of the 2017 ACM on Web Science Conference*, WebSci '17, pages 299–308, New York, NY, USA. Association for Computing Machinery.

Yonatan Oren, Shiori Sagawa, Tatsunori Hashimoto, and Percy Liang. 2019a. 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 4227–4237, Hong Kong, China. Association for Computational Linguistics.

Yonatan Oren, Shiori Sagawa, Tatsunori B. Hashimoto, and Percy Liang. 2019b. Distributionally robust language modeling.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019a. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Katherine Panciera, Aaron Halfaker, and Loren Terveen. 2009. Wikipedians are born, not made: A study of power editors on wikipedia. In *Proceedings of the ACM 2009 International Conference on Supporting Group Work*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? Adapting pretrained representations to diverse tasks. In *RepL4NLP*.

Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022. Lifting the curse of multilinguality by pre-training modular transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

MinhQuang Pham, Josep Maria Crego, and François Yvon. 2021. Revisiting multi-domain machine translation. *Transactions of the Association for Computational Linguistics*, 9:17–35.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. arXiv:1811.01088.

Barbara Plank. 2016a. What to do about non-standard (or non-canonical) language in NLP. In *KONVENS*.

Barbara Plank. 2016b. What to do about non-standard (or non-canonical) language in nlp.

Pushkala Prasad. 2015. *Crafting Qualitative Research: Working in the Postpositivist Traditions: Working in the Postpositivist Traditions*. Routledge.

Project Gutenberg. Project gutenberg.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018a. Improving language understanding by generative pre-training.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018b. Improving language understanding with unsupervised learning. [online; accessed 2022-01-22].

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

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

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021a. Scaling language models: Methods, analysis & insights from training gopher. *arXiv*, abs/2112.11446.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021b. Scaling language models: Methods, analysis & insights from training gopher.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

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.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 248–256, Singapore. Association for Computational Linguistics.

Alan Ramponi and Barbara Plank. 2020. Neural unsupervised domain adaptation in NLP—A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6838–6855, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Sean F Reardon and Ann Owens. 2014. 60 years after Brown: Trends and consequences of school segregation. *Annual Review of Sociology*, 40:199–218.

Steffen Remus and Chris Biemann. 2016. Domain-Specific Corpus Expansion with Focused Webcrawling. In *LREC*.

John R. Rickford. 1985a. Ethnicity as a sociolinguistic boundary. *American Speech*, 60:99.

John R. Rickford. 1985b. Ethnicity as a sociolinguistic boundary. *American Speech*, 60(2):99–125.

John R Rickford and Sharese King. 2016. Language and linguistics on trial: Hearing rachel jeantel (and other vernacular speakers) in the courtroom and beyond. *Language*, 92(4):948–988.

Robin and Gallagher. 1967. *Africa and the Victorians: The Official Mind of Imperialism*, volume 43.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason E Weston. 2021. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems*.

Jonathan Rosa and Nelson Flores. 2017. Unsettling race and language: Toward a raciolinguistic perspective. *Language in Society*, 46(5):621–647.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016. Towards a continuous modeling of natural language domains. In *Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with Bayesian optimization. In *EMNLP*.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks.

Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. 2023. Swarm parallelism: Training large models can be surprisingly communication-efficient.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *EMC2 @ NeurIPS*.

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*, abs/2111.07997.

Lawrence Saul and Fernando Pereira. 1997. Aggregate and mixed-order Markov models for statistical language processing. In *Second Conference on Empirical Methods in Natural Language Processing*.

Dante J. Scala and Kenneth M. Johnson. 2017. Political polarization along the rural-urban continuum? the geography of the presidential vote, 2000–2016. *The ANNALS of the American Academy of Political and Social Science*, 672(1):162–184.

Ari Schlesinger, Kenton P. O'Hara, and Alex S. Taylor. 2018. Let's talk about race: Identity, chatbots, and AI. In *Proceedings of CHI*.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 832–840, Sofia, Bulgaria. Association for Computational Linguistics.

Christopher J. Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. 2018. Measuring the effects of data parallelism on neural network training.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. 2022. knn-prompt: Nearest neighbor zero-shot inference.

Gayatri Chakravorty Spivak. 2003. Can the subaltern speak? *Die Philosophin*, 14(27):42–58.

Ann Laura Stoler. 2009. *Along the archival grain : epistemic anxieties and colonial common sense*. Book collections on Project MUSE. Princeton University Press, Princeton, NJ.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019a. How to fine-tune BERT for text classification? In *CCL*.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019b. Lamol: Language modeling for lifelong language learning.

Dan Svenstrup, Jonas Meinertz Hansen, and Ole Winther. 2017. Hash embeddings for efficient word representations.

Swabha Swayamdipta, Matthew Peters, Brendan Roof, Chris Dyer, and Noah A Smith. 2019. Shallow syntax in deep water. arXiv:1908.11047.

Anissa Tanweer, Emily Kalah Gade, PM Krafft, and Sarah K Dreier. 2021. Why the data revolution needs qualitative thinking. *Harvard Data Science Review*.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science.

Tan Thongtan and Tanasanee Phienthrakul. 2019. Sentiment classification using document embeddings trained with cosine similarity. In *ACL SRW*.

Yonglong Tian, Olivier J. Henaff, and Aaron van den Oord. 2021. Divide and contrast: Self-supervised learning from uncurated data.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning. arXiv:1806.02847.

Michel-Rolph Trouillot. 1995. *Silencing the past: Power and the production of history*. Beacon Press.

Marlon Vanegas, Juan Restrepo, Yurley Zapata, Giovany Rodríguez, Luis Cardona, and Cristian Muñoz. 2016. Linguistic discrimination in an English language teaching program: Voices of the invisible others. *Íkala, Revista de Lenguaje y Cultura*, 21.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Fred Vultee. 2012. A paleontology of style. *Journalism Practice*, 6(4):450–464.

Claudia Wagner, David Garcia, Mohsen Jadidi, and Markus Strohmaier. 2015. It's a man's Wikipedia? Assessing gender inequality in an online encyclopedia. In *Proceedings of the AAAI conference on web and social media*.

Jue Wang, Binhang Yuan, Luka Rimanic, Yongjun He, Tri Dao, Beidi Chen, Christopher Re, and Ce Zhang. 2022a. Fine-tuning language models over slow networks using activation compression with guarantees.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex Wade, Kuansan Wang, Nancy Xin Ru Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. Cord-19: The covid-19 open research dataset.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj

Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks.

Marlies van der Wees, Arianna Bisazza, Wouter Weerkamp, and Christof Monz. 2015. What's in a domain? Analyzing genre and topic differences in statistical machine translation. In *ACL*.

Jason Weismueller, Paul Harrigan, Kristof Coussement, and Tina Tessitore. 2022. What makes people share political content on social media? The role of emotion, authority and ideology. *Computers in Human Behavior*, 129:107150.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzm'an, Armand Joulin, and Edouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. *ArXiv*, abs/1911.00359.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Matthew L Williams, Pete Burnap, and Luke Sloan. 2017. Towards an ethical framework for publishing Twitter data in social research: Taking into account users' views, online context and algorithmic estimation. *Sociology*, 51(6):1149–1168.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. arXiv:1910.03771.

Mitchell Wortsman, Suchin Gururangan, Shen Li, Ali Farhadi, Ludwig Schmidt, Michael Rabbat, and Ari S. Morcos. 2022. lo-fi: distributed fine-tuning without communication.

Dustin Wright and Isabelle Augenstein. 2020. Transformer based multi-source domain adaptation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7963–7974, Online. Association for Computational Linguistics.

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.

Hu Xu, Bing Liu, Lei Shu, and Philip Yu. 2019a. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *NAACL*.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019b. Review conversational reading comprehension. arXiv:1902.00821v2.

Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2021. Bot-adversarial dialogue for safe conversational agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2950–2968, Online. Association for Computational Linguistics.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. 2021. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, volume 34, pages 17084–17097. Curran Associates, Inc.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yelp Reviews. Yelp reviews.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence.

Binhang Yuan, Yongjun He, Jared Quincy Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy Liang, Christopher Re, and Ce Zhang. 2022. Decentralized training of foundation models in heterogeneous environments.

Binhang Yuan, Yongjun He, Jared Quincy Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy Liang, Christopher Re, and Ce Zhang. 2023. Decentralized training of foundation models in heterogeneous environments.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019a. Defending against neural fake news. In *NeurIPS*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019b. Defending against neural fake news. In *NeurIPS*.

Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin, and Jianqiang Zhao. 2018. Multi-domain neural machine translation with word-level domain context discrimination. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 447–457, Brussels, Belgium. Association for Computational Linguistics.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2023. Instruction tuning for large language models: A survey.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level convolutional networks for text classification.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS*.

Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. 2019. Curriculum learning for domain adaptation in neural machine translation. In *NAACL*.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing.

Jun Zhu, Amr Ahmed, and Eric P. Xing. 2012. Medlda: Maximum margin supervised topic models. *Journal of Machine Learning Research*, 13(74):2237–2278.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015a. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of ICCV*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015b. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*.

Caleb Ziems, William Held, Jingfeng Yang, Jwala Dhamala, Rahul Gupta, and Diyi Yang. 2023. Multi-value: A framework for cross-dialectal english nlp.

Computing Infrastructure	56 Intel Xeon CPU Cores
Number of search trials	100
Search strategy	uniform sampling
Best validation F1	90.4

Hyperparameter	**Search space**	**Best assignment**
regularization	*choice*[L1, L2]	L1
C	*uniform-float*[0, 1]	0.977778
solver	64	liblinear
tol	*loguniform-float*[10e-5, 10e-3]	0.000816
ngram range	*choice*["1 2", "1 3", "2 3"]	"1 2"
random state	*uniform-int*[0, 100000]	44555
tokenization	whitespace	whitespace
vectorization	hashing	hashing
remove stopwords	*choice*[Yes, No]	No

Table A.1: Hyperparameter search space and best assignments for our re-implementation of the GPT-3 quality filter.

Chapter A

Appendix

A.1 Chapter 2 Appendix

A.1.1 Quality Filter Hyperparameters

We display the hyperparameters of our logistic regression classifier (reproduction of the filter developed by Brown et al. 2020) in Table A.1.

A.2 Chapter 3 Appendix

A.2.1 Related Work

Table A.2 shows which of the strategies for continued pretraining have already been explored in the prior work from the Related Work (§3.5). As evident from the table, our work compares various strategies as well as their interplay using a pretrained language model trained on a much more heterogeneous pretraining corpus.

A.2.2 State of the Art

In this section, we specify the models achieving state of the art on our tasks. See the caption of Table 3.5 for the reported performance of these models. For ACL-ARC, that is SciBERT Beltagy et al. [2019], a BERT-base model for trained from scratch on scientific text. For ChemProt and SciERC, that is

	DAPT Domains (if applicable)	Tasks	Model	DAPT	TAPT	DAPT + TAPT	kNN-TAPT	Curated-TAPT
This Paper	biomedical & computer science papers, news, reviews	8 classification tasks	RoBERTa	✓	✓	✓	✓	✓
Aharoni and Goldberg [2020a]	-	NMT	DistilBERT + Transformer NMT	-	-	-	similar	-
Alsentzer et al. [2019]	clinical text	NER, NLI, de-identification	(Bio)BERT	✓	-	-	-	-
Chakrabarty et al. [2019]	opinionated claims from Reddit	claim detection	ULMFiT	✓	✓	-	-	-
Chronopoulou et al. [2019]	-	5 classification tasks	ULMFiT†	-	similar	-	-	-
Han and Eisenstein [2019]	-	NER in historical texts	ELMo, BERT	-	✓	-	-	-
Howard and Ruder [2018]	-	6 classification tasks	ULMFiT	-	✓	-	-	-
Khandelwal et al. [2020]	-	language modeling	Transformer LM	-	-	-	similar	-
Lee et al. [2019]	biomedical papers	NER, QA, relation extraction	BERT	✓	-	-	-	-
Logeswaran et al. [2019]	-	zero-shot entity linking in Wikia	BERT	-	✓	-	-	-
Mitra et al. [2020]	-	commonsense QA	BERT	-	✓	-	-	-
Phang et al. [2018]	-	GLUE tasks	ELMo, BERT, GPT	-	✓	-	-	-
Radford et al. [2018a]	-	NLI, QA, similarity, classification	GPT	-	similar	-	-	-
Sun et al. [2019a]	sentiment, question, topic	7 classification tasks	BERT	✓	✓	-	-	-
Swayamdipta et al. [2019]	-	NER, parsing, classification	ELMo	-	similar	-	-	-
Xu et al. [2019a]	reviews	RC, aspect extract., sentiment classification	BERT	✓	✓	✓	-	-
Xu et al. [2019b]	restaurant reviews, laptop reviews	conversational RC	BERT	✓	✓	-	-	-

Table A.2: Overview of prior work across strategies for continued pre-training summarized in Table 3.10. ULMFiT is pretrained on English Wikipedia; ULMFiT† on English tweets; ELMo on the 1BWord-Benchmark [newswire; Chelba et al., 2014a]; GPT on BookCorpus; BERT on English Wikipedia and BookCorpus. In comparison to these pretraining corpora, RoBERTa's pretraining corpus is substantially more diverse.

S2ORC-BERT Lo et al. [2020b], a similar model to SciBERT. For AGNews and IMDB, XLNet-large, a much larger model. For RCT, Cohan et al. [2019]. For HyperPartisan, Longformer, a modified Transformer language model for long documents Beltagy et al. [2020]. Thongtan and Phienthrakul [2019] report a higher number (97.42) on IMDB, but they train their word vectors on the test set. Our baseline

establishes the first benchmark for the HELPFULNESS dataset.

A.2.3 Experimental Setup

Preprocessing for DAPT The unlabeled corpus in each domain was pre-processed prior to language model training. Abstracts and body paragraphs from biomedical and computer science articles were used after sentence splitting using `scispaCy` [Neumann et al., 2019]. We used summaries and full text of each news article, and the entire body of review from Amazon reviews. For both news and reviews, we perform sentence splitting using `spaCy` [Honnibal and Montani, 2017].

Training details for DAPT We train ROBERTA on each domain for 12.5K steps. We focused on matching all the domain dataset sizes (see Table 3.1) such that each domain is exposed to the same amount of data as for 12.5K steps it is trained for. AMAZON reviews contain more documents, but each is shorter. We used an effective batch size of 2048 through gradient accumulation, as recommended in Liu et al. [2019b].

Training details for TAPT We use the same pretraining hyperparameters as DAPT, but we artificially augmented each dataset for TAPT by randomly masking different tokens across epochs, using the masking probability of 0.15. Each dataset was trained for 100 epochs. For tasks with less than 5K examples, we used a batch size of 256 through gradient accumulation.

Optimization We used the Adam optimizer Kingma and Ba [2015], a linear learning rate scheduler with 6% warm-up, a maximum learning rate of 0.0005. When we used a batch size of 256, we used a maximum learning rate of 0.0001, as recommended in Liu et al. [2019b]. We observe a high variance in performance between random seeds when fine-tuning ROBERTA to HYPERPARTISAN, because the dataset is extremely small. To produce final results on this task, we discard and resample degenerate seeds.

Implementation Our LM implementation uses the `HuggingFace transformers` library [Wolf et al., 2019][1] and `PyTorch XLA` for TPU compatibility.[2] Each adaptive pretraining experiment was performed on a single v3-8 TPU from Google Cloud.[3] For the text classification tasks, we used `AllenNLP` Gardner et al.

[1] https://github.com/huggingface/transformers
[2] https://github.com/pytorch/xla
[3] http://github.com/allenai/tpu-pretrain

		Data Sample Unseen During DAPT				
		PT	BIOMED	CS	NEWS	REVIEWS
{410mm[DAPT]	ROBERTA	**1.19**	1.32	1.63	**1.08**	2.10
	BIOMED	1.63	**0.99**	1.63	1.69	2.59
	CS	1.82	1.43	**1.34**	1.92	2.78
	NEWS	1.33	1.50	1.82	1.16	2.16
	REVIEWS	2.07	2.23	2.44	2.27	**1.93**

Table A.3: ROBERTA's (row 1) and domain-adapted ROBERTA's (rows 2–5) masked LM loss on randomly sampled held-out documents from each domain (lower implies a better fit). PT denotes a sample from sources similar to ROBERTA's pretraining corpus. The lowest masked LM for each domain sample is boldfaced.

[2018]. Following standard practice Devlin et al. [2019b] we pass the final layer `[CLS]` token representation to a task-specific feedforward layer for prediction.

A.2.4 Analysis of Cross-Domain Masked LM Loss

In Section §3.2.2, we provide ROBERTA's masked LM loss before and after DAPT. We display cross-domain masked-LM loss in Table A.3, where we evaluate masked LM loss on text samples in other domains after performing DAPT.

We observe that the cross-domain masked-LM loss mostly follows our intuition and insights from the paper, i.e. ROBERTA's pretraining corpus and NEWS are closer, and BIOMED to CS (relative to other domains). However, our analysis in §3.2.1 illustrates that REVIEWS and NEWS also have some similarities. This is supported with the loss of ROBERTA that is adapted to NEWS, calculated on a sample of REVIEWS. However, ROBERTA that is adapted to REVIEWS results in the highest loss for a NEWS sample. This is the case for all domains. One of the properties that distinguishes REVIEWS from all other domains is that its documents are significantly shorter. In general, we find that cross-DAPT masked-LM loss can in some cases be a noisy predictor of domain similarity.

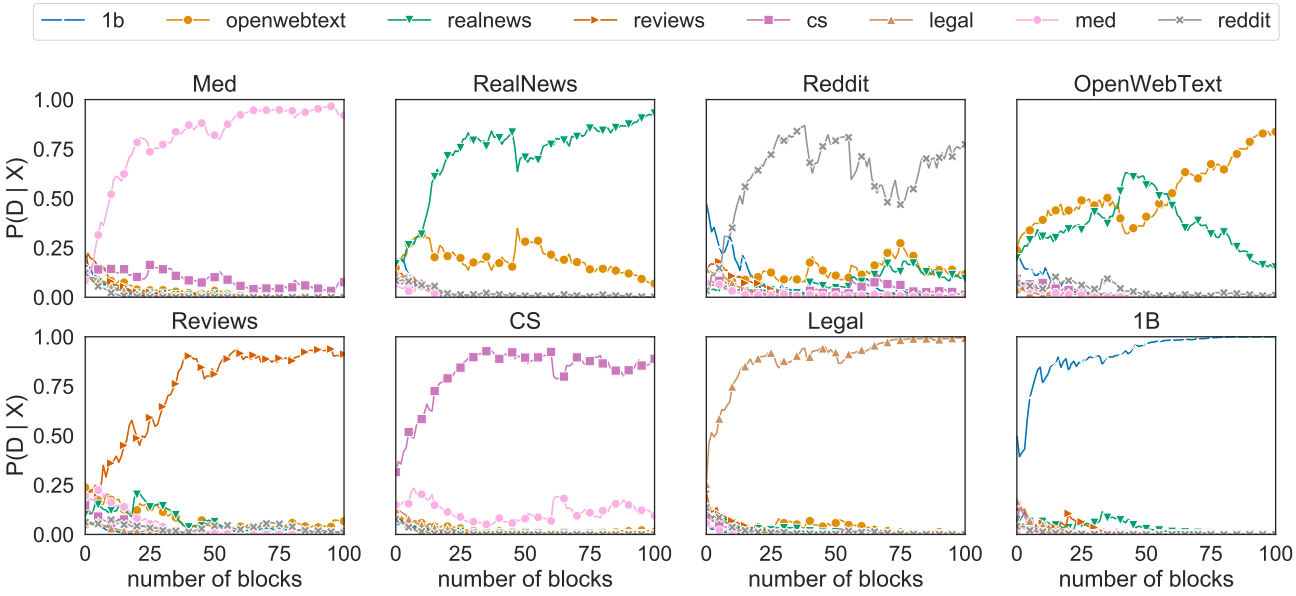

Figure A.1: Calculated domain posteriors for 8 training domains.

		Parameters			
		125M	350M	760M	1.3B
DENSE-DAPT	T	+70.1%	+21.4%	+16.7%	+20.6%
	N	–55.1%	–46.6%	–38.3%	-44.4%

Table A.4: Average change in perplexity in training (T) and novel (N) domains after DENSE-DAPT. Negative values indicate better performance relative to the original DENSE LM. While average perplexity in the novel domains decreases more for DENSE-DAPT, this comes at the cost of a significant deterioration in performance in training domains.

A.3 Chapter 4 Appendix

A.3.1 Domain Posterior Calculations

We track calculated domain posteriors over blocks of development data in Figure A.1 (training domains) and Figure A.2 (novel domains). The calculate domain posteriors are noisier for earlier blocks, stabilizing usually after around 50 blocks. For all experiments, we conservatively use 100 blocks of data to compute the domain posterior, though one may be able to accurately calcuate the domain posterior for some domains with less data.

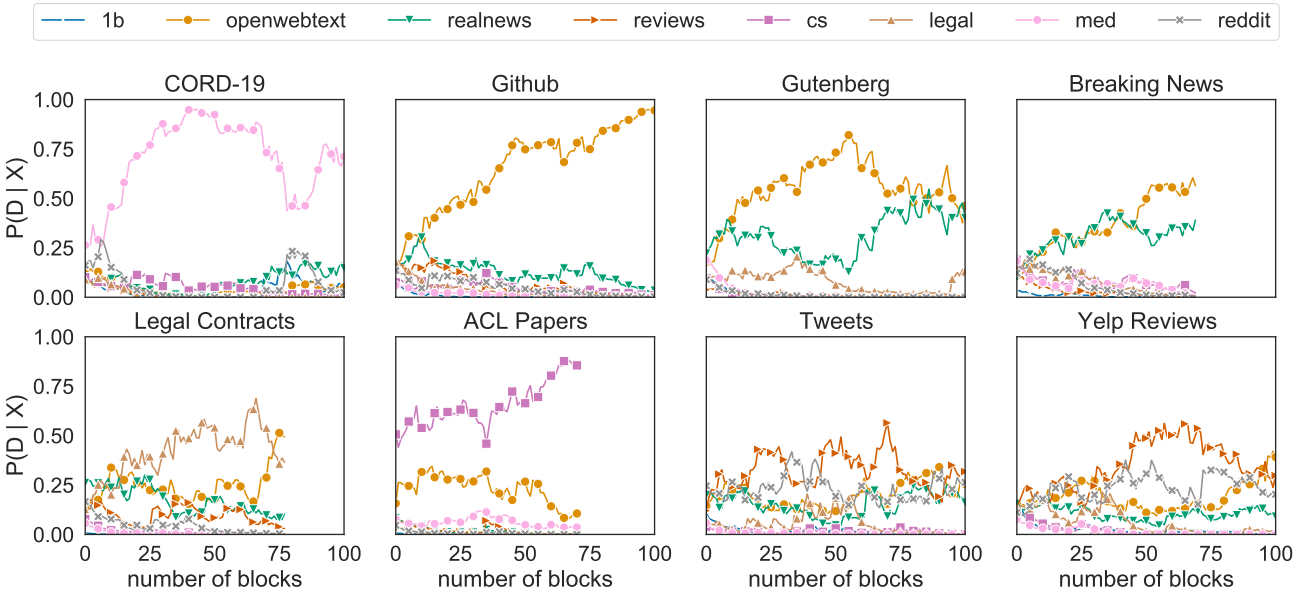

Figure A.2: Calculated domain posteriors for 8 novel domains.

A.3.2 Perplexity changes after DENSE-DAPT

In Table A.4, we display the average perplexity change after performing DENSE-DAPT on a new domain. We observe that across all model sizes, DENSE-DAPT improves performance in the novel domain, at the cost of a large performance hit in the training domains.

A.4 Chapter 5 Appendix

A.4.1 Clusters

Cluster	Term 0	Term 1	Term 2	Term 3	Term 4
0	fig	energy	al	et	field
1	data	patients	social	time	al

Table A.5: Top terms associated with each cluster in S2ORC (2 clusters)

Cluster	Term 0	Term 1	Term 2	Term 3	Term 4
0	eq	energy	quantum	equation	field
1	algorithm	image	model	data	noise
2	cells	cell	fig	al	protein
3	students	social	people	research	education
4	patients	patient	study	participants	children
5	fig	temperature	energy	surface	beam
6	user	data	node	network	nodes
7	et	al	stars	galaxies	velocity

Table A.6: Top terms associated with each cluster in S2ORC (8 clusters)

A.4.2 Comparing FLOP counts via training data size

To make fair comparisons across models with different numbers of ELMs, for a given number of clusters k and total GPU budget n, each ELM is allocated n/k GPUs. This keeps the total effective number of FLOPs fixed across models exposed to the same number of tokens. We can show this analytically; following Artetxe et al. 2021, we calculate the number of FLOPs to train a single ELM in our experiments:

$$F_{\text{ELM}(T,k)} = \frac{96lh^2T}{k}\left(1 + \frac{s}{6h} + \frac{V}{16lh}\right)$$

where T is the total training tokens (i.e., sequence length $\times$ batch size per GPU $\times$ number of GPUs), k is the number of clusters, l is the number of layers, h is the hidden dimension, s is the sequence length, and V is the vocabulary.

Therefore, the total cost in FLOPs to train k ELMs with a particular model architecture (e.g., OPT-1.3B) on T tokens in aggregate is equivalent to that of a single dense LM of the same architecture trained on T tokens:

$$k \cdot F_{\text{ELM}(T,k)} = F_{\text{ELM(T, 1)}}$$

This means that even though C-BTM trains k times more parameters than an equivalent dense model, it does so *at the same overall cost in FLOPs*. So, our comparisons of models with various numbers of ELMs are fair, as long as they have been exposed to the same number of training tokens and have the same underlying architecture for each ELM. Therefore, throughout the paper, we use training data size as a more interpretable metric of the overall compute budget.

Cluster	Term 0	Term 1	Term 2	Term 3	Term 4
0	students	learning	teachers	teaching	education
1	language	word	words	speech	english
2	network	node	nodes	networks	algorithm
3	study	risk	age	data	group
4	power	noise	channel	signal	frequency
5	quantum	spin	state	magnetic	states
6	participants	task	visual	stimulus	et
7	theorem	let	proof	lemma	set
8	training	learning	data	network	model
9	laser	beam	optical	fig	pulse
10	protein	genes	gene	dna	proteins
11	water	soil	data	et	al
12	algorithm	graph	vertices	problem	vertex
13	flow	velocity	field	magnetic	wave
14	user	data	users	information	service
15	stars	galaxies	et	al	star
16	quark	mass	gev	higgs	energy
17	et	al	brain	neurons	fig
18	stress	strain	surface	shear	fig
19	image	images	algorithm	object	segmentation
20	energy	electron	fig	eq	state
21	equation	field	eq	equations	theory
22	patients	patient	treatment	study	disease
23	model	time	robot	state	control
24	children	child	women	health	social
25	surface	nm	film	layer	graphene
26	patient	patients	pain	surgery	treatment
27	social	political	people	policy	public
28	social	participants	health	self	people
29	book	research	social	information	data
30	temperature	water	heat	phase	thermal
31	cells	cell	expression	mice	fig

Table A.7: Top terms associated with each cluster in S2ORC (32 Clusters).

Cluster	Term 0	Term 1	Term 2	Term 3	Term 4
0	like	just	time	new	great
1	new	information	use	business	services

Table A.8: Top terms associated with each cluster in C4 (2 clusters)

Cluster	Term 0	Term 1	Term 2	Term 3	Term 4
0	students	health	research	school	university
1	music	new	love	film	art
2	said	year	state	team	new
3	business	company	services	service	market
4	like	just	time	don	really
5	use	information	data	page	click
6	design	black	white	color	high
7	home	water	food	park	area

Table A.9: Top terms associated with each cluster in C4 (8 clusters)

A.4.3 Interpolating between empirical observations when comparing training costs and performance

We interpolate between our empirical observations using the following function, proposed in Artetxe et al. 2021:

$$c(t) = \exp(\log c_{lo}(t) + r(\log c_{hi}(t) - \log c_{lo}(t)))$$

where $r = \frac{t - t_{lo}}{t_{hi} - t_{lo}}$, t_{hi} and t_{lo} are the closest empirical performances to t and $c_{lo}(t)$ and $c_{hi}(t)$ are their corresponding training cost in FLOPs. We use this interpolation to compute the speedup factor $c_{dense}(t)/c_{cbtm}(t)$.

A.4.4 Effect of cluster balancing on cluster sizes

Using a held out set of 10K documents from C4, we ablate our balancing procedure from §5.1.1, and display a boxplot showing cluster sizes in Figure A.3.

Cluster	Term 0	Term 1	Term 2	Term 3	Term 4
0	site	page	website	web	search
1	art	design	image	images	gallery
2	health	care	children	child	medical
3	just	like	don	know	ve
4	data	information	software	use	management
5	love	great	like	just	really
6	game	team	games	season	play
7	information	email	contact	com	address
8	power	high	use	light	steel
9	students	school	student	education	learning
10	market	company	year	financial	tax
11	patients	treatment	body	cancer	pain
12	room	home	kitchen	bedroom	living
13	music	album	band	song	songs
14	car	vehicle	cars	new	road
15	park	hotel	beach	area	travel
16	day	event	year	time	wedding
17	service	services	quality	customer	best
18	book	books	read	writing	story
19	film	movie	story	new	like
20	black	white	color	dress	look
21	business	company	marketing	management	customers
22	university	research	development	education	science
23	city	community	county	said	police
24	sex	women	porn	dating	girls
25	products	product	online	quality	order
26	religion	life	people	jesus	time
27	water	skin	use	oil	like
28	said	politics	president	government	state
29	app	phone	video	mobile	casino
30	file	windows	use	click	software
31	food	add	recipe	minutes	wine

Table A.10: Top terms associated with each cluster in C4 (32 Clusters).

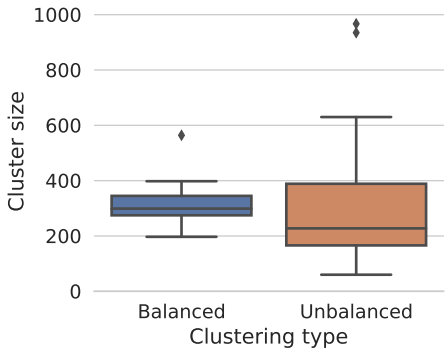

Figure A.3: Cluster balancing narrows the range, and increases the median size, of clusters (§5.6.2). Here, we ablate our balancing procedure from §5.1.1 on a 10K held-out documents in C4.

A.4.5 The Pile experiments

For the experiment in §5.6.4, we additionally train on The Pile, which is a publicly available corpus of diverse language modeling datasets. We use the filtered version from the OPT pretraining data [Zhang et al., 2022], and subselect 8 datasets of the 13 used for OPT pretraining, which enabled easier experimentation §5.6.4. These 8 datasets include: Common Crawl, HackerNews comments, Reddit comments[4], the Gutenberg corpus, STORIES corpus, OpenWebText2, Deepmind-Mathematics, and English Wikipedia. In aggregate, these datasets consist of 420M documents, totaling 116B BPE tokens. For evaluation, we sample an equal number of documents from each constituent dataset. We also train a k=8 clustering model on this dataset with one shard, or 1.5M documents.

5.2B Tokens		
8 Metadata	**8 Clusters**	**1 Cluster**
8.4	8.3	8.5

Table A.11: Experts trained with clusters perform slightly better than experts trained with metadata (§5.6.4). Here, we train each model for 5.2B tokens on 8 corpora of the Pile, and evaluate perplexity on a held out random sample of the constituent datasets. See §A.4.5 for more details on the dataset.

Temperature	Dense	2-cluster		8-cluster			
		top-1	top-2	top-1	top-2	top-4	top-8
0.01	**13.82**	**13.64**	13.60	**13.64**	13.50	13.49	13.49
0.05	-	-	13.51	-	13.32	13.25	13.25
0.1	-	-	**13.50**	-	**13.27**	**13.22**	**13.24**
0.2	-	-	13.54	-	13.29	13.34	13.50
0.3	-	-	13.59	-	13.33	13.45	13.72
0.5	-	-	13.64	-	13.38	13.59	13.97
1	-	-	13.69	-	13.43	13.73	14.19

Temperature	32-cluster					
	top-1	top-2	top-4	top-8	top-16	top-32
	13.55	13.45	13.45	13.45	13.45	13.45
	-	13.25	13.19	13.17	13.17	13.17
	-	13.16	**13.05**	**13.00**	**13.01**	**13.01**
	-	**13.14**	13.06	13.07	13.07	13.28
	-	13.17	13.15	13.27	13.54	13.78
	-	13.22	13.30	13.57	14.02	14.46
	-	13.30	13.49	13.89	14.46	15.04

Table A.12: Results with the dense (1-cluster), 2-cluster, 8-cluster, and 32-cluster C4 models trained on 84B tokens when varying the temperature (T) and $topk$ hyperparameters. Optimal performance for almost every model and $topk$ value is achieved at $T = 0.1$.

A.4.6 Sparsity

A.4.7 Downstream tasks

Tasks We experiment with six text classification tasks which span topic classification (AGNews; Zhang et al. 2016 and DBPedia; Lehmann et al. 2014); sentiment analysis (SST-2; Maas et al. 2011b, Amazon; Zhang et al. 2016, and Phrasebank Malo et al. 2014); and hatespeech detection (Twitter; Barbieri et al. 2020).

Performance Routing We introduce additional routing procedures for few-shot text classification, as the clustering method described in §5.1.2 ignores the significance of labels and does not take into account the context's word order, which may be crucial when the context contains demonstrations for a downstream task. Further, the specific ordering of in-context demonstrations is known to affect model performance Lu et al. [2022]. There is not sufficient evidence to suggest that the relative rank of different models on a task stays constant through these performance variances; thus it may be important to route to experts differently

[4]Reddit comments, like the rest of these datasets, are not collected by us but were part of the Pile [Gao et al., 2021], a publicly available third-party dataset.

depending on demonstration example order. To take into account the order of tokens in the context, we introduce 3 variations on routing, based on expert performance on the end task, which take inspiration from the mixture probability estimation methods of Gururangan et al. [2022]; Li et al. [2022].

To perform *Fixed Performance Routing with demonstrations and validation set examples*, we select 8 examples randomly from the validation set, such that there is no overlap with the 8 demonstration examples used in the context at test time. We concatenate the 8 demonstrations with one validation set context and evaluate the accuracy of each ELM on the sequence, repeating for each of the 8 examples from the validation set. The final routing probability distribution over experts for this task is determined by a softmax over the average accuracy of the model over the 8 examples. We use this fixed probability distribution for all test examples.

To perform *Fixed Performance Routing with only demonstrations*, we adapt the procedure above such that no validation examples are necessary. Instead, we take a random permutation of the 8 demonstration examples. We remove the label of the last, such that we effectively use the first 7 as demonstrations, and rely on the last example to estimate performance. We repeat this 8 times, generating a new random permutation each time, and once again take the softmax over the average accuracy of the model for each permutation, fixing this distribution for all test examples.

Finally, we have *Updating Performance Routing*, in which no estimations are done before test-time. At test time, we begin with a uniform probability over all experts, which we update with an exponential moving average with each test example: after each example (prepended with the 8 demonstrations), we update the expert probabilities with the softmax over the accuracy of each expert on that example. Once again, this distribution is fixed for all test examples.

Results Full results, in Table A.14, show that *Fixed Performance Routing with demonstrations and validation set examples* achieves the best performance overall, with optimal performance occurring at top-4 expert activation, which we also found in the language modeling results of §5.3.1. Both *Fixed Performance Routing* methods perform best with top-4 expert activation, and only suffer small performance degradations when reduced to top-1 expert activation. This aligns well with the patterns observed in §5.3.1, which further supports the incorporation of end task performance in routing when adapting to new tasks, even when we base this evaluation only on the demonstration examples – that is, without any additional data. We leave for

Task	1st	2nd	3rd
AGNews	*game, team, season*	*said, government, president*	*business, company, market*
DBPedia	*students, school, university*	*said, government, president*	*city, park, hotel*
SST-2	*book, film, life*	*love, family, great*	*music, art, band*
Amazon	*book, film, life*	*just, like, know*	*game, team, season*
Phrasebank	*business, company, market*	*service, customer, products*	*said, government, president*
Twitter	*data, software, download*	*click, website, page*	*just, like, know*

Table A.13: Top-terms of clusters associated with top-3 experts for each classification task (§5.4.2) By inspection, the highest probability experts are usually quite relevant to task's domain.

future work further tuning of the optimal settings for Performance Routing.

	Few-shot Text Classification Accuracy (%)						
↓ Model (params)	**AGNews** *Topic*	**DBPedia** *Topic*	**SST-2** *Sentiment*	**Amazon** *Sentiment*	**Phrasebank** *Sentiment*	**Twitter** *Hatespeech*	**Average**
Random chance	25.0	7.10	50.0	50.0	33.3	50.0	35.9
OPT (1.3B)	42.9	57.2	72.8	81.3	72.5	65.1	65.3
OPT (6.7B)	51.9	58.9	77.0	83.8	76.4	39.6	64.6
1-cluster (1.3B)	47.4	61.1	80.2	80.7	66.6	60.9	66.2
1-cluster (6.7B)	**68.1**	62.4	80.7	**84.9**	80.6	37.4	69.0
16-cluster			*Cluster Routing*				
top-1 (1.3B)	47.1	62.9	74.3	79.1	72.9	56.4	65.4
top-4 (5.2B)	49.3	62.3	80.0	81.3	78.7	61.3	68.8
top-16 (20.8B)	50.6	62.0	84.0	83.2	78.6	61.7	69.9
16-cluster			*Updating Performance Routing*				
top-1 (1.3B)	54.5	**63.4**	83.4	83.6	74.7	64.8	63.3
top-4 (5.2B)	51.6	61.5	88.6	83.7	**80.7**	**65.3**	68.8
top-16 (20.8B)	51.1	60.4	86.0	83.5	79.8	62.9	69.1
16-cluster			*Fixed Performance Routing (8 demonstrations)*				
top-1 (1.3B)	45.3	61.9	81.2	83.6	76.4	60.1	68.1
top-4 (5.2B)	51.2	60.9	81.4	83.0	80.5	60.9	69.6
top-16 (20.8B)	50.6	60.2	84.1	83.5	79.1	60.5	69.6
16-cluster			*Fixed Performance Routing (8 demonstrations + 8 validation examples)*				
top-1 (1.3B)	54.5	**63.4**	83.4	83.6	74.7	64.8	70.7
top-4 (5.2B)	51.6	61.5	**88.6**	83.7	**80.7**	**65.3**	**71.9**
top-16 (20.8B)	51.1	60.4	86.0	83.5	79.8	62.9	70.6

Table A.14: C-BTM models with performance routing achieve even better performance on downstream tasks (§A.4.7). We display performance of models on six text classification tasks, using eight demonstrations for each example and no additional fine-tuning. We compare our cluster routing method (described in §5.4) to variants of performance routing (described in §A.4.7). Fixed performance routing with 8 demonstrations and 8 validation examples usually gets the best performance on downstream tasks, consistently outperforming even the 6.7B parameter baselines. Fixed performance routing with top-4 inference always improves performance over using all experts, and top-1 inference does substantially better than the dense baselines at no additional inference costs. We include average performance across tasks for readability.