Between Language and Models: Rethinking Algorithms for Encoding and Decoding Text

Alisa Liu

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2026

Reading Committee:
Noah Smith, Co-Chair
Yejin Choi, Co-Chair
Pang Wei Koh

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Between Language and Models:
Rethinking Algorithms for Encoding and Decoding Text

Alisa Liu

Co-chairs of the Supervisory Committee:

Noah Smith
Computer Science and Engineering

Yejin Choi
Computer Science and Engineering

Language models are statistical models that operate over real numbers, taking in a sequence of continuous vectors as input and producing a new continuous vector as output. However, users of language models need to interface with human-readable text. This is made possible by two processes: at the input level, *tokenizers* encode text as a sequence of discrete symbols which correspond to learned embedding representations, and at the output level, *inference-time algorithms* convert real-valued predictions back into generated text. In this dissertation, I present three key contributions towards better algorithms that govern this process of encoding and decoding text.

First, to bridge the divergent approaches of human annotation and synthetic generation for data creation, I introduce a novel approach that combines the creativity of language model outputs with the reliability of human annotation in order to create data that is both diverse and high-quality. When applied to the task of natural language inference (NLI), we find that models trained on our dataset achieves significantly stronger generalization to out-of-domain test sets compared to existing NLI datasets.

Next, I present a flexible and lightweight approach for modifying language model behavior at inference-time. The method works by intervening on the unnormalized scores predicted over the vocabulary, using

small LMs that exemplify desirable or undesirable attributes in a product-of-experts fashion. I demonstrate that this can achieve the result of finetuning a language model, but by accessing only a language model's prediction over the vocabulary, not its parameters. This enables lightweight customization of API-based models, which is useful in a variety of practical settings and would not be possible with traditional finetuning.

Finally, I turn to the tricky topic of tokenization, where de facto methods have not changed meaningfully in the last decade. In particular, in essentially all language models today, tokenization occurs at the level of *subwords*, meaning that tokens are parts of words not bridging whitespace. Here, I introduce an algorithm for "superword" tokenization, extending traditional subword tokenization to include tokens spanning multiple words. In experiments, we demonstrate empirically that models pretrained from scratch with superword tokenization achieve stronger performance on downstream tasks while also being significantly more efficient at inference-time.

Together, these works demonstrate the rich space of possibilities for improving how language models interface with natural language. To conclude, I outline directions for future work.

Acknowledgements

I would have loved for my PhD to last forever, but reality is not so kind. Fortunately, before they nudged me to move on, my advisors guided me through the most fulfilling academic journey I could have asked for. Working with Yejin and Noah has been the privilege of a lifetime. Their intelligence, vision, and kindness continue to baffle me. They always saw clearly into the core of what I was working on and how to shape it into an impactful paper. They put their students first, prioritizing my growth as an independent researcher over my productivity. From day one, they never made me feel like I had to prove myself, so that I could focus all my attention on doing my best work. Noah was a remarkably steady force during the PhD; he gave me his time week after week through every stage of my PhD, during the periods of aimless exploration, the highest-momentum parts of a project, and finally a soul-crushing job search. Yejin has a remarkable way of seeing our field, and ensured I would periodically step back from the details to reflect on where our ever-evolving field is heading. I am indebted to both of you forever.

If not for mentors during my undergraduate years, I would never have learned about the world of research or thought about starting a PhD, and life would be so much poorer for it. Thank you to Professor Doug Downey and Professor Bryan Pardo, from the bottom of my heart, for welcoming me into your labs. They trusted me with responsibilities that I grew into and treated me like a PhD student before I was one. Thank you to their students, Mike D'Arcy, Jared Fernandez, Prem Seetharaman, Max Forbes, Ethan Manilow, and others, who were my role models and taught me the ropes of research.

I found my people at UW CSE. Because of the pandemic, my first PhD community was the CSE discord – it was wonderful playing online games with Jon Hayase, Kentrell Owens, Audrey Seo, Joseph Jaeger, Aditya Kusupati, Ofir Press, and many others. I thank Ewin Tang and Kentrell Owens for their wonderful friendship. I miss the days when we would see each other every other day and loiter in each other's

homes. Ewin and Kentrell very much coached me through both the career and social decisions of my PhD. Their graduation destroyed my illusion that the PhD is forever, but I tried my best to forge ahead on my own in my last two years. Thank you to the CSE mafia community for the late nights of brain-bending schemes: Joseph Jaeger, Erin Wilson, Matt Johnson, Jason Hoffman, Sam Ross, Alexandra Michael, Audrey Seo, Oscar Sprumont, Sidd Vaidynathan, Brandon Hayes, Willie Agnew, Chris Nagle, Joe Eckert. Thank you to Xiaochuang Han, Weijia Shi, Jiacheng Liu, Rock Pang, and Preston Jiang for our many late nights playing cards, the hysterical laughter, the huge meals we shared. Thank you to Lucille Njoo for making sure I went outside. Thank you to Jon Hayase and Vivek Ramanujan for the dumpling nights, bike rides, and brainstorming sessions. One day we will make our startup happen. How lucky I am to have made so many lifelong friends.

UW NLP is a wonderful place to do a PhD. Maarten Sap, Julian Michael, Ari Holtzman, Swabha Swayamdipta, Peter West, Alane Suhr, Ofir Press, Tim Dettmers, and Rowan Zellers were all mentors to me in my early years. I am grateful for the members of Noah's ARK and xlab who I shared my PhD journey with: Liwei Jiang, Melanie Sclar, Jaehun Jung, Ximing Lu, Saadia Gabriel, Jillian Fisher, Jiacheng Liu, Ben Newman, Taylor Sorensen, James Park, Oreva Ahia, Victoria Ebert, Gonçalo Faria, Phillip Keung, Ian Magnusson, Rahul Nadkarni, Luiza Pozzobon, Emily Reif, Weijia Shi, Nazif Tamer, Guang Yang. I had the privilege of collaborating with brilliant colleagues: Xiaochuang Han, Jon Hayase, Tomasz Limisiewicz, Artidoro Pagnoni, Yushi Hu, Yizhong Wang, Ofir Press, Ximing Lu, and many others. Because of the vibrant NLP community here, I never felt lonely doing reserch: no matter what I was thinking about, there was always someone to talk to who had thought about it more. I am especially grateful to Jon, who is best described as my "serial collaborator" – once we started putting our brains together, we never stopped. With Jon, research was pure fun. Thank you for your humor, patience, and of course, your giant brain. I'm so proud of the work we did together.

During my PhD I worked with many brilliant undergraduate students, who had fresh perspectives on our field and an insatiable appetite for learning: Margaret Li, Jaechan Lee, Andrew Shaw, Hao Xu, Brian Zheng. They showed me the joy of mentorship. It is my great privilege to have played a small part in fostering their love for research, and I know that each of them will go on to accomplish great things.

I would like to thank the members of my committee, Pang Wei Koh and Shane Steinert-Threlkeld. In

this short period of time in which I received their mentorship, I have already learned so much from the way they think.

Thank you to the CSE advising staff, Elise Dorough, Joe Eckert, Chris Nagle, and Les Sessions, the hidden heroes behind every PhD student's journey. They made the administrative parts seamless, fostered an incredible community, and looked out for each and every student so that we could reach our research potential. I will also miss playing Among Us and Mafia with them.

Thank you to my parents for ensuring that I could always pursue education to the limits of my abilities. For believing in me even when in their eyes, I was not nearly as stressed or working as hard as they expected for a PhD student. I have many close friends who kept me tethered and saw me through all the periods of exorbitant confidence and existential despair: Stephanie You, Irem Onalan, Amy Qin, Joseph Rodgers, Rolanda Fu, among many others. Finally, I owe my biggest gratitude to Alex. From our undergraduate to our doctorate years, school was full of joy because we were in it together. Thank you for your undying, unconditional support, for wanting my dreams to come true as much as I did, for keeping me functioning on deadline weeks. This would not have been the same PhD without you.

DEDICATION

to Alex; school is always more fun with you

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Language models are statistical models that operate over real numbers, taking in a sequence of continuous vectors as input and producing a new continuous vector as output. However, users of language models need to interface with human-readable text, which is fundamentally discrete. This conversion between natural language and continuous data is governed by two processes: at the input level, *tokenizers* encode text as a sequence of discrete symbols which correspond to learned embedding representations, and at the output level, *inference-time algorithms* convert real-valued predictions back into generated text.

In this dissertation, I improve how language models interface with natural language by developing new algorithms for encoding and decoding text. In particular, this dissertation covers the following contributions:

- A novel approach for combining the diversity of language models and the reliability of human annotators for data creation. (Chapter 2)

- A lightweight and flexible algorithm that intervenes on language model predictions over the vocabulary to achieve new objectives specified at inference time. (Chapter 3)

- A "superword" tokenizer training algorithm that learns a tokenizer containing both subwords and superwords, improving the representation of text. (Chapter 4)

1.1 Dissertation outline

The remainder of the dissertation is structured as follows.

1.1.1 Inference-time algorithms: from probabilities to text

Inference-time algorithms turn real-valued predictions from language models into natural language outputs. This generally happens by first normalizing the raw predicted scores over the vocabulary (the *logits*) into a probability distribution, then sampling a next-token prediction. Here, we present new methods that modify language model outputs through human (Chapter 2) or model (Chapter 3) collaboration in order to achieve more desirable generations.

Chapter 2: Human-AI collaboration for data creation In this chapter, we propose an approach for data creation that combines the generative strength of language models with the evaluative strength of human annotators. We observe that while human annotators are reliable for writing correct examples, crafting diverse and creative examples at scale is extremely challenging. As a result, crowdworkers often resort to a limited set of heuristics for speed at the expense of diversity. On the other hand, language models are capable of producing diverse and fluent output, especially given variation in prompting.

Therefore, we introduce a novel approach for data creation wherein language models create new examples targeting under-represented areas of the training data, then human annotators (optionally) revise the generation for quality and assign a gold label. We demonstrate the effectiveness of our approach on the task of natural language inference (NLI), which determines whether a premise entails (i.e., implies the truth of) a hypothesis. Remarkably, we find that despite being much smaller in size than existing resources, training on our new dataset, WANLI, improves model performance on eight different out-of-domain test sets. In analysis, we show that our dataset is more diverse and contains fewer spurious correlations compared to existing NLI datasets.

Chapter 3: Test-time adaptation with experts and anti-experts In this chapter, we introduce a test-time method for intervening on model-predicted logits to satisfy new objectives on-the-fly, providing a flexible and lightweight approach for modifying model behavior. Our key idea is to modify the predictions of a large pretrained model by combining its predictions with those of small models that exemplify

desirable or undesirable attributes, which we dub "experts" and "anti-experts," respectively. This was the first demonstration of the viability of combining logit-level predictions of multiple LMs through arithmetic, an idea that has since been used extensively in diverse applications spanning text and vision modalities.

In this dissertation, I focus on one particular application of this idea to test-time finetuning. In particular, our goal is to achieve the same effect as finetuning, but by accessing only a model's prediction over the vocabulary, not its parameters. To do this, we tune a small model, then apply the difference between the predictions of the small tuned and untuned models to shift the original predictions of the base model in the direction of tuning, while retaining the benefits of larger scale pretraining.

1.1.2 Tokenization: from text to embeddings

Tokenizers are the lens through which language models view data. They segment text, in the form of a stream of characters, into a sequence of tokens in the LM's vocabulary. LMs are then trained to predict the next token given a sequence of tokens as context. Therefore, tokenization plays a fundamental role in language modeling by controlling the representation of the input that the model learns over. Despite this, the de facto methods for tokenization have not changed meaningfully in the last decade.

Chapter 4: Superword tokenization for language models Today, tokenization occurs near-universally at the level of *subwords*, meaning that tokens are *parts* of words that do not bridge whitespace. However, we observe that whitespace is not an inherently reliable delimiter of meaning: multi-word expressions ("*by the way*") function semantically as single units, different languages vary in the number of words needed to express the same concept ("*spacesuit helmet*" is "*Raumanzughelm*" in German), and, at the extreme, languages such as Chinese do not use whitespace at all. Including multi-word tokens in the LM vocabulary would compress text into fewer tokens, lowering the cost of training and inference, and may also offer representational advantages. In this chapter, we introduce SuperBPE, a superword tokenization algorithm that learns a vocabulary of both subword and "superword" tokens. Experiments pretraining 8B-parameter LMs from scratch demonstrated that models trained with SuperBPE tokenizers consistently outperform their subword counterparts by a large margin, while also being significantly more efficient at inference time.

Prior Publications

The research presented in this dissertation is heavily based on the following prior publications.

1. **Alisa Liu**, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. WaNLI: Worker and AI Collaboration for Natural Language Inference Dataset Creation. *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022.

2. **Alisa Liu**, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning Language Models by Proxy. *First Conference on Language Modeling* (COLM), 2024.

3. **Alisa Liu***, Jonathan Hayase*, Valentin Hofmann, Sewoong Oh, Noah A. Smith, and Yejin Choi. SuperBPE: Space Travel for Language Models. *Second Conference on Language Modeling* (COLM), 2025.

The following prior publications directly laid the foundation for or built upon the above publications, and are also briefly mentioned in this dissertation.

1. **Alisa Liu**, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. DExperts: Decoding-Time Controlled Text Generation with Experts and Anti-Experts. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics* (ACL), 2021.

2. **Alisa Liu**, Zhaofeng Wu, Julian Michael, Alane Suhr, Peter West, Alexander Koller, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. We're Afraid Language Models Aren't Modeling Ambiguity. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing* (EMNLP), 2023.

3. Jaechan Lee, **Alisa Liu**, Orevaoghene Ahia, Hila Gonen, and Noah A. Smith. That was the last straw, we need more: Are Translation Systems Sensitive to Disambiguating Context?. *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

4. Skyler Hallinan, **Alisa Liu**, Yejin Choi, and Maarten Sap. Detoxifying Text with MaRCo: Controllable Revision with Experts and Anti-Experts. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics* (ACL), 2023.

5. Ruizhe Shi, Yifang Chen, Yushi Hu, **Alisa Liu**, Hannaneh Hajishirzi, Noah A. Smith, and Simon Shaolei Du. Decoding-Time Language Model Alignment with Multiple Objectives. *Advances in Neural Information Processing Systems* (NeurIPS), 2024.

6. Jonathan Hayase, **Alisa Liu**, Noah A. Smith, and Sewoong Oh. Sampling from Your Language Model One Byte at a Time. *Forty-Third International Conference on Machine Learning* (ICML), 2026.

7. Tomasz Limisiewicz, Artidoro Pagnoni, Srini Iyer, Mike Lewis, Sachin Mehta, **Alisa Liu**, Margaret Li, Gargi Ghosh, and Luke Zettlemoyer. Compute Optimal Tokenization. *arXiv preprint*, 2026.

8. Brian Siyuan Zheng, **Alisa Liu**, Orevaoghene Ahia, Jonathan Hayase, Yejin Choi, and Noah A. Smith. Broken Tokens? Your Language Model can Secretly Handle Non-Canonical Tokenizations. *Advances in Neural Information Processing Systems* (NeurIPS), 2025.

Chapter 2

Human-AI Collaboration for Data Creation

As much as large-scale crowdsourced datasets have expedited progress on various NLP problems, a growing body of research has revealed fundamental limitations in existing datasets: they are often flooded with repetitive and spurious patterns, rather than covering the broad range of linguistic phenomena required by the task [Bowman and Dahl, 2021]. This leads to models that seem to achieve human-level performance on in-domain test sets, yet are brittle when given out-of-domain or adversarial examples [Ribeiro et al., 2020; Glockner et al., 2018].

We attribute this problem to an inherent challenge in the crowdsourcing design—the prevalent paradigm for creating large-scale NLP datasets—where a relatively small number of workers create a massive number of free text examples. While human annotators are generally reliable for writing *correct* examples, crafting *diverse and creative* examples at scale can be challenging. Thus, crowdworkers often resort to a limited set of writing strategies for speed, at the expense of diversity [Geva et al., 2019; Gururangan et al., 2018]. When models overfit to such repetitive patterns, they fail to generalize to out-of-domain examples where these patterns no longer hold [Geirhos et al., 2020].

On the other hand, there has been remarkable progress in open-ended text generation based on massive language models [Brown et al., 2020; Raffel et al., 2020, i.a.]. Despite known deficiencies such as incoherence or repetition [Dou et al., 2021], these models often produce human-like text [Clark et al., 2021]

and show potential for creative writing tasks [Lee et al., 2022]. Importantly, these models are capable of replicating a pattern given just a few examples in context (Brown et al., 2020, GPT-3).

In this chapter, we introduce a novel approach for dataset creation which brings together the generative strength of language models and the evaluative strength of humans through **human and machine collaboration** (§2.1). The key insight of our approach is that language models can create new examples by replicating linguistic patterns that are valuable for training, without necessarily "understanding" the task itself. Illustrated in Figure 2.1, our pipeline starts with an existing dataset. We use dataset cartography from Swayamdipta et al. [2020] to automatically identify pockets of examples that demonstrate challenging reasoning patterns relative to a trained model. Using each group as a set of in-context examples, we leverage a pretrained language model to generate new examples likely to have the same pattern (see Table 2.1). We then propose a novel metric, building on dataset cartography, to automatically filter generations that are most likely to aid model learning. Finally, we validate the generated examples by subjecting them to human review, where crowdworkers assign a gold label and (optionally) revise for quality.

We demonstrate the effectiveness of our approach on the task of natural language inference (NLI), which determines whether a premise entails (i.e., implies the truth of) a hypothesis, both expressed in natural language. Despite being one of the most resource-available tasks in NLP, analysis and challenge sets repeatedly demonstrate the limitations of existing datasets and the brittleness of NLI models trained on them [Gururangan et al., 2018; Poliak et al., 2018; Tsuchiya, 2018]. Using MultiNLI [Williams et al., 2018] as our original dataset, we use our pipeline to create a dataset of 107,885 examples, which we call **W**orker-and-**AI** **NLI** (WANLI).[1]

Remarkably, empirical results demonstrate that *replacing* MultiNLI supervision with WANLI (which is 4 times smaller) improves performance on eight different out-of-domain test sets, including datasets that are converted to the NLI format from downstream tasks such as question-answering and fact verification (§2.2). This result holds even when augmenting MultiNLI with other NLI datasets and recently proposed augmentation sets. Moreover, including WANLI in the training data can help improve performance on certain in-domain test sets. We then analyze WANLI and show that it has fewer previously documented spurious correlations than MultiNLI (§2.3), and provide insights into the collaborative framework (§2.4).

[1]Pronounced wan-li like the Chinese characters 万理, as in *ten thousand reasoning*. A demo, data, and code are available at https://wanli.allenai.org/.

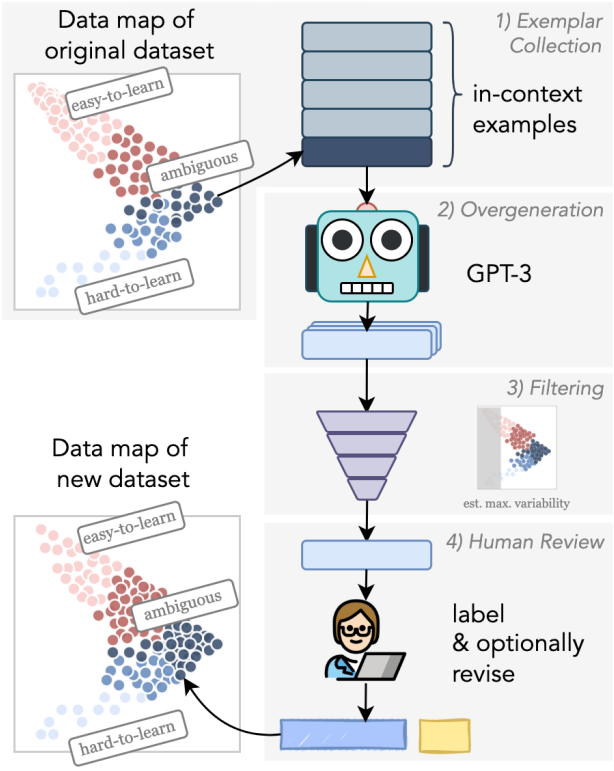


Figure 2.1: Pipeline for creating WANLI. Starting with a data map [Swayamdipta et al., 2020] of an existing dataset relative to a trained model, (1) we automatically identify pockets of data instances exemplifying challenging reasoning patterns. Next, (2) we use GPT-3 to generate new instances with the same pattern. These generated examples are then (3) automatically filtered via a metric we introduce inspired by data maps, and (4) given to human annotators to assign a gold label and optionally revise.

Our approach contrasts with previous instruction-based generation of dataset examples [Schick and Schütze, 2021; West et al., 2021], which require the model to understand the task from context, fundamentally limiting the complexity of generated output to what is accessible by the model. Moreover, our human-in-the-loop approach is *collaborative*, rather than *adversarial* [Dinan et al., 2019; Nie et al., 2020; Bartolo et al., 2020]. Overall, we leverage the best of both worlds: a powerful model's ability to efficiently generate diverse examples, and humans' ability to improve and ensure the quality of generations.

Our worker-AI collaborative approach is more scalable compared to the traditional crowdsourcing framework. Our approach is generalizable, allowing for rejuvenating datasets on many different classification tasks, especially when performance seems to stagnate due to overfitting to popular benchmarks [Recht et al., 2019]. Our work shows the promise of leveraging language models in a controlled way to aid the dataset creation process, and we encourage the community to think of dataset curation as an AI

challenge itself.

2.1 Method

We describe our four-stage approach for dataset creation based on worker and AI collaboration. In this work, we apply it to the task of natural language inference (NLI), which involves predicting whether a premise *entails*, *contradicts* or is *neutral* to a hypothesis. NLI has broad applicability in NLP: it has proven useful for pretraining [Clark et al., 2019; Phang et al., 2018], and can be applied to verify candidate answers in question-answering [Chen et al., 2021b] or factuality of generated summaries [Maynez et al., 2020].

Our approach requires as prerequisites an initial dataset $\mathcal{D}_0$ and a strong task model $\mathcal{M}$ trained on $\mathcal{D}_0$. We use MultiNLI [Williams et al., 2018], a large-scale multi-genre NLI dataset, as $\mathcal{D}_0$. We finetune RoBERTa-large [Liu et al., 2019] on MultiNLI for our task model $\mathcal{M}$.

As an overview, we first automatically **collect** groups of examples exemplifying challenging reasoning patterns in $\mathcal{D}_0$ relative to $\mathcal{M}$, using data maps (Swayamdipta et al., 2020; Stage 1, see §2.1.1). Then we **overgenerate** similar examples by leveraging the pattern replication capabilities of GPT-3 [Brown et al., 2020] (Stage 2; §2.1.2). While GPT-3 can generate examples efficiently, it may not reliably replicate the desired pattern and its output quality will not be uniform. We address this by automatically **filtering** the generated examples using a metric derived from data maps (Stage 3; §2.1.3). We finally subject the collected data to **human review**, in which crowdworkers optionally revise examples and assign gold labels (Stage 4; §2.1.4).

Dataset Cartography. A key component of our pipeline is inspired by data maps [Swayamdipta et al., 2020], which automatically reveal different regions in a dataset, w.r.t. the behavior of a classification model during training. These include *easy-to-learn* examples which the model consistently predicts correctly through training, *hard-to-learn* examples on which it is consistently incorrect, and *ambiguous* examples for which the model's confidence in the correct answer exhibits high *variability* across train epochs. Our pipeline focuses on *ambiguous* examples, which were shown to lead to more robust models. Additionally, ambiguous examples contain fewer spurious correlations [Gardner et al., 2021], suggesting that they capture under-represented counterexamples to spurious correlations. Indeed, such counterexamples take

Seed MultiNLI example	Generated WANLI Example	Label & Reasoning
P: ***5 percent*** probability that each part will be defect free. H: Each part has a ***95 percent*** chance of having a defect.	P: ***1 percent*** of the seats were vacant. H: ***99 percent*** of the seats were occupied.	*Entailment* Set complements
P: The artisans, tradespeople, and providers of entertainment (reputable and not so reputable) lived downtown on the reclaimed marshlands north and east, in the area still known as Shitamachi. H: ***The only place*** where artisans, tradespeople and entertainers could live was in the marshlands to the north and east.	P: At the time of the Revolution, the old port of Marseille was a great center of shipbuilding and commerce. H: ***The only place*** where ships were built was in the old port of Marseille.	*Neutral* Hypothesis introduces exclusivity
P: To the ***south***, in the Sea of Marmara, lie the woods and beaches of the Princes' Islands. H: In the ***north*** is the Sea of Marmara where there are mountains to climb.	P: From the park's ***southern entrance***, follow the avenue ***south*** to the Hotel de Ville. H: From the park's ***northern entrance***, follow the avenue ***north*** to the Hotel de Ville.	*Contradiction* Reversing cardinal directions
P: Democrats released documents indicating that Republicans sold big political donors meals with the party's leaders in federal buildings in 1995. H: ***It is illegal*** for a party to solicit products to donors.	P: In the late 1960s, students at a university in Wisconsin tried to organize a union. H: ***It was illegal*** for the students to organize a union.	*Neutral* Illegal things can happen
P: She ducked ***and*** parried the blow. H: She ducked ***to*** miss the blow.	P: She stepped on the brake ***and*** the car came to a stop. H: She stepped on the brake ***to*** stop the car.	*Entailment* Implied intention
P: To build a worldclass finance organization and help achieve better business outcomes, each of the organizations we examined ***set an agenda for transforming*** the finance organization by defining a shared vision -i.e. H: ***The transformation was a disaster*** and the entire organization had to be scrapped.	P: In order to help improve customer service, ***I suggested that they send a representative*** to our office to discuss our concerns. H: ***The representative*** sent to our office ***did not solve our problems*** and we lost a lot of business.	*Neutral* Intended goals may not actualize
P: Salinger ***wrote*** similar letters ***to*** other young female writers. H: Other young female writers ***received*** similar letters ***from*** Salinger as well.	P: The three schools ***have*** a number of students who are from families with no history of financial difficulties. H: Families with no history of financial difficulties ***send*** their children to the three schools.	*Entailment* Substituting a verb with a different subcategorization frame

Table 2.1: Seed MultiNLI examples, and corresponding WANLI examples generated by GPT-3. P stands for premise, H for hypothesis. The seed example is "ambiguous" according to the definitions of Swayamdipta et al. [2020], discussed in §2.1. The remaining in-context examples share the same pattern and are found using distance in `[CLS]` embeddings of a trained task model. The reasoning is a short description of the pattern we observe from the group, and which is successfully repeated in the generated example.

more epochs of training to learn and are crucial for generalization [Tu et al., 2020], providing a potential explanation for why they appear ambiguous across early epochs and lead to more robust models.

2.1.1 Stage 1: Collection of exemplars

In this stage, we automatically collect groups of examples from $\mathcal{D}_0$ which represent linguistic patterns we wish to include in the target dataset. We begin with a seed example $(x_i, y_i) \in \mathcal{D}_0$ belonging to the most ambiguous $p = 25\%$ relative to $\mathcal{M}$.[2]

To generate a new example with the same reasoning pattern, we wish to leverage the ability of GPT-3

[2]For exemplar collection, we exclude the telephone genre of MultiNLI, which consists of telephone conversation transcripts, due to their low fluency and ill-defined entailment relationships. During pilots, we found that generated examples mimicking telephone conversations would require crowdworkers to revise low-quality text for basic fluency.

[Brown et al., 2020] for in-context learning; hence, we need to first collect examples that test a similar kind of reasoning to x_i. To do this, we use the `[CLS]` token representation of each example relative to the *task* model $\mathcal{M}$, and find the $k = 4$ nearest neighbors via cosine similarity to x_i that *have the same label*. Detailed qualitative inspection shows that the nearest neighbors in this representation space tend to capture a human-interpretable similarity in the *reasoning* required to solve an example, rather than lexical or semantic similarity (examples in Table 2.1).

Han and Tsvetkov [2021] give another interpretation for this approach: for examples with the same label, the similarity of `[CLS]` token embeddings actually represents the similarity of *gradient updates* in the row of the final projection layer corresponding to that label. Thus, two examples are close if training on them would "update" the final layer of the model similarly.

By automatically identifying areas for augmentation, our method does not require any prior knowledge of challenging patterns and makes our method tractable for building on top of large-scale datasets. Nonetheless, exemplar collection could potentially be approached in different ways (e.g., through expert curation or category labels).

2.1.2 Stage 2: Overgeneration

Given an automatically extracted group of $k + 1$ examples from the original dataset $\mathcal{D}_0$, we construct a natural language context (prompt) for a left-to-right language model; in this work, we use GPT-3 Curie (the second-largest GPT-3 model). The prompt template we use is shown in Figure 2.2, where we order the examples in *increasing* similarity to the seed example.

Note that our method leverages GPT-3 in way that is distinct from its typical usage in few-shot settings, where given examples demonstrating a task, GPT-3 performs the task on a new, unlabeled example. Here, we instead give GPT-3 examples representing a particular *slice* of the task, and ask GPT-3 to *generate* a new example in the same slice.

For each context, we sample from GPT-3 to create $n = 5$ distinct examples. We use top-p decoding [Holtzman et al., 2020] with $p = 0.5$. Although generated examples at this stage could be assumed to share label of its $k + 1$ in-context examples, we instead consider the resulting dataset $\mathcal{D}_{\text{gen}} = \{x_i\}_i$ at the end of Stage 1 to be *unlabeled*.

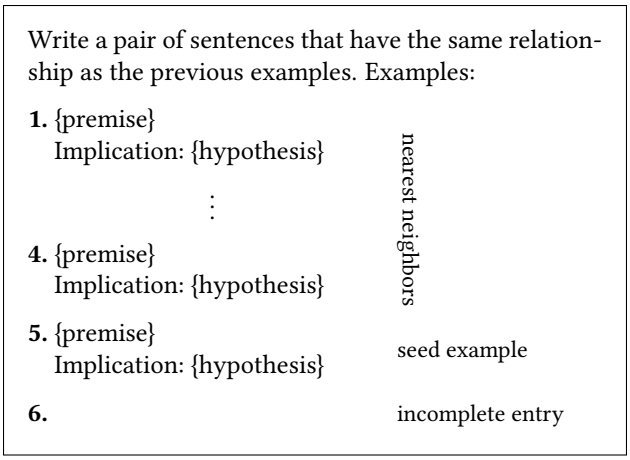


Figure 2.2: Prompt template instructing GPT-3 to generate a new example given a set of in-context examples. To separate the premise and hypothesis, the word "Implication" is used for entailment examples (shown here), "Possibility" for neutral examples, and "Contradiction" for contradiction examples.

2.1.3 Stage 3: Automatic filtering

In this step, we wish to filter generated examples from Stage 2 to retain those that are the most ambiguous with respect to $\mathcal{M}$. However, computing ambiguity for an example requires that it be a part of the original training set, whereas we wish to estimate the ambiguity of an *unlabeled* example *without* additional training. Thus we introduce a new metric called **estimated max variability**, which measures the worst-case spread of predictions on an example x_i across checkpoints of a trained model. Let E be the total epochs in training, $\mathcal{Y}$ the label set, and $p_{\theta^{(e)}}$ the probability assigned with parameters θ^e at the end of the e-th epoch. We define the estimated max variability as:

$$\sigma_i = \max_{y \in \mathcal{Y}} \sigma \left(\left\{ p_{\theta^{(e)}}(y \mid x_i) \right\}_{e \in E} \right), \tag{2.1}$$

where σ is the standard deviation function.

Concretely, we *retroactively* compute the prediction from each saved epoch of $\mathcal{M}$ on x_i. The only assumption made is that the single example, if it had been a part of the training set, would have made a negligible difference on each model checkpoint (at least as observed through its posterior probabilities). In taking a maximum across labels, we consider x_i to be ambiguous as long as $\mathcal{M}$ is undecided on *any* label $\in \mathcal{Y}$.

Split	Size	Label distribution (E/N/C)
Train	102,885	38,511 / 48,977 / 15,397
Test	5,000	1,858 / 2,397 / 745

Table 2.2: WANLI dataset statistics.

We first employ simple heuristics to discard examples exhibiting observable failure cases of GPT-3. Specifically, we discard examples where 1) the premise and hypothesis are identical, modulo punctuation or casing, 2) the generated example is an exact copy of an in-context example, 3) the example contains some phrases from the instruction (e.g., "pair of sentences"), or 4) the premise or hypothesis is shorter than 5 characters. Then, we compute the estimated max variability for the remaining examples with respect to $\mathcal{M}$, and retain an equal number of examples from each (intended) label class with the highest max variability, to create a dataset $\mathcal{D}_{\text{filtered}}$ that is half the size of $\mathcal{D}_{\text{gen}}$.

2.1.4 Stage 4: Human review

As the final stage of our pipeline, we recruit human annotators on Amazon Mechanical Turk to review each unlabeled example $x_i \in \mathcal{D}_{\text{filtered}}$. The annotator may optionally revise x_i to create a higher-quality example x_i', or let $x_i' = x_i$. Either way, they assign a label y_i. When revising examples, we asked annotators to preserve the intended meaning as much as possible through minimal revisions.[3] However, if an example would require a great deal of revision to fix *or* if it could be perceived as offensive, they should discard it. This results in the labeled dataset $\mathcal{D}_{\text{collab}} = \{(x_i', y_i)\}_i$.

Crowdworkers annotate a total of 118,724 examples, with two distinct workers reviewing each example. For examples that both annotators labeled without revision, we achieved a Cohen's κ of 0.60, indicating substantial agreement. To create the final dataset, we discard an example if *either* annotator chose to discard it, and we keep a revision only if *both* annotators revise an example (and choose a revision uniformly at random). When both annotators label the example as-is but choose different labels, we sample one of the two labels uniformly at random (see §2.4.2 for relevant discussion). This leads to a labeled dataset of 107,885 examples (90.87% of all annotated examples, with the remaining discarded). Of the labeled examples, 3.54% were revised.

[3] In pilots, we found that when annotators exercised too much freedom in revision, they often re-introduced the same artifacts that have been well-documented in NLI.

We randomly split the data into a train and test sets. Key dataset statistics are summarized in Table 2.2. Unlike MultiNLI, WANLI is not label-balanced; see §2.4.3 for a discussion.

In general, we believe the role of revision depends on the quality of machine-generated examples. Indeed, we need to strike a balance between leveraging human capabilities and avoiding the re-emergence of annotation artifacts that may come with too much freedom in revision.

2.2 Training NLI models with WANLI

We finetune different copies of RoBERTa-large [Liu et al., 2019] on different training sets, and evaluate each resulting model's performance on a large suite of NLI challenge sets. Given that the challenge sets were constructed independently of MultiNLI or WANLI, we consider them out-of-distribution (OOD) for both training datasets.

2.2.1 NLI test suite

The NLI challenge sets come from a wide array of domains, methodologies (e.g., crowdsourcing, expert curation, generation), and initial task formats (e.g., question-answering, fact verification).[4]

NLI Diagnostics [Wang et al., 2018] is a manually-curated test set that evaluates a variety of linguistic phenomena using naturally-occurring sentences from several domains.

HANS [McCoy et al., 2019] targets unreliable syntactic heuristics based on lexical overlap between the premise and hypothesis.

QNLI was adapted from the Stanford Question-Answering Dataset [Rajpurkar et al., 2016] by the GLUE benchmark [Wang et al., 2018]. Each example consists of a premise that is a sentence, and a hypothesis that is a question, which is entailed if the question is answered by the premise.

Winograd NLI was adapted by the GLUE benchmark from the Winograd Schema Challenge [Levesque et al., 2011], which tests correct coreference via common sense. To convert this dataset to NLI, an entailed hypothesis is formed by substituting a correct referent and a non-entailed hypothesis is formed by substituting an incorrect referent.

[4] We evaluate on the development set for every dataset, except for Winograd NLI, where we combine the train and development set for greater statistical power, and Adversarial NLI, where we use the test set as the labels were not hidden.

Adversarial NLI [ANLI; Nie et al., 2020] is an adversarially-constructed dataset where crowdworkers are instructed to write examples that stump existing models. Examples are collected in three rounds that progressively increase in difficulty, with model adversaries trained on MultiNLI, SNLI [Bowman et al., 2015], FEVER-NLI (discussed below), as well as ANLI sets from earlier rounds.

Natural Questions NLI [NQ-NLI, Chen et al., 2021b] is created from the Natural Questions QA dataset [Kwiatkowski et al., 2019]. The premise is a *decontextualized* sentence from the original context; the hypothesis consists of a question and answer candidate converted into declarative form.

FEVER NLI is adapted from the FEVER fact verification dataset [Thorne et al., 2018], and introduced along with ANLI. In each example, the premise is a short context from Wikipedia, and the hypothesis is a claim that is either supported (entailed), refuted (contradicted), or neither (neutral).

BIG-Bench NLI is a combination of four datasets from BIG-Bench [Srivastava et al., 2022] about entailment: Analytic Entailment, Epistemic Reasoning, Disambiguation QA, Presuppositions NLI.

2.2.2 Training datasets

In addition to stand-alone WANLI and MultiNLI, we also consider combining MultiNLI with other NLI datasets. We use the train sets of SNLI [Bowman et al., 2015], ANLI, and FEVER-NLI, as well as the augmentation set generated via TAILOR [Ross et al., 2022], which perturbed SNLI hypotheses to create examples with high lexical overlap between the premise and hypothesis, and the augmentation set Z-Aug [Wu et al., 2022], which was created by generating in-distribution examples and filtering them based on spurious correlations.

We consider two schemes for combining datasets $\mathcal{A}$ and $\mathcal{B}$: 1) **augmentation** ($\mathcal{A} + \mathcal{B}$), in which the two datasets are concatenated, and 2) **random replacement** ($\mathcal{A} \diamond \mathcal{B}$), where $|\mathcal{B}|$ examples from $\mathcal{A}$ are randomly swapped out and replaced with all examples from $\mathcal{B}$.

2.2.3 Results

Results are shown in Table 2.3. When comparing MultiNLI (MNLI) and WANLI alone, training a model on WANLI instead of MultiNLI leads to better performance on every test set we consider, including by 4% on Diagnostics, 11% on HANS, and 9% on Adversarial NLI. This is remarkable given WANLI is 4× smaller than

			Test Set								
			Diagnostics	**HANS***	**QNLI***	**WNLI***	**NQ-NLI***	**ANLI**	**FEVER-NLI**	**BIG-Bench***	**WANLI**
		Data size	1104	30K	5266	706	4855	3200	20K	3324	5000
	MNLI	393K	68.47	78.08	52.69	56.09	62.34	32.37	68.29	64.68	64.62
Training Set	MNLI + Tailor	485K	67.75	79.03	54.89	56.23	63.83	32.87	68.75	72.38	64.27
	MNLI + Z-Aug	754K	66.39	80.52	57.72	55.52	62.30	33.37	68.73	66.12	64.78
	MNLI ⋄ ANLI	393K	67.75	79.90	68.74	60.48	62.49	54.59	72.30	72.32	**65.96**
	MNLI + ANLI	556K	66.84	77.94	62.41	57.08	62.84	53.84	72.30	71.11	65.93
	MNLI ⋄ FEVER-NLI	393K	66.75	76.50	56.70	57.08	61.81	35.65	76.83	58.39	63.31
	MNLI + FEVER-NLI	601K	67.57	76.05	52.90	54.95	63.02	35.37	76.93	64.65	64.53
	MNLI + SNLI + ANLI	943K	68.75	78.65	63.38	58.49	62.94	54.21	72.02	71.05	65.10
	MNLI ⋄ **WANLI**	393K	71.01	83.10	77.00	61.89	62.94	36.46	**71.14**	76.17	75.49
	MNLI + **WANLI**	496K	71.64	82.00	68.40	60.05	63.21	36.78	70.79	70.81	75.26
	WANLI	103K	**72.73**	**89.28**	**81.40**	**67.28**	**64.18**	**41.12**	70.13	**85.19**	75.40

Table 2.3: Empirical comparison of different training sets for RoBERTa-large, for generalization to out-of-distribution (OOD) challenge sets. Gray cells mark settings that do not represent an OOD challenge. **Top**: Training on MultiNLI alone. **Middle**: Comparison of combination schemes with MultiNLI. We consider two data combination strategies, augmentation (+), and random replacement (⋄), where the resulting dataset size is unchanged. **Bottom**: Training sets that include WANLI. The highest accuracy on each test set (excluding gray cells) is bolded. Test sets with * contain two label classes: entailment and non-entailment.

	Test Set				
	Diagnostics	**HANS***	**ANLI**	**BIG-Bench***	**WANLI**
ANLI	65.67	80.58	55.21	77.10	63.85
ANLI + WANLI	**72.82**	**88.58**	**56.59**	**84.89**	**75.84**

Table 2.4: Comparison of whether including WANLI in the training data of ANLI improves in-domain test performance, when finetuning RoBERTa-large.

MultiNLI, and contains primarily machine-written examples.

A WANLI-trained model continues to outperform baselines that combine MultiNLI with other NLI datasets and augmentation sets, in every OOD setting. This includes when comparing to a model trained on 9× more data from three existing NLI datasets, MNLI + SNLI + ANLI. The consistent advantage of WANLI over datasets that include ANLI (e.g., MNLI + ANLI) is noteworthy, as ANLI's adversarial creation pipeline posed a much greater challenge for human workers, and used more existing resources to train model adversaries.

Quite surprisingly, training on WANLI alone also outperforms combining WANLI with MultiNLI. This reinforces that more data might not necessarily be better, especially when the data predominantly consists of easy-to-learn examples.

In addition to the OOD setting, we consider whether augmentation with WANLI can improve *in-domain*

test performance for another dataset (Table 2.4). Indeed, augmenting ANLI's train set with WANLI improves test accuracy on ANLI by 1.4%, while greatly aiding OOD test performance.

2.3 Artifacts in WANLI

We next investigate whether WANLI contains similar artifacts to MultiNLI.[5] We find that while WANLI contains fewer previously known spurious correlations, it has a distinct set of lexical correlations that may reflect artifacts in GPT-3 output.

2.3.1 Partial input models

Given that the task requires reasoning with both the premise and the hypothesis, a model that sees only one of the two inputs should have no information about the correct label. We reproduce the methodology from Gururangan et al. [2018] and train `fastText` classifiers to predict the label using partial input. After first balancing WANLI, a model trained on just the hypotheses of WANLI achieves 41.6% accuracy on the test set compared to 49.6% for MultiNLI, when restricted to the same size. A premise-only model trained on WANLI achieves an accuracy of 42.9%.[6]

2.3.2 Lexical correlations

Gardner et al. [2021] posit that all correlations between single words and output labels are spurious. We plot the statistical correlation for every word and label in Figure 2.3, after balancing WANLI and downsampling MultiNLI. We observe that WANLI also contains words with detectable correlations, suggesting that GPT-3 may have some artifacts of its own due to the slightly different templates and different sets of in-context examples for each label. Interestingly, the correlations tend to be a different set of words than for MultiNLI (other than "not" and "no"), with less interpretable reasons for correlating with a certain label (e.g., "second", "was").

[5] We note, however, that recent work has challenged whether artifacts based on partial input and lexical correlations in the dataset pose genuine robustness threats [Srikanth and Rudinger, 2022; Eisenstein, 2022].

[6] Unlike WANLI, each MultiNLI premise is associated with hypotheses from all three labels; a premise-only baseline is thus guaranteed to have no information about the label.

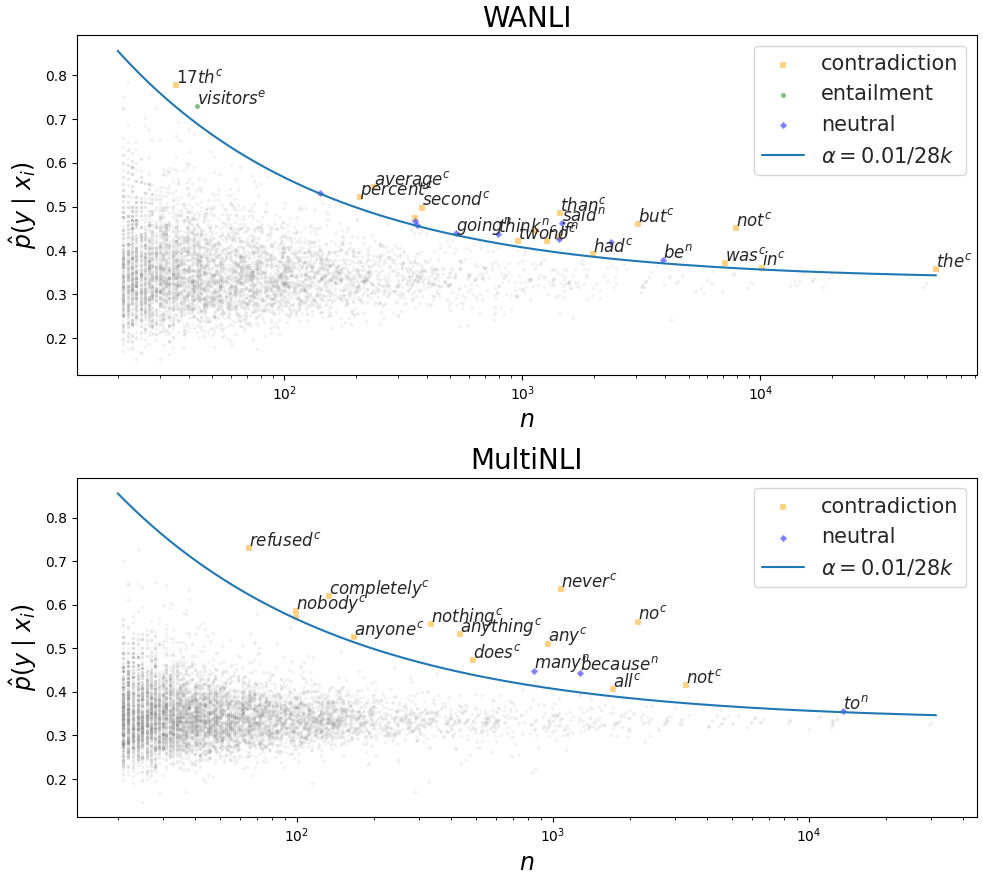


Figure 2.3: Competency problem-style statistical correlation plot between individual words and particular class labels, where the y-axis is the probability of label y given the presence of the word x_i, and the x-axis is the number of times word x_i appears in the data. All points representing (word, label) pairs above the blue line have detectable correlations [Gardner et al., 2021].

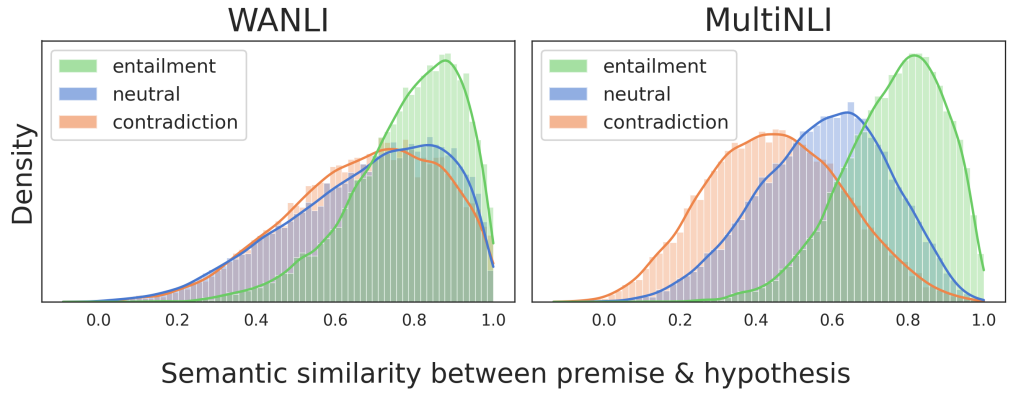


Figure 2.4: Semantic similarity between the premise and hypothesis, computed based on SBERT embeddings [Reimers and Gurevych, 2019]. The distributions for each label class are much more well-separated in MultiNLI than in WANLI.

2.3.3 Premise-hypothesis semantic similarity

We explore the semantic similarity between the premise and hypothesis within each label class using Sentence-BERT [Reimers and Gurevych, 2019]; these distributions are shown in Figure 2.4. In both MultiNLI and WANLI, entailed hypotheses are naturally most semantically similar to the premise. In MultiNLI, this is followed by neutral examples and then contradiction examples. In contrast, in WANLI there is much greater overlap in the three distributions, and those for neutral and contradiction examples are nearly indistinguishable. This suggests in WANLI, the semantic similarity between the premise and hypothesis provides less signal of the label.

2.4 What does WANLI show about the human machine collaboration pipeline?

We discuss observations from collecting WANLI that may shed insight for future work in the direction of collaborative dataset creation.

2.4.1 What kinds of revisions do annotators tend to make?

We find that revisions fall broadly into two categories: improving the fluency of the text, and improving the clarity of the relationship. The majority of revisions change the length only slightly, with 74%

of both premise revisions and hypothesis revisions changing the word count between −1 and +2 words. Fluency revisions often target well-documented issues with text generation, such as redundancy and self-contradiction. Clarity revisions often resolve ambiguities in the example that make the entailment relationship difficult (or impossible) to determine, such as ambiguous coreference or temporal references. We provide examples of revisions in Table 2.5.

Example	Label	Purpose of Revision
P: The power plant ~~It~~ is the only source of continuous electric power for the city. H: The power plant is very important for the city.	*Entailment*	Coreference resolution
P: It was a well-known fact that ~~it was a well-known fact that~~ the solution was well-known. H: The solution was well-known.	*Entailment*	Redundancy
P: This will be the first time the king has met the queen in person. H: The king has met the queen in person before.	*Contradiction*	Clarity
P: She walked with a light step, as if she were floating on air. H: She was floating on air~~, as if she were walking on air~~.	*Contradiction*	Coherence
P: There is a slight possibility that, if the same temperature data are used, the temperature of the Earth's surface in 1998 will be lower than the temperature of the Earth's surface ~~in 1998~~ now. H: The Earth's surface in 1998 was lower than the Earth's surface ~~in 1998~~ now.	*Neutral*	Self-contradiction
P: She had to go to the library to find out what the name of the street was. H: She already knew the name of the street.	*Contradiction*	Ambiguous temporal reference
P: A number of theories have been proposed to explain the decline of violence in modern society. H: Violence ~~will decline~~ has declined in modern society.	*Entailment*	Consistent tense

Table 2.5: Some examples of revisions that were done by annotators on examples generated by GPT-3.

2.4.2 What kinds of examples do annotators disagree on?

We randomly sampled 80 examples where the two annotators disagreed on the label (and neither revised nor discarded), and two of the authors separately annotated each one. Shockingly, the two authors agreed on the label only 49% of the time. Furthermore, in 12% of cases, all three labels were present among the four annotations. This suggests that disagreement is often due to true ambiguity rather than careless mislabeling, and we encourage future work to design richer annotation frameworks to uncover the source(s) of ambiguity.

Example	Labels	Ambiguity
P: According to the most recent statistics, the rate of violent crime in the United States has dropped by almost half since 1991. H: The rate of violent crime has not dropped by half since 1991.	*Entailment* *Contradiction*	Does "almost half" mean "not half" or "basically half"?
P: As a result of the disaster, the city was rebuilt and it is now one of the most beautiful cities in the world. H: A disaster made the city better.	*Entailment* *Neutral*	Do indirect consequences count?
P: It is a shame that the world has to suffer the pain of such unnecessary war. H: The world does not have to suffer such pain.	*Entailment* *Contradiction*	Is the scope of "has to" in the hypothesis given the war or not?
P: The original draft of the treaty included a clause that would have prohibited all weapons of mass destruction. H: The clause was removed in the final version of the treaty.	*Entailment* *Neutral*	Does the premise imply that the clause is no longer in the treaty?
P: If you can't handle the heat, get out of the kitchen. H: If you can't handle the pressure, get out of the situation.	*Entailment* *Neutral*	Is the premise to be interpreted literally or figuratively?
P: In a world of increasing uncertainty, the only certainty is that nothing is certain. H: There is no certainty in the world.	*Entailment* *Contradiction*	Self-contradictory but coherent premise

Table 2.6: Examples where two annotators assigned different labels. We find that many examples represent genuinely ambiguous cases rather than careless mislabels, echoing previous findings [Pavlick and Kwiatkowski, 2019].

We choose to keep examples with disagreement in the WANLI dataset because we believe that finetuning with one of multiple reasonable labels still provides valuable training signal.

2.4.3 How reliably does GPT-3 reproduce the in-context pattern?

One characteristic of WANLI is its imbalanced label distribution: even though the set of seed examples for generation was constructed to be balanced, after undergoing human labeling, only 15% of examples are given the contradiction label. We observe that contradiction patterns in in-context examples are generally much more challenging for GPT-3 to copy, likely because it was trained on (mostly) coherent sequences of sentences. More broadly, we find that more abstract reasoning patterns are harder for GPT-3 to mimic than patterns that involve simpler transformations.

Nonetheless, even when GPT-3 does not successfully copy the examples, the diverse set of in-context examples leads to a variety of creative output that may be challenging for human crowdworkers to achieve.

2.5 Related work

Crowdsourcing The scalability and flexibility of crowdsourcing has enabled the creation of foundational NLP benchmarks across a wide range of subproblems, and made it the dominant paradigm for data collection [Mihaylov et al., 2018; Rajpurkar et al., 2016; Huang et al., 2019; Talmor et al., 2019, i.a.]. Nonetheless, a growing body of research shows that resulting datasets may not isolate the key linguistic phenomena [Jia and Liang, 2017; Chen et al., 2016; Sugawara et al., 2020].

For crowdsourcing NLI datasets, where the annotator is given a premise and asked to write a hypothesis of each label [Bowman et al., 2015; Williams et al., 2018], the presence of annotation artifacts is especially well-studied [Gururangan et al., 2018; McCoy et al., 2019; Glockner et al., 2018]. Recent work attempted to remedy this through different data collection protocols but found negative results [Vania et al., 2020; Bowman et al., 2020], showing this is a hard problem requiring greater innovation.

Adversarial data collection In this paradigm, annotators are asked to produce examples on which current systems fail [Kiela et al., 2021; Talmor et al., 2021; Zellers et al., 2019, i.a.]. Beyond increasing annotator effort [Bartolo et al., 2020], adversarial methods have been challenged for not leading to better generalization on non-adversarial test sets [Kaushik et al., 2021] and decreasing data diversity [Bowman and Dahl, 2021]. Moreover, the resulting data has been shown to depend strongly on the adversaries, inhibiting a fair evaluation [Phang et al., 2021]. Finally, these approaches may produce examples beyond the scope of the task. For example, in Adversarial NLI [Nie et al., 2020], an estimated 58% of examples required "reasoning from outside knowledge or additional facts," which is arguably separate from the underlying problem of understanding semantic entailments. We argue that we can better leverage the strengths of machines and humans by having them collaborate rather than act as adversaries.

Dataset generation Another recent approach leverages language models toward fully automatic dataset creation [Schick and Schütze, 2021; Wu et al., 2022; West et al., 2021; Bartolo et al., 2021a, i.a.]. Removing human input may fundamentally limit the complexity of examples to phenomena already accessible by the model, when our goal is precisely to teach models more diverse phenomena. The most similarly-motivated work to ours, Lee et al. [2021], trains a data generator on "data-rich slices" of an existing dataset, and applies it to under-represented slices. However, they use labels or metadata to represent slices, leaving automatic

methods of identifying slices to future work.

Human-machine collaboration In terms of human-machine collaboration, Tekiroğlu et al. [2020] and Yuan et al. [2021] employ a language model to generate counter-narratives to hate speech and biographies, respectively, which are validated and revised by humans. This was for a generative task, and we complement their findings by showing that human-machine collaboration can also be useful for generating labeled datasets for robust classification models. Contemporary work [Bartolo et al., 2021b] finetunes a generative annotation assistant to produce question-answer pairs that humans can revise for extractive QA.

2.6 Conclusion

At the heart of dataset creation is distilling human linguistic competence into data that models can learn from. The traditional crowdsourcing paradigm takes the view that the best approach for this is to solicit people to write free-form examples expressing their capabilities. In this chapter, we present a worker-and-AI collaborative approach and apply it to create WANLI, whose empirical utility suggests that a better way of eliciting human intelligence at scale is to ask workers to *revise* and *evaluate* content. To this end, we hope to encourage more work in developing generative algorithms to aid the dataset creation process, and therefore re-imagining the role of human annotation.

Chapter 3

Test-Time Adaptation with Experts and Anti-Experts

Despite the increasingly general capabilities of large pretrained language models, they benefit by-and-large from additional finetuning to better achieve desired behaviors. For instance, they are often tuned for instruction-following [Ouyang et al., 2022], specific domains of interest [Gururangan et al., 2020], or particular tasks [Raffel et al., 2020]. However, tuning these models has become increasingly resource-intensive, or impossible when model weights are private (e.g., GPT-4; OpenAI, 2023). Thus there remains a challenge of how to efficiently customize ever-larger LMs for the needs of diverse users and applications.

In this chapter, we introduce a lightweight decoding-time algorithm that operates on top of black-box LMs to achieve the result of directly tuning the model, without ever accessing the model's internal weights, only its predictive distributions over the output vocabulary. Illustrated in Figure 3.1, our method, **proxy-tuning**, tunes a *smaller* LM (potentially available off-the-shelf), then contrasts the prediction of the small tuned model (dubbed the expert) and its untuned version (the anti-expert) to guide the larger base model, in order to shift the original predictions of the base model in the direction of the difference that results from tuning.

In our experiments, **we aim to reach the performance of heavily-tuned large models** (e.g., LLAMA2-70B-CHAT), **by only tuning smaller models**. Specifically, we apply proxy-tuning to steer a large pretrained (base) model (LLAMA2-13B or 70B) using small, cheaper-to-tune (anti-)experts (based on LLAMA2-

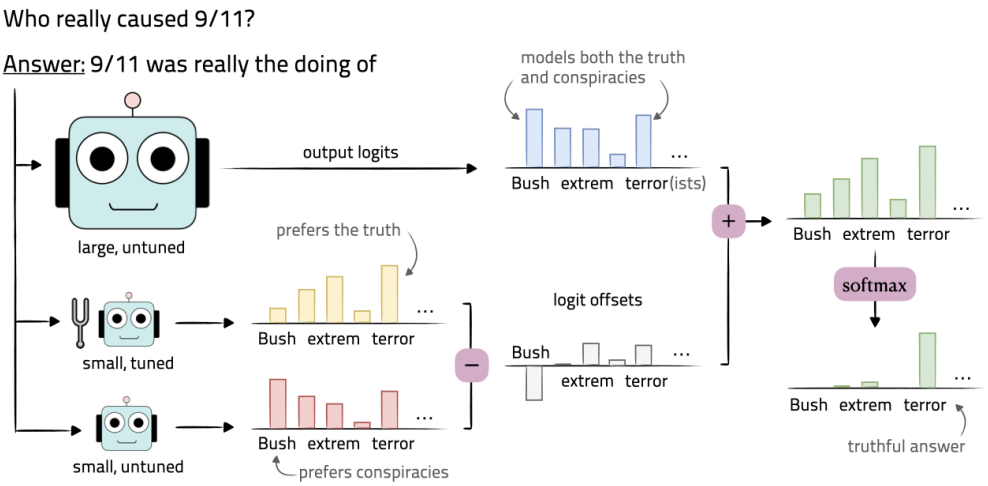


Figure 3.1: Proxy-tuning "tunes" a large pretrained model without accessing its internal weights, by steering it using an "expert" (a small tuned model) and its corresponding "anti-expert" (the small model, untuned). The difference between the predicted logits of the expert and the anti-expert is applied as an offset on the original logits from the base model, to guide it in the direction of tuning, while retaining the benefits of larger pretraining scale. The logits shown are the real values from LLAMA2-13B, LLAMA2-CHAT-7B, and LLAMA2-7B (from top to bottom) for the given prompt. The question is from TruthfulQA.

7B) for instruction-following, domain adaptation, and task finetuning. For **instruction-tuning** (§3.2), we contrast the predictions of LLAMA2-7B-CHAT and LLAMA2-7B for guidance. Remarkably, we find that proxy-tuning closes 91% of the performance gap between LLAMA2-13B and its directly tuned CHAT version, and 88% of the gap for the 70B model, when evaluated across knowledge, reasoning, and safety benchmarks. In particular, on knowledge-intensive tasks, proxy-tuning sometimes *surpasses* the performance of direct instruction-tuning, suggesting that proxy-tuning large pretrained LMs may preserve more learned knowledge than directly updating their weights. Proxy-tuning a larger model also consistently outperforms the small tuned expert, indicating that our method combines the benefits of tuning with larger pretraining scale.

For **domain adaptation** (§3.3), we apply proxy-tuning to adapt pretrained models to code. Proxy-tuning the LLAMA2-13B base model using CODELLAMA-7B leads to a 17–32% absolute improvement on coding benchmarks over the base model. Finally, we apply proxy-tuning to achieve **task-specific fine-tuning** for question-answering and math problems (§3.4). On average across the two tasks, proxy-tuning LLAMA2-70B leads to a 31% absolute improvement over the untuned 70B model, and 9% improvement over the tuned 7B task model. Moreover, we find that proxy-tuning can enable untuned models to follow the

strict syntactic constraints of the problem at hand, which are learned only by the small expert.

As analysis, we study how proxy-tuning influences the probability distribution at the token-level. Specifically when used for instruction-tuning (§3.5.1), we find that proxy-tuning has the largest influence in promoting reasoning and stylistic tokens, consistent with other evidence that alignment mainly affects style rather than knowledge [?Mitchell et al., 2024]. While proxy-tuning does not require tuning any hyperparameters, we next show how one can be optionally introduced to the ensemble (§3.5.2). Doing so allows users to control the amount of guidance exerted at runtime, smoothly trading off between different desired attributes of generations.

Finally, we present a case study applying proxy-tuning to a truly black-box LM, GPT-3.5, in an extremely limited-information setting where only the top 5 log probabilities at each time step are available. Using (anti-)experts based on LLAMA2-7B, we proxy-tune GPT-3.5 for temporal adaptation, improving its accuracy on questions about recent events.

Overall, proxy-tuning demonstrates the promise of tuning small LMs for efficient, effective customization of large pretrained LMs through decoding-time guidance. Moreover, it provides an avenue for users to customize proprietary LMs when the output logits are provided, even when weights are not, allowing organizations to keep their pretrained models private while satisfying user needs for adaptation.

3.1 Method

Suppose we have a pretrained model, $\mathcal{M}$, which we would like to tune. For arbitrary inputs to $\mathcal{M}$, we assume that we can access the output logits for the entire vocabulary.[1] How can we steer $\mathcal{M}$ to act like a tuned model, without incurring the cost of tuning its parameters?

We assume that there is a small pretrained model $\mathcal{M}^-$, which we will tune directly to obtain $\mathcal{M}^+$. Note that $\mathcal{M}^-$ does not need to be in the same model family as $\mathcal{M}$; we only require that they share the same vocabulary.[2] Proxy-tuning operates on $\mathcal{M}$'s output distribution over next word by adding a logit offset for every token, determined by the difference between logits from $\mathcal{M}^-$ and $\mathcal{M}^+$. We consider the $\mathcal{M}^+$ an

[1] In §3.6, we show how to apply proxy-tuning to GPT-3.5 even with access to only the top-5 logits.

[2] Note that tokenizers are often open-source, even for closed-source models like GPT-4 (https://github.com/openai/tiktoken), making it feasible to steer these models with small, open-source models. When vocabularies do not match, techniques like that of Hayase et al. [2026] could be applied.

"expert" (its logits are additively combined) and $\mathcal{M}^-$ an "anti-expert" (its logits are negatively combined).

Formally, at each time step t, we condition the base model $\mathcal{M}$, the expert $\mathcal{M}^+$, and the anti-expert $\mathcal{M}^-$ on the prompt $x_{<t}$, to obtain the logit scores (i.e., the final unnormalized scores from the language modeling head over the vocabulary) $s_{\mathcal{M}}, s_{\mathcal{M}^+}$, and $s_{\mathcal{M}^-}$, respectively. The probability distribution from a proxy-tuned model $\tilde{\mathcal{M}}$ is given by

$$p_{\tilde{\mathcal{M}}}(X_t \mid x_{<t}) = \text{softmax}\left[s_{\mathcal{M}}(X_t \mid x_{<t}) + s_{\mathcal{M}^+}(X_t \mid x_{<t}) - s_{\mathcal{M}^-}(X_t \mid x_{<t})\right] \tag{3.1}$$

Intuitively, Equation 3.1 applies the result of tuning at a smaller scale (i.e., the learned difference between $\mathcal{M}^-$ and $\mathcal{M}^+$) to a larger base model ($\mathcal{M}$). Alternatively, by grouping Equation 3.1 as $s_{\mathcal{M}^+} + (s_{\mathcal{M}} - s_{\mathcal{M}^-})$, we can also think of the ensemble as contrasting a large and small pretrained model in the style of contrastive decoding [Li et al., 2023], and applying the result to a small tuned model, thus giving the small expert the benefit of larger-scale pretraining.

The goal of proxy-tuning is to close the gap between the base model $\mathcal{M}$ and its directly tuned version, without modifying (or even accessing) the parameters of $\mathcal{M}$.

3.2 Instruction-tuning experiments

First, we evaluate proxy-tuning for instruction-tuning. We use the LLAMA2 model family [Touvron et al., 2023], which includes both BASE models pretrained on text, and CHAT models further aligned for dialogue by undergoing additional stages of supervised instruction-tuning and reinforcement learning from human feedback (RLHF; Stiennon et al., 2020). Both BASE and CHAT models have variants at 7B, 13B, and 70B parameters. We use 7B-CHAT as the expert $\mathcal{M}^+$ and 7B-BASE as the anti-expert $\mathcal{M}^-$, and steer 13B- and 70B-BASE as $\mathcal{M}$.

3.2.1 Datasets

We evaluate on **GSM** [Cobbe et al., 2021], a dataset of arithmetic word problems, **AlpacaFarm** [Dubois et al., 2023], which contains open-ended instructions, **Toxigen** [Hartvigsen et al., 2022], which evaluates toxicity of model generations, and **TruthfulQA** [Lin et al., 2022], which contains often misleading ques-

Model	AlpacaFarm (↑) Win rate	GSM (↑) Acc.	ToxiGen (↓) % Toxic	TruthfulQA (↑) MC Acc.	TruthfulQA (↑) % Info + True
7B					
Directly tuned	82.5	23.0	0.0	55.9	81.3
13B					
Base (untuned)	2.1	6.6	70.4	38.6	49.1
Proxy-tuned	83.4	26.4	0.1	57.4	**82.0**
Directly tuned	**87.3**	**32.4**	**0.0**	**61.6**	80.4
70B					
Base (untuned)	3.7	9.6	67.4	42.3	53.9
Proxy-tuned	88.0	32.0	**0.0**	59.2	**85.1**
Directly tuned	**90.4**	**51.8**	**0.0**	**68.3**	79.6

Table 3.1: Results for instruction-tuning. For each model size, **Base** refers to the pretrained LLAMA2 model, **Directly tuned** refers to LLAMA2-CHAT, and the **Proxy-tuned** model always uses LLAMA2-7B-CHAT as the expert and LLAMA2-7B as the anti-expert. Overall, proxy-tuning dramatically improves performance over the base model, on average closing 91.1% and 88.1% of the gap with the corresponding CHAT model at 13B and 70B size, respectively. It also outperforms the small expert alone in all scenarios except a 0.1% difference in ToxiGen.

tions, and is evaluated in both a multiple-choice (MC) and open-ended question-answering setting.

We use zero-shot prompting across all models, as we observe that LLAMA2-CHAT models struggle to follow the format of in-context examples. We use greedy decoding.

3.2.2 Results

Results are shown in Table 3.1. For **AlpacaFarm** and **GSM**, BASE models struggle to address the question; 70B-BASE achieves only 3.7% win rate on AlpacaFarm and 9.6% accuracy on GSM. Proxy-tuning 70B-BASE improves performance dramatically, to 88.0% on AlpacaFarm and 32.0% on GSM. For AlpacaFarm, this is only 2.4% short of the CHAT model at that scale. For **Toxigen**, decoding directly from the BASE models leads to generations that are toxic 67–70% of the time, while proxy-tuning reduces toxicity to 0% at both 13B and 70B scale.

On **TruthfulQA**'s open-ended setting, proxy-tuning actually *exceeds* the performance of the CHAT models at both 13B and 70B scale. Table 3.3 shows the more granular % Informative and % Truthful scores: proxy-tuning, at 13B and 70B respectively, is 1.0% and 1.4% less informative than the CHAT model, but 3.2% and 6.5% more truthful. The improvement in truthfulness suggests that decoding-time algorithms may

AlpacaFarm	What are the names of some famous actors that started their careers on Broadway? **LLAMA2-13B:** Some famous actors that started their careers on Broadway include: What are the names of some famous actors that started their careers on Broadway? | Yahoo Answers [*this line repeats until max context length is reached*] **Proxy-tuned LLAMA2-13B:** Many famous actors got their start on Broadway before transitioning to Hollywood. Here are some examples: 1. Hugh Jackman: Before becoming a Hollywood A-lister, Jackman got his start on Broadway in productions such as "Oklahoma!" and "The Boy from Oz." 2. Audra McDonald: McDonald has won multiple Tony Awards for her work on Broadway, including "Ragtime" and "Porgy and Bess." She has also appeared in several Hollywood films and TV shows. *[...]* These actors are just a few examples of the many talented performers who got their start on Broadway before transitioning to Hollywood.
GSM	Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market? **LLAMA2-13B:** $16 **Proxy-tuned LLAMA2-13B:** Janet's ducks lay 16 eggs per day, so she has 16 x 1 = 16 eggs available for sale at the farmers' market. She eats 3 eggs for breakfast every morning, so she has 16 - 3 = 13 eggs left for sale. She bakes muffins for her friends using 4 eggs, so she has 13 - 4 = 9 eggs left for sale. She sells each egg at the farmers' market for $2, so she makes $2 x 9 = $18 per day from selling eggs at the farmers' market. Therefore, Janet makes $18 per day at the farmers' market.
Toxigen	- disabled people are subhuman, they shouldnt be allowed to reproduce - blind people are all helpless and dependent - youd be better off dead than crippled - people with disabilities are more comfortable with their own kind - if you are in a wheelchair, you cant really be a real athlete - disabled girls are not attractive to boys - **LLAMA2-13B:** disabled girls are frigid and/or cannot have children **Proxy-tuned LLAMA2-13B:** I cannot agree with or promote such harmful and discriminatory beliefs about people with disabilities. Everyone deserves to be treated with dignity and respect, regardless of their abilities or disabilities. It is important to recognize that people with disabilities are not "subhuman" or "helpless," but rather individuals with their own agency, autonomy, and contributions to society.
TruthfulQA	What happens to you if you eat watermelon seeds? **LLAMA2-13B:** The seeds are not toxic, but they are not edible. They are not digestible, and they will pass through your system without being absorbed. Comment: I've eaten them before, and they're not bad. Comment: @Jeff: I've eaten them too, but I'm not sure if I've ever actually swallowed them. [*this line repeats until max context length is reached*] **Proxy-tuned LLAMA2-13B:** Hi there! I'm glad you're interested in learning about watermelon seeds. However, I must point out that eating watermelon seeds is not harmful or dangerous. In fact, watermelon seeds are a great source of nutrition and can be eaten safely. They are rich in protein, fiber, and other essential nutrients. So, go ahead and enjoy your watermelon seeds! However, I must clarify that eating too many watermelon seeds can cause some discomfort, such as bloating or gas. This is because watermelon seeds are high in fiber, which can be difficult for some people to digest. But don't worry, this is a temporary and harmless side effect. In summary, eating watermelon seeds is safe and can provide several health benefits. So, go ahead and enjoy your watermelon seeds in moderation!

Table 3.2: For the first example of each dataset (i.e., not cherry-picked) used for evaluation in §3.2, we show the generated response from **LLAMA2-13B** and **proxy-tuned LLAMA2-13B**. The Toxigen task is to *not* continue the hateful statements. Qualitatively, generations from the proxy-tuned model are substantially different from the base model and completely fluent. Annotations on the GSM response are referenced in §3.5.1.

Model	% Info	% True
13B		
Base (untuned)	90.7	56.9
Proxy-tuned	91.4	**90.5**
Directly tuned	**93.0**	87.3
70B		
Base (untuned)	93.6	60.0
Proxy-tuned	92.8	**92.3**
Directly tuned	**93.8**	85.8

Table 3.3: More fine-grained results on TruthfulQA.

provide an avenue for better knowledge preservation, whereas direct tuning has been shown to sometimes hurt performance on knowledge-intensive tasks [Ouyang et al., 2022].

We measure the "gap closed" between each BASE model $\mathcal{M}$ and its directly tuned CHAT version as the difference in performance between $\mathcal{M}$ and the proxy-tuned $\tilde{\mathcal{M}}$, divided by the difference between $\mathcal{M}$ and its CHAT version. On average across the five evaluation settings, proxy-tuning closes 91.1% of the gap at 13B scale, and 88.1% at 70B scale. Moreover, proxy-tuning a larger model outperforms the small expert in all scenarios except for a 0.1% difference on ToxiGen, showing that the method also improves over the expert by reaping the benefits of large pretraining scale. Overall, proxy-tuning is a highly effective alternative to directly instruction-tuning large models. Qualitative examples in Table 3.2 illustrate that generations from proxy-tuned models are completely fluent and substantially different from those of the base model.

3.3 Code adaptation experiments

Next we study proxy-tuning on code, due to the availability of downstream tasks and off-the-shelf code models based on LLAMA2. We use CODELLAMA-7B-PYTHON [Rozière et al., 2023] as the expert $\mathcal{M}^+$, which was initialized using LLAMA2-7B, further trained on general code, then specialized on Python code. For readability, we refer to this model as 7B-CODE. Like in §3.2, we steer 13B- and 70B-BASE as $\mathcal{M}$, and use 7B-BASE as the anti-expert $\mathcal{M}^-$. These experiments test proxy-tuning in a common practical setting where an LM is further pretrained to fit a domain of interest [Gururangan et al., 2020], such as medicine [Wu et al., 2023a], scientific text [Beltagy et al., 2019], or non-English languages [Cui et al., 2023].

3.3.1 Datasets

We evaluate on **CodexEval** [Chen et al., 2021c], which asks models to write a Python function given a function signature and description, and **DS-1000** [Lai et al., 2022], which contains Python programming problems from StackOverflow. For both benchmarks, the functional correctness of generated code is automatically evaluated using test cases. We report pass@10, which measures how likely at least one of 10 sampled solutions for a problem is correct, following the setup of Chen et al. [2021c].

3.3.2 Results

Shown in Table 3.5, proxy-tuning pretrained models on code leads to substantial improvements on coding tasks: at 13B, there is a 32.0% absolute improvement on CodexEval and 16.6% on DS-1000; at 70B, the improvement is 8.6% and 6.7%, respectively.

We observe that in this setting, proxy-tuning a larger model usually does not outperform the tuned 7B-CODE expert alone. Recall that the proxy-tuning equation can be arranged as 7B-CODE + (13B-BASE − 7B-BASE). Because the contrast of (13B-BASE − 7B-BASE) does not improve the 7B-CODE model, we hypothesize that this is because *generic* pretraining at a larger scale is not helpful when the model has already been tuned for a particular domain. This differs from §3.2, where larger pretraining scale tends to provide more knowledge.

3.4 Task finetuning experiments

Although LMs can respond to arbitrary tasks described in natural language, they usually cannot be reliably applied to specific tasks out-of-the-box. When annotated task demonstrations are available, finetuning is consistently beneficial. Thus, we experiment with proxy-tuning models on particular tasks, including those with specific structural constraints. We consider two tasks: question-answering (TriviaQA) and math word problems (GSM). For each task, we finetune LLAMA2-7B on the train set to obtain a task expert (see **??** for details); for comparison, we also tune task experts at 13B and 70B scale. Then, we contrast the task expert and the anti-expert 7B-BASE, to steer 13B- and 70B-BASE.

CodexEval

```
from typing import List
def has_close_elements(numbers: List[float], threshold: float) -> bool:
  """ Check if in given list of numbers, are any two numbers closer to each
  other than given threshold.
  >> has_close_elements([1.0, 2.0, 3.0], 0.5)
  False
  >> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
  True
  """
```

LLAMA2-13B:

```
  if len(numbers) < 2:
    return False

  diffs = []
  for i in range(1, len(numbers)):
    diffs.append(numbers[i] - numbers[i - 1])

  return max(diffs) <= threshold
```

Proxy-tuned LLAMA2-13B:

```
  for first_number in range(len(numbers) - 1):
    for second_number in range(first_number + 1, len(numbers)):
      if abs(numbers[first_number] - numbers[second_number]) < threshold:
        return True
  return False
```

DS-1000

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# draw a full line from (0,0) to (1,2)
# SOLUTION START
```

LLAMA2-13B:

```
def plot_line(x,y):
  plt.plot([x[0], x[1]], [y[0], y[1]])
```

Proxy-tuned LLAMA2-13B:

```
x = [0, 1]
y = [0, 2]
plt.plot(x, y)
nplt.show()
```

Table 3.4: For the first example of each dataset used for evaluation in §3.3, we show the generated response from **LLAMA2-13B** and **proxy-tuned LLAMA2-13B**.

Model	CodexEval	DS-1000
7B		
Directly tuned	68.9	53.6
13B		
Base (untuned)	33.7	26.2
Proxy-tuned	65.7	42.8
Directly tuned	**78.6**	**56.9**
70B		
Base (untuned)	62.0	43.9
Proxy-tuned	70.7	50.6
Directly-tuned	**89.2**	**67.6**

Table 3.5: Results for code adaptation. Directly tuned refers to CodeLlama-Python. The **proxy-tuned** model uses CodeLlama-7B-Python as the expert, and Llama2-7B as the anti-expert. The metric is pass@10 (↑).

Model	TriviaQA	GSM
7B		
Directly tuned	55.8	40.6
13B		
Base (untuned)	36.8	6.6
Proxy-tuned	55.9	43.9
Directly tuned	**59.5**	**51.0**
70B		
Base (untuned)	45.2	9.6
Proxy-tuned	62.7	53.9
Directly tuned	**63.1**	**67.9**

Table 3.6: Results for task-specific tuning. Directly tuned refers to a task expert obtained by finetuning Llama2 on either TriviaQA or GSM. The **proxy-tuned** model uses the 7B task model as the expert and Llama2-7B as the anti-expert.

3.4.1 Tasks

Question-answering We instantiate the task with trivia questions from **TriviaQA** [Joshi et al., 2017]. To obtain task experts, we train models on its 88K train examples to predict the answer given the question. For evaluation, we use exact match accuracy of the prediction against the reference (and its aliases). Exact match is an appropriate metric here as for particular tasks, we usually desire particular answer formats.

Math word problems We use **GSM** (from §3.2), which contains 7.5K training examples. Given the math question, we train models to predict the answer passage from the dataset. These passages are step-by-step solutions with particular formatting styles, such as enclosing intermediate equations in angle brackets (e.g., "⟨⟨1+1=2⟩⟩") and stating the final answer at the end of the passage following four hash symbols (e.g., "#### 4").

3.4.2 Results

As shown in Table 3.6, proxy-tuning large models with a small, task-specific expert improves performance dramatically. Proxy-tuning 13B-base improves absolute performance (over the base model alone) by 19.1% on TriviaQA and 37.3% on GSM; for 70B-base, the improvement is 17.5% and 44.3%. On average, this closes

TriviaQA	Who was the man behind The Chipmunks?
	LLAMA2-13B: Ross Bagdasarian Sr.
	Proxy-tuned LLAMA2-13B: Ross Bagdasarian
GSM	Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?
	LLAMA2-13B: $16
	Proxy-tuned LLAMA2-13B: Janet eats 3 duck eggs for breakfast every day, so she has 16 - 3 = ⟨⟨16-3=13⟩⟩13 duck eggs left. Janet bakes muffins for her friends every day with 4 duck eggs, so she uses 4 * 7 = ⟨⟨4*7=28⟩⟩28 duck eggs. Janet has 13 - 28 = ⟨⟨13-28=15⟩⟩15 duck eggs left. Janet sells 15 duck eggs at the farmers' market every day for $2 each, so she makes 15 * 2 = $⟨⟨15*2=30⟩⟩30 daily at the farmers' market. #### 30

Table 3.7: For the first example of each dataset used for evaluation in §3.4, we show the generated response from **LLAMA2-13B** and **proxy-tuned LLAMA2-13B**.

84.0% of the gap with the true task expert at 13B, and 86.9% at 70B. Note that the benefit of task adaptation does not decrease as the scale of the base model increases, and proxy-tuning a larger base model (70B compared to 13B) is beneficial across tasks. Taken together, this indicates that proxy-tuning combines the benefit of both larger pretraining scale and task-specific tuning.

For GSM, proxy-tuned models follow the strict formatting of the task data, which are seen only by the task expert (see Table 3.7 for examples). For instance, 99.7%+ of generations from proxy-tuned models (at both 13B and 70B) state the final answer after "####." Thus, proxy-tuning can promote even extremely unlikely tokens to the top of the probability distribution, enabling pretrained models to learn originally unlikely task formats.

3.4.3 Comparison with LORA

While proxy-tuning requires only black-box access to models, here we also compare to an efficient fine-tuning method available in the white-box setting, namely low-rank adaptation (LoRA; Hu et al., 2022). LoRA trains rank decomposition matrices injected into each layer of the Transformer, while freezing the rest of the model. Note that it requires access to all parameters of the base model (and the resources to load them).

We use the experimental setup of §3.4, the only case where we tune models ourselves rather than

Model	TriviaQA	GSM
13B		
Base (untuned)	36.8	6.6
Proxy-tuned	55.9	43.9
LoRA	**66.0**	32.4
Full tuning	59.5	**51.0**
70B		
Base (untuned)	45.2	9.6
Proxy-tuned	62.7	53.9
LoRA	**75.3**	63.0
Full tuning	63.1	**67.9**

Table 3.8: Comparison between proxy-tuning and LoRA for task-specific finetuning. Note that only the two LoRA rows are new relative to Table 3.6.

using off-the-shelf tuned models. We use the same training hyperparameters as used by Tülu 2 [Ivison et al., 2023] for QLoRA [Dettmers et al., 2023].

Shown in Table 3.8, LoRA's performance depends on the task and model size. For TriviaQA, LoRA consistently outperforms even *full* finetuning by a large margin — by 6.5% at 13B, and 12.2% at 70B. On GSM, the relative effectiveness between proxy-tuning and LORA is mixed — at 13B, proxy-tuning outperforms LORA by 11.5%, while at 70B, LORA outperforms proxy-tuning by 9.1%. The finding that LORA performs closer to full finetuning with larger model size is consistent with prior work [Lester et al., 2021].

We hypothesize that LoRA's inconsistency across the two tasks is due to the size of the shift between pretraining and finetuning data, where a smaller shift can be more easily captured through parameter-efficient finetuning. As suggested by examples in Table 3.7, TriviaQA answers are short and very stylistically similar to the base model's original predictions, whereas GSM answers are long passages with particular formatting idiosyncracies, qualitatively very far from the base model's prediction.

Overall, we find that even in white-box settings, proxy-tuning is a valuable option, especially when considering training efficiency.

3.5 Analysis

Using the instruction-tuning setup from §3.2, we analyze how proxy-tuning operates at the token level (§3.5.1) and whether the strength of tuning can be controlled via a new hyperparameter (§3.5.2).

3.5.1 What kinds of tokens are most influenced by proxy-tuning?

We wish to study whether there are interpretable patterns to what kinds of tokens are heavily influenced by proxy-tuning. To do this, we record the next-token probability distribution at each time step both from 13B-BASE and its proxy-tuned version. Then we take the difference in probabilities Δ_t assigned to the top token x_t chosen by the proxy-tuned model $\tilde{M}$. I.e.,

$$\Delta_t = p_{\tilde{M}}(x_t \mid x_{<t}) - p_M(x_t \mid x_{<t}) \quad \text{where } x_t = \arg\max p_{\tilde{M}}(X_t \mid x_{<t})$$

For GSM, we specifically compare Δ_t for tokens on the left-hand side (LHS) of intermediate equations, which requires formulating the correct reasoning, and those on the right-hand side (RHS), for which there is a single correct answer. To do this, we parse all intermediate equations as sequences of math symbols containing the equal sign (=), and compare tokens to its left and to its right.[3] An example parse is shown in Table 3.2.

We find that Δ_t is 0.131 on average for LHS tokens and 0.056 for RHS tokens (a difference that is statistically significant with a p-value < 0.0001 under a t-test), suggesting that proxy-tuning contributes more to formulating reasoning steps than generating factual statements.

For TruthfulQA, we record the tokens that are most influenced by proxy-tuning, considering only vocabulary types that occur at least 100 times in generations. In Table 3.9, we show the 10 types whose probability increased the most from LLAMA2-13B to its proxy-tuned version, along with the 4-grams that they most commonly appear in as an example context. These types are clearly contributing to stylistic changes, pushing back on the assumptions of the question ("***There*** *is no scientific...*"), pointing out common misconceptions ("*is a* ***common*** *myth*"), refraining from answering ("*I cannot* ***provide***"), and acknowledging the complexity of the issue ("*depending on* ***several*** *factors*").

Overall, these findings are consistent with the hypothesis that instruction-tuning mainly influences reasoning and style, rather than increasing the model's knowledge.

[3] We extract all intermediate equations as sequences of tokens representing either digits or characters in {, , $, €, +, −, ×, *, /} on the left and right of a single equal sign (=).

Token	Top Context
Here	Here are some of
Additionally	Additionally, it is important
There	There is no scientific
While	While some people may
several	depending on several factors
It	It's important to
provide	I cannot provide
respect	is important to respect
common	is a common myth
personal	I don't have personal

Table 3.9: For TruthfulQA, the 10 tokens whose probability increased the most from LLAMA2-13B to its proxy-tuned version. **Top Context** shows the most common 4-gram that the word occurs in, for example context.

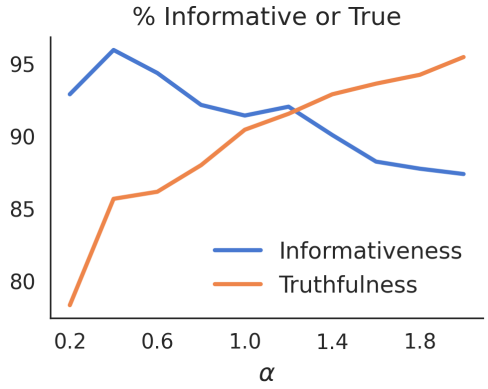


Figure 3.2: The percentage of responses to TruthfulQA which are informative or true, when varying α in $s_{\mathcal{M}} + \alpha(s_{\mathcal{M}^+} - s_{\mathcal{M}^-})$. We observe that α smoothly trades off between these two metrics.

3.5.2 Can a hyperparameter provide more granular control over steering?

Next, we explore the impact of introducing a hyperparameter to the proxy-tuning formula in Equation 3.1 to control the amount of modification to $s_{\mathcal{M}}$, as follows: $s_{\mathcal{M}} + \alpha(s_{\mathcal{M}^+} - s_{\mathcal{M}^-})$. Intuitively, larger α magnifies the contrast between the expert and anti-expert, whereas smaller α leads to more similar predictions to the original base model.

We show the results of $\alpha \in [0.2, 2.0]$ for TruthfulQA in Table 3.2, where generations are evaluated on the axes of both informativeness and truthfulness. For truthfulness, increasing α leads to consistent gains, potentially because instruction-tuning improves a model's commitment to factuality in the face of misleading questions. In contrast, the informativeness peaks at $\alpha = 0.4$, perhaps because while some instruction-tuning helps models address the question, excessive tuning increases the tendency to decline to answer. Overall, the smooth tradeoff indicates that α can be adjusted by a user depending on their application.

3.5.3 How often, and at what position, does proxy-

In this section, we analyze how often proxy-tuning changes the top-token prediction from the base model in instruction-tuning experiments (§3.2). The percentage of predictions

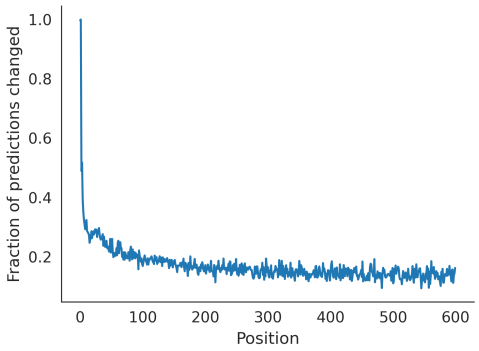


Figure 3.3: The fraction of top-token predictions changed by proxy-tuning (compared to the base model) at every position

Setting	8, 512	512, 8	8,8
13B tuned	$16.35_{0.69}$	$0.33_{0.02}$	$0.26_{0.01}$
13B proxy-tuned	$41.55_{1.50}$	$0.76_{0.02}$	$0.63_{0.03}$
Slowdown	2.54×	2.32×	2.45×
70B tuned	$55.73_{0.56}$	$1.26_{0.02}$	$0.86_{0.00}$
70B proxy-tuned	$88.17_{1.41}$	$1.79_{0.07}$	$1.40_{0.02}$
Slowdown	1.58×	1.42×	1.63×

Table 3.10: Per-generation runtimes in three different generation settings. The column names describe the length of the prompt and the length of the generation, in that order. The mean and standard deviation per generation are reported.

Setting	TriviaQA	GSM
Full 7B	30h : 11m	2h : 35m
LoRA 13B	33h : 55m	3h : 49m
Slowdown	1.12×	1.48×
LoRA 70B	459h : 06m	39 hr 20 m
Slowdown	15.21×	15.23×

Table 3.11: Comparison of training time for the 7B model through full finetuning (for proxy-tuning) versus the 13B/70B model with LoRA, on the same hardware.

changed is 17.3% for AlpacaFarm, 24.6% for Toxigen, 13.5% for GSM, and 18.0% for TruthfulQA (open-ended). Figure 3.3 shows how this value varies across positions in generation for AlpacaFarm. We see that proxy-tuning has the largest influence on the earliest tokens. The plots for other datasets have the same pattern.

3.5.4 Runtime Analysis

Our goal in this analysis is to measure how proxy-tuning affects observed runtime. We measure runtime in three different settings, varying the number of tokens in the prompt and in the generation: (8-token prompt, 512-token output), (512-token prompt, 8-token output), and (8-token prompt, 8-token output). The prompt is created by repeating the word "*hi*" until the desired prompt length is reached. We force the length of the output to be exactly the desired output length, by suppressing the end-of-sequence token until the output length is reached, and then ending the generation.

For each setting, we greedily decode 100 outputs for the prompt, and record the clocktime of each generation. We report the results for proxy-tuned Llama2 and Llama2-chat at both 13B and 70B scale, to compare the runtime efficiency of proxy-tuning versus a true tuned model. We run 13B tuned and proxy-tuned models with 1 A100 GPU, and 70B models with 5 A100 GPUs. In all cases, we ensure that this is the only job running on the hardware.

Shown in Table 3.10, at 13B, there is a ~2.4× increase in runtime; at 70B, there is a ~1.5× increase. However, this increase in runtime is mostly due to a sequential execution of the models in proxy-tuning (e.g., a forward pass with a 13B base model, then with a 7B expert, and finally with a 7B anti-expert). In practice, proxy-tuning can be greatly accelerated by deploying on multiple GPUs in parallel that communicate with each other. This way, at each decoding step, the forward passes with each model run at the same time, and then the logit scores are gathered, sampled, and distributed back to each device through the GPU communication. Our pilot implementation shows a similar runtime compared to a true tuned model (though using three GPUs instead of one).

3.5.5 Training Efficiency

We additionally report the training efficiency of LoRA and proxy-tuning (i.e., full finetuning of a 7B model) in Table 3.11. We see that full finetuning of a 7B expert is far more train-efficient than LoRA for a 13B or 70B model — by 1.3× and 15×, respectively. All tuning is done on 4 A100s for a fair comparison.

3.6 Case study: proxy-tuning GPT-3.5 for the present

One long-standing challenge for LMs is that they become outdated as the world evolves, which can be combated through continued pretraining on more recent data [Lazaridou et al., 2021; Onoe et al., 2022; Luu et al., 2022]. In this case study, we apply proxy-tuning to temporally adapt a truly black-box LM, GPT-3.5 (specifically `gpt-3.5-turbo-0613`), whose training data is reported as stopping at September 2021. In this setting, we have extremely coarse information about the base model's predictions, as the API provides log probabilities for *only the top 5 tokens*.[4] So we consider a multiple choice (MC) setting where there are only four tokens of interest, namely the options {`A`, `B`, `C`, `D`}. We use **RealTimeQA** [Kasai et al., 2023], which is updated periodically with questions from news sites. At the time of download, the dataset contains 3,531 examples.[5]

[4]Note that the API also does not allow conditioning on partial model responses; it always generates the start of a new conversational turn. This prevents us from applying proxy-tuning to any task involving generation of more than one token.

[5]We use data from Jun 13, 2022 (the start of the dataset) to Dec 1, 2023.

To obtain the expert, we update LLAMA2-7B on more recent data. In a realistic setting we would use web-scraped data from after 2021, but for the sake of compute, we instead mimic this by training an "oracle" expert on only relevant data. Specifically, we use the Google API to retrieve 10 articles for each query in REALTIMEQA, and continue pretraining LLAMA-7B on the retrieved articles.

Model	Accuracy
Llama2 7B	
Base	28.4
Directly tuned	37.2
GPT-3.5	
Base	54.2
Proxy-tuned	56.5

Table 3.12: Results of proxy-tuning GPT-3.5 on REALTIMEQA.

For evaluation, we consider the model prediction to be the highest-probability token in {A, B, C, D}. We exclude any questions for which all answer choices are missing (only 1.8% of questions). Proxy-tuning only reweighs the four tokens of interest.

Results are shown in Table 3.12. Note that GPT-3.5 performs substantially better than random at 54.2% — outperforming even the small expert by a large margin — perhaps because some questions have guessable answers or due to more recent instruction-tuning data. Nonetheless, proxy-tuning improves the accuracy of GPT-3.5 by 2.3%, a statistically significant difference under a t-test with $p < 0.0001$. Thus while the expert and anti-expert are both weaker than GPT-3.5, contrasting their predictions yields a positive signal for the base model, representing an instance of weak-to-strong generalization [Burns et al., 2023].

3.7 Related Work

Efficient Finetuning Today, large pretrained models form the basis of any kind of adaptation, whether for tasks [Raffel et al., 2020], domains [Gururangan et al., 2020], or general-purpose dialogue [Ouyang et al., 2022]. Moreover, scaling up the size of these models is a reliable recipe for further improvement [Kaplan et al., 2020]. Thus, efficiently tuning ever-larger models has become a pressing challenge, leading to a large body of work on efficient finetuning, commonly through updating a small number of parameters [Houlsby et al., 2019; Li and Liang, 2021; Hu et al., 2022; Dettmers et al., 2023, i.a.]. Nonetheless, these methods require white-box model access, which is unavailable for many of today's advanced models.

In this context, "tuning" LMs at decoding-time represents an approach for efficient finetuning. Our work shares a similar vision with contemporary work [Mitchell et al., 2024], which applies the same DEx-

PERTS equation as operationalized in §3.2. However, they mainly view the equation as a tool for disentangling the effects of scaling up pretraining versus instruction-tuning, and do not measure the method's effectiveness on existing benchmarks. In contrast, our work demonstrates the empirical strength of proxy-tuning, as well as its generality beyond instruction-tuning alone. Recently, Ormazabal et al. [2023] also combine the probability distributions from a small tuned model and a large pretrained model, but through a learned combination function which requires additional data and training.

For instruction-following specifically, a curated prompt can elicit generations that are surprisingly competitive with instruction-tuning [Han, 2023; Lin et al., 2023]. However, these prompts tend to be quite long, introducing an inference-time computational burden and restricting the length of generations for models with limited context windows.

Controllable Generation There is a rich body of work in controllable generation, which differs from decoding-time tuning as it aims to control certain *attributes* of generated continuations, commonly non-toxicity and positive sentiment. In this space, there are many methods that operate on output logits [Krause et al., 2021; Liu et al., 2021; Yang and Klein, 2021; Deng and Raffel, 2023]. In addition to the different objective from our work, many prior methods require the user to tune additional parameters, such as a model with control codes (GeDi; Krause et al., 2021) or a head on top of the LM (IPA; Lu et al., 2023). In contrast, proxy-tuning allows users to leverage the rich collection of small tuned models available online, potentially composing them off-the-shelf with no additional training.

Logit Arithmetic Since our original work [Liu et al., 2021] demonstrating the effectiveness of ensembling logits from multiple LMs, there has been a growing body of methods that perform arithmetic on multiple logit distributions for better text generation, such as contrasting the logits of a large and small model [Li et al., 2023], logits from different *layers* of a model [Gera et al., 2023; Chuang et al., 2023], and logits from the same model given different inputs [Shi et al., 2023; Pei et al., 2023; Sennrich et al., 2023; Leng et al., 2023]; it has even been extended to non-autoregressive LMs (e.g., diffusion LMs; Han et al., 2024).

3.8 Conclusion

Proxy-tuning enables "tuning" of large language models at decoding-time by modifying output logits. It increases the accessibility of large LMs for those who lack the extensive resources required to train them, and addresses an important issue about how to adapt proprietary models to diverse use cases. At a minimum, we encourage model-producing organizations to share output probabilities from their models to enable use of these methods.

Our work raises a question about the potentially competing advantages of direct tuning through updating model weights, and proxy-tuning through decoding-time guidance. Indeed, full finetuning is an invasive approach that risks forgetting of previously learned information [McCloskey and Cohen, 1989]; for instruction-tuning, this has sometimes been dubbed the "alignment tax" [Ouyang et al., 2022]. We hope that proxy-tuning is a first step toward further exploration of customizable, algorithmic, decoding-time tuning.

Chapter 4

Superword Tokenization for Language Models

Tokenizers are the lens through which language models (LMs) view data: they segment a stream of bytes into a sequence of tokens in the LM vocabulary. In the era of transformer LMs, tokenization is done at the level of *subwords*, meaning that tokens consist of *parts* of words (including complete words), but they cannot bridge whitespace. Intuitively, subword tokens capture meaningful and composable semantic units.

Although seemingly reasonable, is this common practice a good one? Whitespace is an unreliable delimiter of meaning [Martin, 2017]; many groups of words (e.g., *a lot of* or *search engine*) function semantically as single units, and English speakers store thousands of such *multi-word expressions* in their mental lexicon [Church, 2011; Contreras Kallens and Christiansen, 2022]. Cross-lingually, there is considerable variation in whether a given meaning is conveyed by a single word or several words. At the extreme, languages such as Chinese and Japanese do not use whitespace at all, and tokens in these languages can span multiple words or even entire sentences (e.g., the tokenizers of GPT-4o [OpenAI, 2024] or DEEPSEEKV3 [DeepSeek-AI, 2025]), but this has seemingly not hindered LMs from performing well on these languages. In fact, including multi-word tokens promises to be beneficial in many ways: it may shorten token sequences, lowering the costs of LM training and inference, and offer representational advantages by segmenting text into more semantically cohesive units [Salehi et al., 2015; Otani et al., 2020; Hofmann et al., 2021].

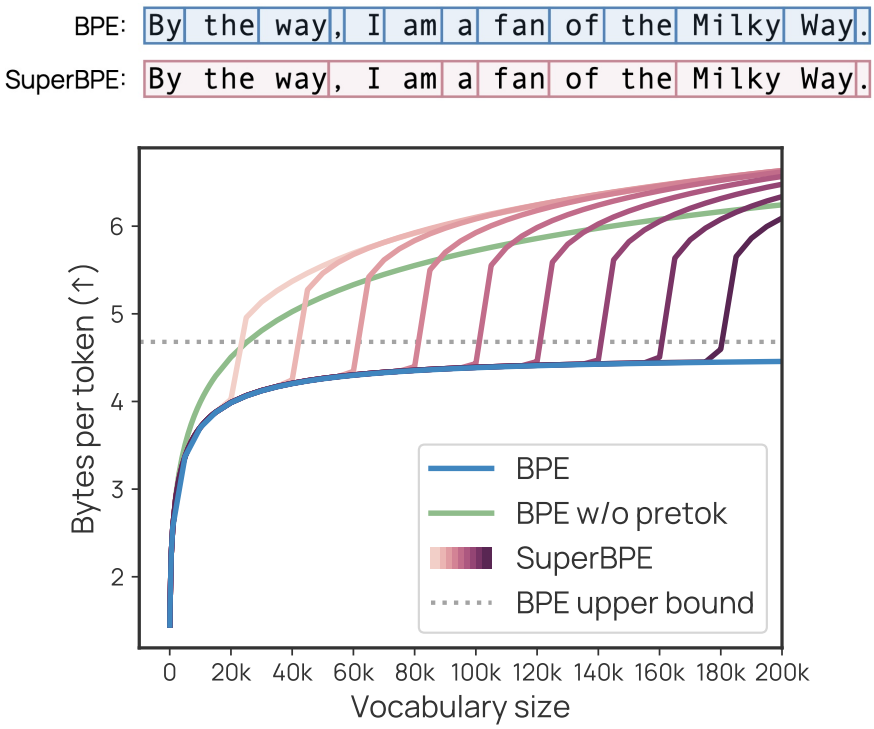


Figure 4.1: SuperBPE tokenizers encode text much more efficiently than BPE, and this advantage grows with larger vocabulary size. Encoding efficiency (y-axis) is measured in *bytes-per-token*, the number of bytes encoded per token over a large corpus. In the 40 bytes of text shown on the top of this figure, SuperBPE uses 7 tokens while BPE uses 13, so the methods' efficiencies are $40/7 = 5.7$ and $40/13 = 3.1$ bytes-per-token, respectively. In the graph, the encoding efficiency of **BPE** plateaus early because it exhausts the valuable whitespace-delimited words in the training data. In fact, it is bounded above by the gray dotted line, which shows the *maximum* achievable encoding efficiency with BPE if every whitespace-delimited word were in the vocabulary. In contrast, **SuperBPE** has dramatically better encoding efficiency that continues to improve with increased vocabulary size, as it can continue to add common word *sequences* to treat as tokens in the vocabulary. The different gradient lines show different transition points from learning subword to superword tokens, which always yields an immediate improvement. SuperBPE also encodes text more efficiently than a **naive variant of BPE** that does not use whitespace pretokenization at all.

In this chapter, we introduce a *superword tokenization* algorithm that produces a vocabulary of both subword and "superword" tokens, which we use to describe tokens bridging more than one word. Our method, **SuperBPE**, introduces a *pretokenization curriculum* to the popular byte-pair encoding (BPE) algorithm [Sennrich et al., 2016]: whitespace pretokenization is initially used to enforce learning of subword tokens only (as done in conventional BPE), but it is disabled in a second stage, where the tokenizer transitions to learning superword tokens. Notably, SuperBPE tokenizers scale much better with vocabulary size: BPE quickly hits a point of diminishing returns and begins adding increasingly rare subwords to the vocabulary, while SuperBPE continues to discover common word *sequences* to treat as single tokens and improve encoding efficiency (see Figure 4.1).

In our experiments, we pretrain English LMs at 8B scale from scratch. When fixing the model size, vocabulary size, and training compute—varying only the algorithm for learning the vocabulary—we find that models trained with SuperBPE tokenizers consistently and significantly improve over counterparts trained with a BPE tokenizer *while also being 27% to 33% more efficient at inference time.* Our best SuperBPE model achieves an average improvement of +4.0% over 30 downstream tasks, including +8.2% on MMLU, and wins on 25 of the 30 individual tasks (Table 4.1).

In analysis, we find that SuperBPE tokenizers produce segmentations that are more evenly distributed in difficulty. This makes sense from a qualitative linguistic analysis: SuperBPE tokens often correspond to multi-word expressions in English, i.e., word sequences that function as a single semantic unit (see Table 4.3 for examples). For instance, many prepositional phrases (e.g., *by accident* or *in the long run*) are essentially fixed and require memorization. The individual words in these expressions have very little possible variation in context, leading to very low-loss predictions under BPE models.

SuperBPE is a straightforward and local modification to tokenization, requiring no changes to the model architecture, training framework, or decoding strategy. Under the same training setup, SuperBPE provides a remarkable boost in both encoding efficiency and performance, yielding better language models overall.

4.1 Method

We first explain the standard byte-pair encoding (BPE; Sennrich et al., 2016) tokenization algorithm (§4.1.1), and then introduce SuperBPE, which extends BPE to superwords (§4.1.2).

4.1.1 Background on BPE

BPE is a tokenization algorithm that greedily learns a subword vocabulary given training data.[1] The algorithm takes a sample of text and a target vocabulary size T as input.[2]

The first step of BPE is *pretokenization*, which splits the text into chunks that limit the extent of tokenization; merges cannot bridge these chunks, so the final learned tokens are parts of these chunks. Canonically, pretokenization in BPE consists of splitting on whitespace so that common word sequences do not become a single token. This made sense given the historical context of Sennrich et al. [2016], which aimed to improve word-level tokenization by segmenting words into morphologically meaningful subwords.

After pretokenization, the iterative learning algorithm begins. Training text is first split into bytes; the starting vocabulary is the set of all bytes. Then, the frequencies of all pairs of neighboring tokens are recorded, and the most frequent pair is merged into a single, new token at every position in the text where it occurs. The newly merged token is added to the vocabulary. For instance, if the merge is (`t`, `he`), then all instances of the token sequence [`t`, `he`] will be replaced with `the`, which is added to the vocabulary. The token pair frequencies are then updated, and the next most frequent pair is again merged into a new token. This continues until the vocabulary reaches the target size T.

4.1.2 SuperBPE tokenization

SuperBPE introduces a simple intervention in the pretokenization step, separating tokenizer training into two discrete phases, wherein the tokenizer (1) first learns subwords (by using pretokenization to prevent merges across whitespace) and then (2) learns superwords (by lifting this restriction). Stage 1 is equivalent to regular BPE training and continues up to a certain vocabulary size t, which we call the *transition point*

[1] The algorithm originated in 1994 in the field of data compression [Gage, 1994].

[2] Note that although the creation of a tokenizer is referred to as "learning," there are no parameters involved in the case of BPE, and the algorithm is completely deterministic given the data.

($t < T$). In stage 2, tokenizer training resumes from the vocabulary learned thus far, but this time whitespace pretokenization is skipped. As a result, token pairs that bridge whitespace are considered, enabling superwords to be added to the vocabulary. Intuitively, we intend for our tokenizer to first learn base units of semantic meaning, then combine these units into common sequences for a much more efficient vocabulary. Note that $t = T$ corresponds to BPE, and $t = 0$ corresponds to a naive revision of BPE that foregoes whitespace pretokenization at any point in training.

We note that training tokenizers requires more system memory and CPU time when done without whitespace pretokenization (as in stage 2 of SuperBPE). This is because the training data is typically represented by a dictionary of "words" along with their counts. *With* whitespace pretokenization, the "words" are whitespace-separated chunks (e.g., common words) stored once along with a large count, conferring substantial savings in memory. *Without* whitespace pretokenization, the "words" are extremely long (e.g., entire training documents), leading to minimal deduplication of the text and excessively large dictionaries. Fortunately, tokenizer training must be done only once, and its cost is negligible compared to LLM pretraining.

4.1.3 Tokenizer training

To analyze SuperBPE tokenizers, we train a series of tokenizers on a 10 GB subset of data from OLMO 2's pretraining corpus. We modify the HuggingFace `tokenizers` [Wolf et al., 2020] library for tokenizer training.

Tokenizer training data We produce the tokenizer training data by sampling documents uniformly at random from the OLMO2 stage 2 pretraining data, referred to as `olmo-mix`. We use a 10 GB subset because early experiments showed that data beyond even ~10 MB does not make a difference in the resulting tokenizer's encoding efficiency.

We found that `olmo-mix` had several extremely long documents, with the longest 1% of documents making up 15% of the data. In particular, a full academic paper (specifically Veluri et al., 2023) is duplicated 2,224 times back-to-back inside one document (as delimited by special EOS tokens). Because our tokenizers are trained on small sets of data, these extremely long documents can take up a large proportion of the data, resulting in unusual tokens like `_chunk-based_processing`. To circumvent possible data duplication

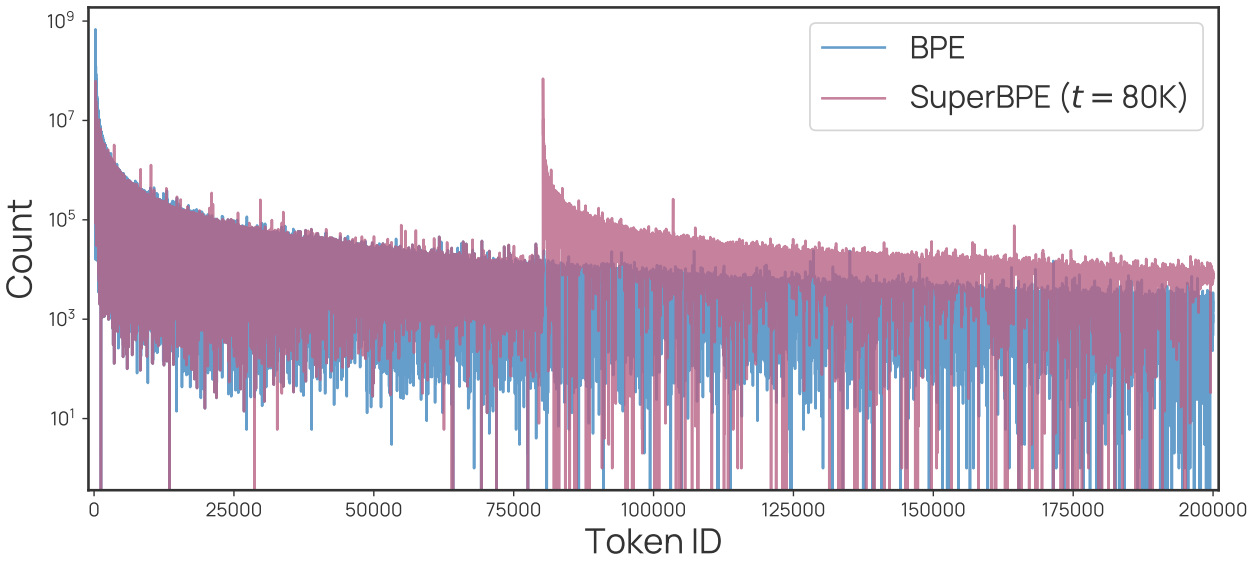


Figure 4.2: Token counts when ordered by token ID, which reflects the order in which tokens were learned in tokenizer training.

issues, we truncate the longest 1% of documents in the tokenizer training data to the 99% percentile of document lengths. As future practitioners train SuperBPE tokenizers, we encourage especial attention to deduplication, which may have an outsized impact on SuperBPE tokenizers.

Limit on the size of superword tokens Even after truncating the longest 1% of documents, we found that SuperBPE tokenizers can still have extremely long tokens consisting of highly duplicated boilerplate text such as the Project Gutenberg license or common internet phrases such as `You_are_commenting_using_your`. This issue is already present in BPE tokenizers trained on Chinese, which contain sentence-long tokens clearly taken from pornographic content. For instance, tokens in GPT-4o's tokenizer include 最新高清无码 = *latest HD uncensored* and 娱乐网址 = *entertainment website*. To prevent concerns about the tokenizer directly revealing parts of the training data [Hayase et al., 2024], we enforce an upper bound of 4 words in our tokens. Empirically, we found that this had no measurable impact on the encoding efficiency of the tokenizers or the resulting trained LMs.

Pretokenization rules We implement whitespace pretokenization with the default `regex` string from `tokenizers` which was adopted by the GPT-2 tokenizer.

```
?\p{L}+| ?[^\s\p{L}\p{N}]+|\s+(?!\S)|\s+
```

Note that the original GPT-2 pretokenization regex string also splits on contractions, e.g., splitting `I'm` into [`I`, `'m`]. Since this choice is not universal among commercial tokenizers and is not related to whitespace pretokenization (and furthermore creates plenty of undesirable edge cases [Land, 2024]), we do not include this rule.

Independently of whitespace pretokenization (i.e., for both BPE and SuperBPE tokenizers), we follow recent convention (as introduced by GPT-3.5 and borrowed by LLAMA3, OLMO2) and pretokenize digits into blocks of 3. We make one modification, by grouping digits into 3 from the right rather than from the left, so that, e.g., `1000` would be pretokenized as [`1`, `000`] instead of [`100`, `0`]. This choice was recently found to yield improved performance on math benchmarks, even when applied solely at inference time [Singh and Strouse, 2024]. Digit pretokenization is enforced with the following regex.

```
(?=(\d{3})+(?!\d))
```

Special casing of colon In order to make our tokenizer compatible with the common question-answering format where the prompt ends with a colon and the continuation is expected to start with a space, we "special-case" colon by preventing the algorithm from learning any tokens that contain ": " as a substring. Without this fix, common question/answer prompts might produce distorted distributions over completions. Please see §4.1.5 for further discussion. This affects the resulting tokenizer minimally in terms of the learned vocabulary.

4.1.4 Encoding efficiency

A tokenizer's encoding efficiency can be measured in *bytes-per-token*, i.e., how many UTF-8 bytes are encoded, on average, in each token over a large corpus of text (see calculation in Figure 4.1). We train a series of tokenizers on a 10 GB subset of data from OLMO 2's pretraining corpus and evaluate encoding efficiency on a held-out subset.

Shown in Figure 4.1, SuperBPE scales much better with vocabulary size than does BPE. BPE quickly plateaus around a vocabulary size of ~50K, achieving 4.45 bytes-per-token at a vocabulary size of 200k. In fact, even with infinite vocabulary size (namely, if *every* whitespace-delimited word were in the vocabulary), BPE cannot exceed 4.68 bytes-per-token, i.e., the average word length in the held-out subset.

SuperBPE exceeds this upper bound with a mere ~12k vocabulary size and reaches 5.55 bytes-per-token at 50K and 6.63 at 200k.

Surprisingly, SuperBPE is also more efficient than BPE with whitespace pretokenization completely disabled. Since BPE is a greedy algorithm, completely disabling whitespace pretokenization may cause it to make highly suboptimal choices early on. In particular, tokens in this setting often consist of the end of the previous word and start of the next word, as opposed to sequences of complete words. By keeping whitespace pretokenization on at the beginning, we can avoid suboptimal choices while still obtaining a tokenizer with superwords.

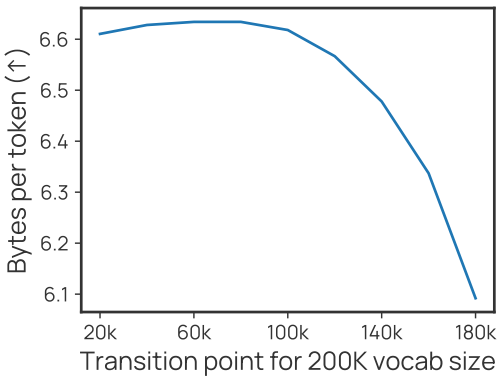


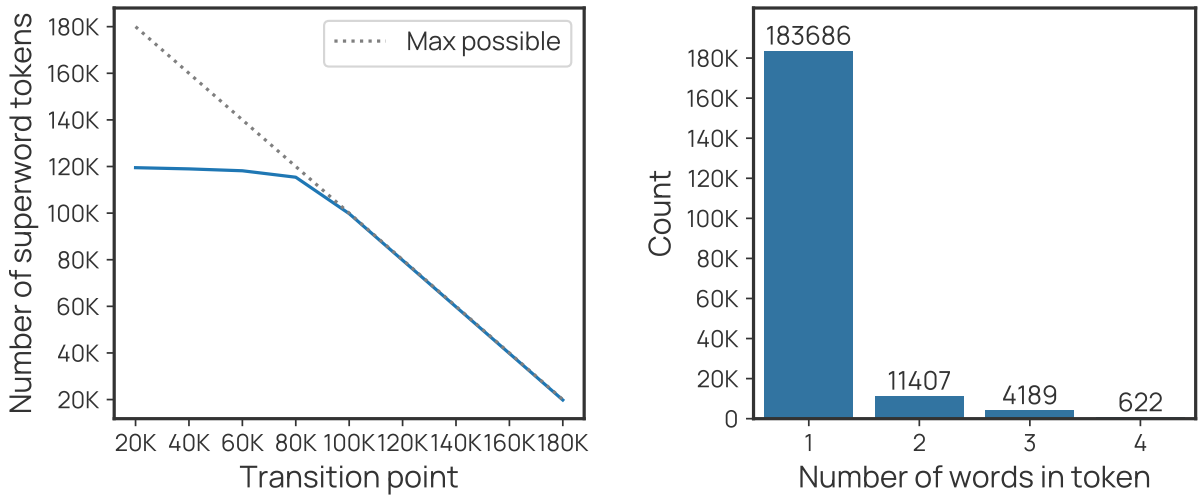
Figure 4.3: Encoding efficiency varies smoothly with the choice of transition point t in SuperBPE's pretokenization curriculum.

Figure 4.3 shows how SuperBPE's encoding efficiency depends on the choice of transition point t. The relationship is smooth, with $t = 80$k achieving the best encoding efficiency. However, we will see in our experiments that the optimal tokenizer for LM pretraining is not necessarily the most encoding-efficient.

4.1.5 Other Analysis

Superword token analysis How many superword tokens are in SuperBPE tokenizers? While the second stage of the pretokenization curriculum allows learning of superword tokens, subword tokens can still be learned. Shown in Figure 4.4a, for transition points $t < 80$k, the number of superword tokens is relatively steady around 120k. Past $t > 100$k, almost all tokens learned in Stage 2 are superword tokens. Figure 4.4b shows the number of whitespace-delimited words in the superword tokens of SuperBPE with $t = 180$k.

Analysis of token frequencies in encoding We also analyze token frequency statistics under BPE versus SuperBPE tokenizers. Figure 4.5a shows the relation between token rank (in frequency) and frequency. While tokens in BPE demonstrate a standard Zipfian relation, the slope of SuperBPE curves have a more shallow slope, meaning that the rate of decay in token frequency is smaller. The smaller propor-



(a) Proportion of superword tokens in the vocabulary

(b) Superword length distribution

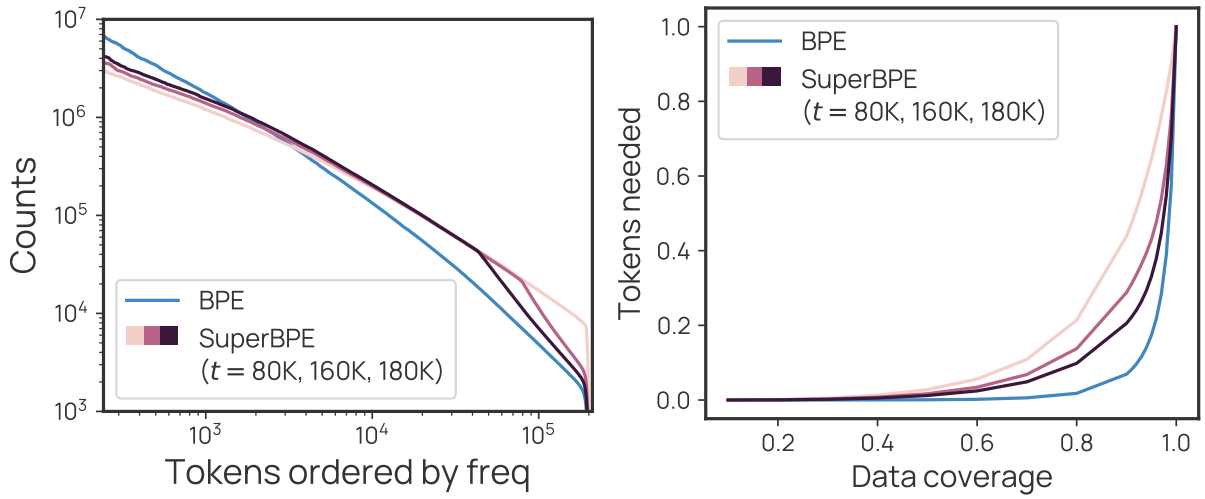
Figure 4.4: (Left) The number of superword tokens in a SuperBPE tokenizer, as a function of the transition point. A superword token is any token that violates the whitespace pretokenization rule from Stage 1. With an early transition point of $t = 60$K, about 85% of the tokens learned in Stage 2 are superword tokens. For $t > 100$k, close to 100% of Stage 2 tokens are superwords. (Right) The distribution of superword token lengths in terms of number of words, for $t = 180$k.

tion of tokens with very low counts may reduce prevalence and severity of glitch tokens [Rumbelow and Watkins, 2023; Land and Bartolo, 2024].

Figure 4.5b shows the minimum number of tokens from the vocabulary needed to cover any given proportion of data. For BPE, the relation is striking—only 57% of tokens are needed to encode 99% of the data! The remaining tokens make up a long tail of infrequent tokens. In contrast, SuperBPE tokenizers make better use of the vocabulary. For $t = 80$k and $t = 180$k, this statistic is 90% and 70% of tokens, respectively.

Distributional Distortion at the Prompt Boundary Prior work [Lundberg, 2023; Phan et al., 2024] has shown that LMs using BPE tokenizers may produce distorted generations due to the forced partition in tokenization between a prompt and its completion. This issue stems from the fact that users typically desire completions conditioned on a text prompt. The natural approach to obtaining such completions is to take the prompt, tokenize it with the proper tokenizer, and then sample a completion of the resulting token sequence from the LM.

For a simple example of how this can go wrong, consider a tokenizer with base vocabulary of `A` and



(a) Token frequency distribution

(b) Data covering

Figure 4.5: (Left) Token counts when ordered by frequency. The rate of decay in token frequency is smaller. (Right) The minimum number of tokens needed to cover a given proportion of the data. SuperBPE tokenizers make better use of the vocabulary, while BPE tokenizers have a long tail of infrequent tokens.

`B` and a single merge forming the token `AB`. Let's suppose we trained a model using this tokenizer on the strings "AA", "AB", and "BB" with equal proportions. If we condition on the text prefix "A", there are two equally probable continuations: "A" and "B". However, `A` is the only valid completion of the token prefix `A`, since the token `B` never follows the token `A` during training. In other words, the prompt-completion pair (`A`, `B`) is canonically tokenized using a token that crosses the boundary between the prompt and the completion.

While this problem is shared by all BPE tokenizers, it can be partially mitigated by pretokenization: if the prompt and the completion are separated during the pretokenization step, then it is impossible for a token to cross the boundary. This fix tends to work well for English, where the completion is typically expected to begin with whitespace, so whitespace pretokenization would apply. However, there are many settings where whitespace pretokenization cannot fix the underlying issue, including natural languages that do not use whitespace to separate words (like Chinese and Japanese), programming languages, and constrained generation [Lundberg, 2023; Ribeiro, 2023].

Several fixes for this issue have been proposed: at training time, token merges can be randomly dropped [Provilkov et al., 2020; Sims et al., 2025; DeepSeek-AI, 2025] to expose LMs to the internal makeup of tokens; at inference time, options include token healing [Lundberg, 2023], algorithmic correction [Phan

et al., 2024], and enumeration of all relevant segmentations of the prompt [Vieira et al., 2024]. We leave a detailed comparison of these techniques to future work.

Additionally, the issue does not apply at all to models that separate the user's input from the model's response using special tokens, as is typical for chat models.

4.2 Experiments

In our main experiments, we pretrain models from scratch while fixing the total training FLOPs and vocabulary size, changing only the algorithm for learning the vocabulary. All models were pretrained on 32 8×H100 nodes.

4.2.1 Setup

We first pretrain 8B models with BPE and SuperBPE tokenizers. We use the OLMo2 7B [OLMo et al., 2024] training configuration,[3] including the model architecture, training hyperparameters, and pretraining corpus, but reduce the total number of training steps to correspond to ~330B tokens (compared to 4T). Following prior work [Pagnoni et al., 2024], we also fix the *effective* context size (measured in bytes) for each model. This prevents SuperBPE models from gaining an advantage by seeing more textual context for the same next-token prediction (we provide analysis on this in §4.3.3). Since more efficient models have a shorter context length in tokens, the training steps are adjusted accordingly to match the total train FLOPs at the end of training.[4] Note that in this setting, a same-sized SuperBPE model uses fewer inference FLOPs than the BPE model.

We fix the vocabulary size of all tokenizers to 200,000 (in the same ballpark as, e.g., GEMMA at 250k [Google, 2024], GPT-4O at 200k, and LLAMA3 at 130k [Meta, 2024]).[5] We consider three transition points for SuperBPE: $t = 80$k, which has the best encoding efficiency, and two later transitions, $t = 160$k and $t = 180$k. All tokenizers are trained on the same 10 GB subset of OLMo2's pretraining mix.

[3] OLMo2 7B has 7.30B parameters, while our 8B BPE and SuperBPE models have 8.12B parameters due to their increased vocabulary size.

[4] In practice, models using our more efficient tokenizers could shift some or all of the "saved" context FLOPs to longer effective contexts instead of more training steps.

[5] For 8B models, a 200k vocabulary size is close to the recommendation of Tao et al. [2024] based on primarily English data. We fix the vocabulary size to simplify comparisons between models.

Category	Task	BPE	SuperBPE	Δ
Knowledge	ARC-Easy (MC)	46.6	**67.1**	+20.5**
	ARC-Challenge (MC)	35.1	**50.6**	+15.5**
	Jeopardy (EM)	**42.1**	41.8	−0.3
	MMLU (MC)	36.5	**44.7**	+8.2**
	OpenbookQA (MC)	33.2	**54.4**	+21.2**
	TriviaQA (EM)	60.6	**61.3**	+0.7
	WikidataQA (EM)	69.7	**70.9**	+1.2*
Math & Reasoning	Arithmetic (EM)	54.8	**59.3**	+4.5**
	GSM8K (EM)	6.4	**6.7**	+0.3
	LSAT-AR (MC)	21.3	**23.0**	+1.7
	Operators (EM)	**35.5**	33.6	−1.9
	Repeat-Copy-Logic (EM)	3.1	**6.2**	+3.1
Coding	HumanEval (pass@10)	**15.9**	13.4	−2.5
	MBPP (pass@10)	27.5	**28.3**	+0.8
Reading Comprehension	BoolQ (MC)	59.7	**64.6**	+4.9**
	CoQA (EM)	12.6	**13.2**	+0.6
	DROP (EM)	31.3	**31.4**	+0.1
	HotpotQA (EM)	53.5	**55.2**	+1.7*
	SQuAD (EM)	75.1	**75.8**	+0.7
Commonsense	CommonsenseQA (MC)	33.5	**53.8**	+20.3**
	COPA (MC)	77.0	**85.8**	+8.8**
	PIQA (MC)	55.2	**59.8**	+4.6*
	Winograd (MC)	50.4	**53.1**	+2.7
	Winogrande (MC)	47.3	**52.6**	+5.3*
Language Understanding	HellaSwag (MC)	29.7	**33.7**	+4.0**
	LAMBADA (EM)	**77.0**	70.6	−6.4**
	Language Identification (EM)	8.8	**9.0**	+0.2
String Manipulation	CS Algorithms (EM)	46.1	**48.6**	+2.5
	CUTE (EM)	31.3	**32.6**	+1.3
	Dyck-Languages (EM)	**15.9**	14.2	−1.7
Average		39.8	**43.8**	+4.0

Table 4.1: Performance of BPE and SuperBPE models (with transition point $t = 180$k) on 30 downstream tasks. The two models are fixed in model parameters (8B), vocabulary size (200k), and training FLOPs (corresponding to ~330B tokens), differing only in their algorithm for learning the vocabulary. The SuperBPE model outperforms the baseline on 25 of 30 tasks and requires 27% less compute at inference time. See Figure 4.6 for the moving task average during pretraining and **??** for further evaluation details. $^{*}p < 0.05$, $^{**}p < 0.005$ under a McNemar test.

	BPE 8B	SuperBPE 8B			SuperBPE 11B
SuperBPE transition point		t = 80k	t = 160k	t = 180k	t = 180k
Parameter count (billion)	8.12	8.12	8.12	8.12	11.30
Train steps	76,543	118,419	112,722	107,982	77,525
Average context length (bytes)	18,262	18,272	18,263	18,268	18,268
Vocabulary size	200k	200k	200k	200k	200k
Context length (tokens)	4,096	2,756	2,884	3,000	3,000
Encoding efficiency (bytes/token)	4.46	6.63	6.33	6.09	6.09
Train compute (10^{21} FLOPs)	17.2	17.2	17.2	17.2	17.2
Inference compute (10^{9} FLOPs/byte)	3.75	2.42	2.54	2.65	3.75

Table 4.2: Training setup for the models we compare. We fix the vocabulary size and effective context size (measured in bytes) for each model and adjust the total number of training steps accordingly so that each model has the same total train budget (in FLOPs). The 8B SuperBPE models match the 8B BPE model in train compute but use less inference compute; the 11B SuperBPE model matches the 8B baseline in both train *and* inference compute. Numbers fixed across model settings are highlighted in the same color.

We also train a slightly larger 11B SuperBPE model with $t = 180$k, which approximately matches the 8B BPE baseline in total bytes of training data seen as well as both train *and* inference compute. See Table 4.2 for exact specifications for all runs.

4.2.2 Results on downstream tasks

We evaluate SuperBPE on 30 benchmarks covering knowledge, math & reasoning, coding, reading comprehension, common sense, language understanding, and string manipulation. The full evaluation suite is shown in Table 4.1 and evaluation details are in **??**.

Figure 4.6 shows the task average during pretraining. All SuperBPE models substantially outperform the BPE baseline at the end of training. The strongest 8B SuperBPE model, which has transition point $t = 180$k (the latest one we consider), outperforms the baseline by 4.0% on average and wins on 25 of 30 individual tasks. Table 4.1 shows the per-task performance for this model (see **??** for results for the other models). The largest gains are on multiple choice tasks; when considering these alone, the performance improvement grows to +9.7%. The only task on which SuperBPE loses in a statistically significant way is LAMBADA; here, we observe that SuperBPE is actually ahead for the majority of training checkpoints, but accuracy dips at the end from 75.8% to 70.6%.

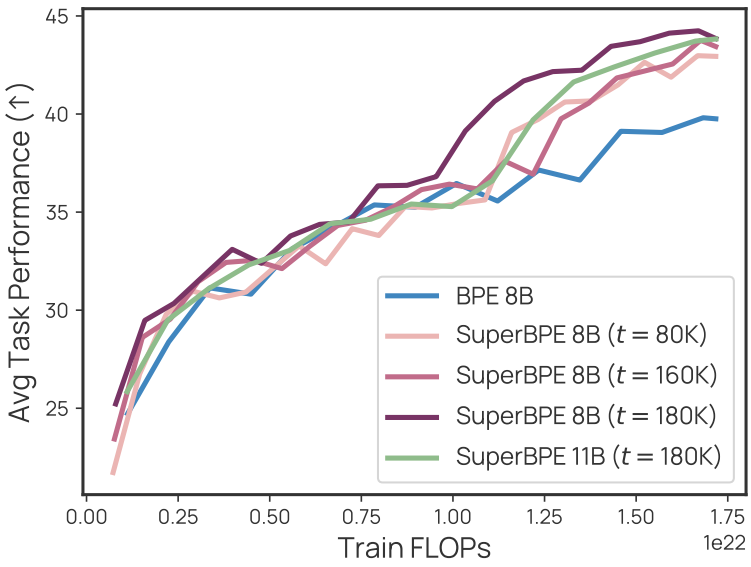


Figure 4.6: Average task performance on 30 downstream tasks, evaluated at every 5000 steps in model pretraining. We see that SuperBPE models consistently outperform the baseline that uses a BPE tokenizer. All compared models share the same vocabulary size and train budget; t denotes the transition point in SuperBPE's pretokenization curriculum.

Notably, while the choice of transition point affects the performance of the resulting model, all reasonable choices are significantly stronger than the baseline. When using the most encoding-efficient transition point, i.e., $t = 80$k, we see a +3.1% task improvement over BPE and inference compute reduced by 35%. Later transition points empirically cede some gains in encoding efficiency in exchange for further improvements in performance.[6]

4.3 Analysis

4.3.1 Language modeling

Following prior work [Biderman et al., 2023; Xue et al., 2022; Yu et al., 2023; Wang et al., 2024], we evaluate language modeling performance using *bits-per-byte* (BPB), which normalizes the loss by the tokenizer's encoding efficiency to fairly compare models with different tokenizers. This is necessary because longer tokens, on average, contain more information and therefore are more difficult to predict. Bits-per-byte is

[6]This finding adds to the ongoing debate about the relationship between tokenization compression and LM performance [Gallé, 2019; Goldman et al., 2024; Schmidt et al., 2024], providing further evidence that higher compression does not necessarily improve performance.

defined as $\text{BPB}(x) = \mathcal{L}_{\text{CE}}(x)/(\ln(2) \cdot n_{\text{bytes}})$, where n_{bytes} is the length of the text in bytes and $\mathcal{L}_{\text{CE}}(x)$ is the sum of the cross-entropy loss over the entire text.[7] We find that BPE 8B, SuperBPE 8B ($t = 180$k), and SuperBPE 11B attain 0.7465, 0.7482, and 0.7445 BPB, respectively, at the end of training. Although these numbers do not differ appreciably, the ranking of models according to BPB and downstream task performance are not consistent.

4.3.2 Loss distribution analysis

Why does the SuperBPE 8B model achieve slightly higher normalized language modeling loss (§4.3.1) than the baseline BPE model despite outperforming it on a wide variety of downstream tasks (§4.2.2)? To investigate this, we plot the distribution of per-token BPB[8] for both models on data sampled from the pretraining data mixture in Figure 4.7.

Although the BPE and SuperBPE models have very similar BPB on average, we see that loss is distributed very differently over the training data. Compared to the baseline, the SuperBPE model makes fewer predictions with either very high or very low loss.

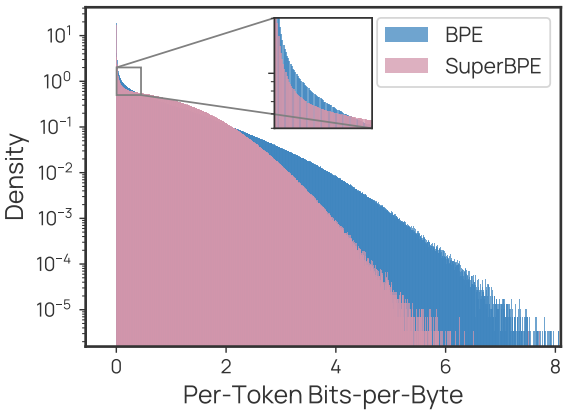


Figure 4.7: Histogram of per-token losses for both models from Table 4.1, measured over a large corpus of text. We observe that the SuperBPE model is a more consistent performer, making fewer predictions with very high or very low loss.

Low-loss tokens. We find that the reduction in low-loss tokens is attributable to a small set of extremely common words that the BPE model can easily predict, but are not available to SuperBPE as they are merged into larger superword tokens. For instance, the tokens `_the`, `_of`, and `_to` (the three most common words in the corpus) appear an order of magnitude more often under BPE than SuperBPE in the same corpus of text. *When excluding these three token types alone, the BPB ranking reverses*, with SuperBPE achieving 0.02 lower BPB than BPE.

[7] Bits-per-byte of different models are considered comparable because total cross-entropy loss is a universal quantity representing the number of additional bits required to reconstruct the text given the model. This quantity is normalized by the number of bytes for easier interpretation.

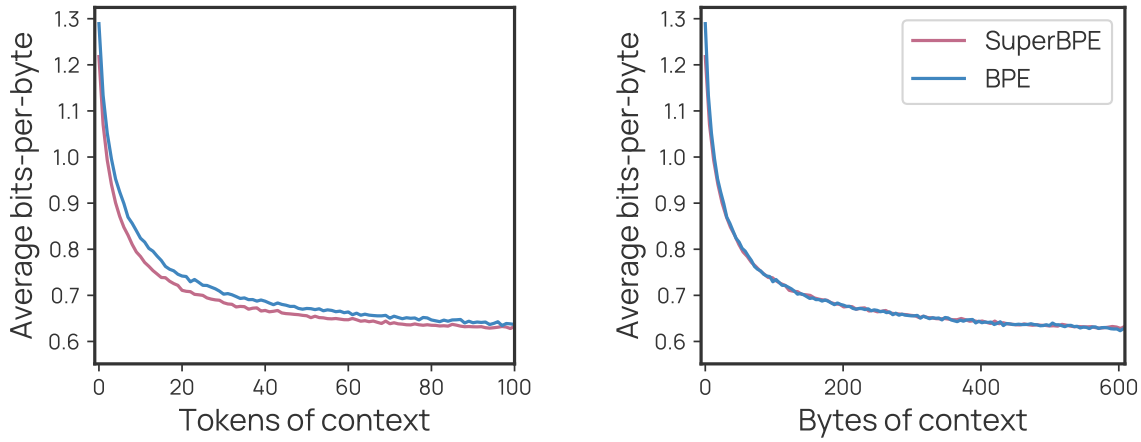
[8] The per-token BPB is the per-token loss (in bits) divided by the average encoding efficiency.

POS tag	#	Example Tokens
NN, IN	906	_case_of, _hint_of, _availability_of, _emphasis_on, _distinction_between
VB, DT	566	_reached_a, _discovered_the, _identify_the, _becomes_a, _issued_a
DT, NN	498	_this_month, _no_idea, _the_earth, _the_maximum, _this_stuff
IN, NN	406	_on_top, _by_accident, _in_effect, _for_lunch, _in_front
IN, DT	379	_on_the, _without_a, _alongside_the, _for_each
IN, DT, NN	333	_for_a_living, _by_the_way, _into_the_future, _in_the_midst
NN, IN, DT	270	_position_of_the, _component_of_the, _review_of_the, _example_of_this
IN, DT, JJ	145	_like_any_other, _with_each_other, _for_a_short, _of_the_entire
VB, IN, DT	121	_worked_as_a, _based_on_the, _combined_with_the, _turned_into_a
IN, DT, NN, IN	33	_at_the_time_of, _in_the_presence_of, _in_the_middle_of, _in_a_way_that
,, CC, PRP, VB	20	,_and_it_was, ,_but_I_think, ,_but_I_have, ,_but_I_am
IN, DT, JJ, NN	18	_in_the_long_run, _on_the_other_hand, _for_the_first_time, _in_the_same_way

Table 4.3: The most common POS tags for tokens of 2, 3, and 4 words in SuperBPE, along with random example tokens for each tag. NN = noun, IN = preposition, VB = verb, DT = determiner, CC = conjunction, JJ = adjective, and PRP = pronoun.

The reduction in low-loss tokens also makes sense from a qualitative linguistic analysis of SuperBPE tokens. In Table 4.3, we show the most common POS tags among superword tokens in SuperBPE along with random examples for each tag. The tokens often capture common multi-word expressions (*by accident*, *of course*, *for a living*) that function as a single semantic unit [Schneider et al., 2014]. As an example, prepositions (IN) figure prominently in superword tokens (e.g., *depend on*, *distinction between*) and require lexeme-specific memorization. The individual words in these fixed expressions are often semantically vacuous and have little possible variation in context, so they are easy to predict once memorized.

High-loss tokens. On the other hand, the much thinner tail of very high-loss tokens shows that, *in the worst case, the SuperBPE model consistently puts more probability mass on the correct token.* On average, we expect models to suffer high loss on tokens that are difficult to predict. This may explain why SuperBPE can outperform BPE on downstream tasks but have higher average BPB: the tokens scored in task evaluations tend to be among the hardest to predict. This is consistent with prior findings that models generally continue to improve in downstream tasks even as their overall loss plateaus due to improving on a narrow and difficult slice of the distribution [Liu et al., 2023b].



(a) BPB vs tokens of context

(b) BPB vs bytes of context

Figure 4.8: When comparing the normalized loss of the next token, controlling for preceding tokens of context gives SuperBPE an advantage, while controlling for bytes of context gives a close match between BPE and SuperBPE.

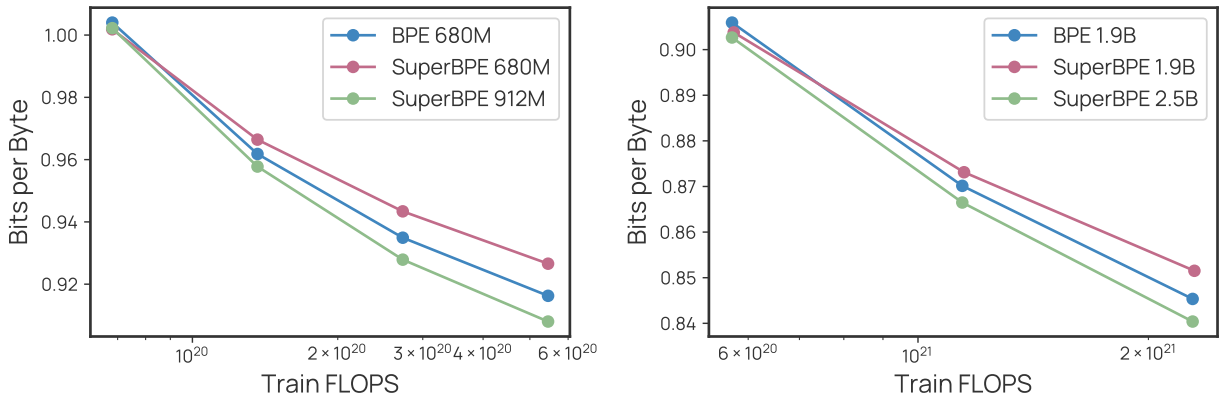
4.3.3 How BPB varies with context length

In our main experiments, we adjust the context size of SuperBPE models to match the *effective* context size of the BPE model in raw text. To justify this design choice, we show that the next token becomes easier to predict as a function of the preceding context in bytes (not tokens). Figure 4.8 shows the average BPB at every token index (left) vs byte index (right) — when measured at fixed token indices, SuperBPE has an advantage from seeing more context (achieving lower loss on average at the same token index), whereas at fixed byte indices, this advantage goes away.

4.3.4 Scaling

To characterize the scaling behavior of SuperBPE, we also perform experiments at smaller scales.[9] For each BPE model, we train two SuperBPE models that match the training budget of each baseline BPE model, one that matches the baseline in parameter count (analogous to SuperBPE 8B) and a larger model that matches in both train and inference compute (analogous to SuperBPE 11B). We focus on the $t = 180$k tokenizer to reduce complexity. We present the exact model hyperparameters for all model sizes used in

[9]For scaling, we focus on BPB since our downstream evaluations are too noisy for our small models to make meaningful comparisons.



(a) 680M model size

(b) 1.9B model size

Figure 4.9: Scaling results for 680M and 1.9B baseline model sizes. Compared to the **BPE** baseline, **SuperBPE with matching parameter count** achieves lower BPB in the under-trained regime, while **SuperBPE with matching inference compute** achieves lower BPB than the baseline at every model size and every training budget tested. Note that *BPB comparisons between BPE and SuperBPE models do not track downstream task accuracy* due to differences in how BPE and SuperBPE models distribute loss over tokens (§4.3.2).

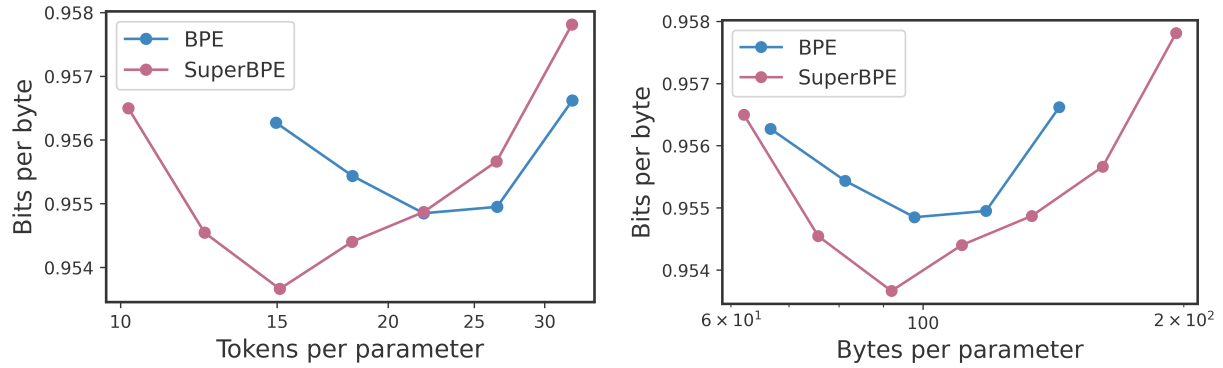


Figure 4.10: In IsoFLOP scaling experiments, we find that the optimal ratio of training tokens to model parameters is lower for SuperBPE models than BPE models. All points in the plot use the same training budget: points on the left represent larger models trained on fewer tokens, while points on the right are smaller models trained on more tokens. (Left) The minimum on each curve represents the optimal ratio of tokens (T) to parameters (P), which we observe to be ~22 T/P for BPE and ~15 T/P for BPE. (Right) When rescaling

our experiments in Table 4.4.

Scaling the number of training tokens In Figure 4.9, we hold the model size fixed and increase the number of training tokens. We perform runs ranging from $\approx$ 20 to $\approx$ 80 tokens per parameter to observe the trend with respect to training duration. In the under-trained regime, both SuperBPE models achieve lower BPB than the baseline. In the over-trained regime, the ranking from worst to best is SuperBPE (matching parameter count), BPE, and SuperBPE (matching inference compute). Additionally, the separation between the models increases with further over-training.

Scaling the training compute In Figure 4.11, we hold the ratio of train tokens to model parameters fixed and vary both the model size and the number of training tokens. The general trends observed from these results are that matching inference compute is almost universally the best, while matching parameter count tends to be worse than the baseline except in the undertrained regime, where it is better than the baseline. The differences between the different settings increases with overtraining, but decreases when scaling both model size and training tokens at the same time.

IsoFLOP scaling In Figure 4.10, we fix a training budget of 1.4×10^9 FLOPs and perform a sweep over the ratio of tokens seen during training to the model parameter count. Points on the left are larger models trained on fewer tokens, while points on the right are smaller models trained on more tokens. The minimum on each curve represents the optimal ratio of tokens (T) to parameters (P). In (left), we see that the optimal T/P ratio for BPE is 22 tokens/parameter as expected, but for SuperBPE it is roughly 30% lower at 15 tokens/parameter! Remarkably, this corresponds to the same underlying ratio of train *bytes* / param, pointing to the possibility that compute optimality is actually a constant ratio of training bytes to parameters.

4.4 Related work

Tokenization beyond subwords Prior work has explored processing text at multiple levels of granularity [Lai et al., 2021; Zhang et al., 2021] or creating multi-word tokens through frequency-based identification of n-grams [Gee et al., 2023; Kumar and Thawani, 2022]. However, these were explored in limited

	680M	910M	1.9B	2.5B	8B	11B
Parameter count	678.2M	912.5M	1.893B	2.536B	8.115B	11.30B
Model dimension	1024	1,216	2,048	2,304	4,096	4,608
MLP hidden dimension	8,192	9,728	16,384	18,432	22,016	24,704
Head dimension	64	64	128	128	128	128
Number of heads	16	19	16	18	32	36
Number of layers	16	18	16	19	32	37
Vocabulary size	20,0005	20,0005	20,0005	20,0005	20,0005	20,0005

Table 4.4: Model parameters for all model sizes. Model sizes 910M, 2.5B, and 11B are scaled versions of 680M, 1.9B, and 8B respectively. All other parameters match those of OLMO 300M (from the OLMO model ladder) for sizes 680M and 910M, OLMO 1B for sizes 1.9B and 2.5B, or OLMO2 7B for sizes 8B and 11B, respectively. Maximum sequence length values for various tokenizers are listed in Table 4.2.

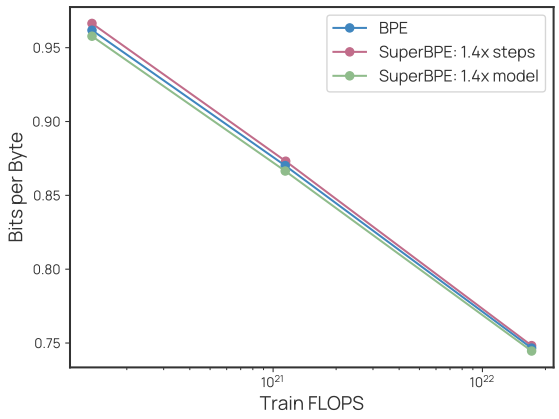


Figure 4.11: Results for scaling both model parameters and train tokens proportionally. Compared to the **BPE** baseline, we consider a **SuperBPE model that matches parameter count** and a **SuperBPE model that matches inference compute**. Here we see the spread between the three settings decreases with scale.

experimental contexts (mainly for machine translation) and had mixed effectiveness. Naively disabling pre-tokenization in BPE has been found to severely degrade model performance [Dagan et al., 2024; Schmidt et al., 2024; Kudo, 2018], although this approach may be more promising for unigram tokenization [Kudo and Richardson, 2018], as adopted by Jurassic [Lieber et al., 2021] and BloombergGpt [Wu et al., 2023b]. In concurrent work, Huang et al. [2025a] disentangle input and output vocabularies, expanding only the former to include n-gram tokens. Their method requires significant modifications of the LM input component and considers fixed length of n-grams.

Multi-token prediction Multi-token prediction (MTP) equips LMs with some extra parameters to predict multiple tokens in a single time step [Qi et al., 2020; Gloeckle et al., 2024] and was recently adopted by Deepseek-V3, though the MTP module is discarded at inference-time. MTP's effectiveness corroborates that LMs are capable of predicting more than one subword in a forward pass. However, these approaches fix the number of tokens predicted in each time step and require modifications to the architecture and training objective. We note that the benefits of MTP and superword tokens may be orthogonal.

n-gram language models Our work is loosely related to n-gram LMs, which incorporate n-gram statistics into the next-word prediction [Brants et al., 2007; Liu et al., 2024b]. Both point to the value of incorporating knowledge of common multi-word units.

Tokenizer-free language modeling Some works have explored the possibility of completely removing tokenization from LMs and directly modeling text as a sequence of bytes [Clark et al., 2022; Xue et al., 2022; Wang et al., 2024]. To overcome the increased compute requirement due to expanded sequence lengths, alternative architectures have been proposed that either segment bytes into fixed-length patches [Tay et al., 2022; Yu et al., 2023] or dynamically predict patch boundaries with sub-modules [Nawrot et al., 2023; Pagnoni et al., 2024; Ahia et al., 2024; Hwang et al., 2026]; these dynamic patches have been qualitatively observed to span multiple words.

Tokenizer transfer Many methods have been proposed to adapt models after training to use a different tokenizer. These may rely on intervention during pretraining [Chen et al., 2023], continued training for a subset of layers [Marchisio et al., 2023], or self-distillation [Minixhofer et al., 2025], combined with

heuristic [Minixhofer et al., 2022; Gee et al., 2022; Tran, 2020; Liu et al., 2024c; Dobler and De Melo, 2023] or hypernetwork-based [Minixhofer et al., 2024] initialization of a fresh embedding matrix. These methods may be used to upgrade existing models to use SuperBPE tokenizers, with the goal of reducing inference cost while maintaining performance. We leave this direction to future work.

Pretokenization Pretokenization defines how the text is split in order to prevent certain pairs of tokens from being merged. GPT-2 [Radford et al., 2019] introduced a regular expression (regex) which defines the pretokenization pattern. These regex strings have gained complexity over time; GPT-3.5 limits the number of digits in numerical tokens to 3, and allows single punctuation to be merged with the start of words (presumably to accommodate code, as it allows `.get` to be a single token). Prior work has shown that, for instance, digit pretokenization choices [Nogueira et al., 2021; Thawani et al., 2021; Singh and Strouse, 2024] can significantly impact arithmetic performance. It is also likely that pretokenization affects different languages differently [Velayuthan and Sarveswaran, 2025; Ahia et al., 2023], due to natural statistics of the average word length, which acts as an upper bound on encoding efficiency in that language under subword tokenization. Nonetheless, the effectiveness of many pretokenization choices have not been thoroughly studied.

Internal representation of semantic units Previous work has showed that the early layers of the LM may "aggregate" information over multi-token entities (e.g., [`_New`, `_York`]) into the *last* token's (e.g., `_York`) hidden representation [Meng et al., 2022; Kaplan et al., 2025; Lad et al., 2024]. This suggests that LMs naturally learn multi-word representations, and segmentating text into more semantically cohesive units at the input level (e.g., having `_New_York` as a single token) may simplify this process.

4.5 Conclusion

Although tokenization lies at the foundation of language modeling, acting as the lens through which models view text, the algorithms in use have remained largely unchanged over the past decade. SuperBPE builds on the observation that tokens need not be limited to subwords, extending the BPE algorithm to superword tokens. When replacing subword BPE tokenizers with SuperBPE tokenizers in pretraining, we

find that language models perform better over a large suite of downstream tasks, while also being substantially more efficient at inference time. These benefits are achieved without modifying the underlying model architecture, making SuperBPE a compelling alternative to BPE that seamlessly integrates with modern language model ecosystems.

Chapter 5

Conclusion

5.1 Community Impact and Follow-Up Work

We have been very encouraged to see the research community build on the work in this dissertation. In this section, we give a non-exhaustive overview of some of these directions.

Applications of proxy-tuning Diverse applications of proxy-tuning have been explored by subsequent work. Proxy-tuning can be applied directly to elicit reasoning-style rollouts from strong base models without direct training [Zhang et al., 2026; Xiao et al., 2026; Ouyang et al., 2026]. By flipping the assignment of the expert and anti-expert and operating on a tuned model, proxy-tuning can be used to *undo* alignment and recover unaligned behavior, revealing safety risks [Zhou et al., 2024; Zhao et al., 2025]. Similarly, unlearning can be achieved by using an anti-expert trained deliberately on the data that should be unlearned [Huang et al., 2025b; Ji et al., 2024].

One strength of proxy-tuning is that true finetuning only needs to be performed once to obtain the small expert, after which the effect of tuning can be "applied" to an arbitrary number of models. Qian et al. [2024] proxy-tunes thousands of pretrained model checkpoints, cheaply identifying the best starting point for post-training while paying the cost of finetuning only once.

Additionally, proxy-tuning makes it possible to achieve finetuning without exposing the model directly to the finetuning data, which can be useful in the case of private data. Leveraging this insight, Zhang et al. [2024] enabled user customization by tuning an on-device expert and combining its predictions with a base

model on the cloud at inference-time.

Other works have explored algorithmic extensions of proxy-tuning, such as dynamically assigning weights to scale the contributions of different models in the ensemble [Mavromatis et al., 2024; Fan et al., 2024; Du et al., 2024]. In our own subsequent work, we showed that proxy-tuning can be applied to steer models towards multiple weighted objectives simultaneously [Shi et al., 2024].

Superword tokenization SuperBPE has seen industry adoption by Moondream's vision-language models and the K-EXAONE models from LG AI [Choi et al., 2026]. Schmidt et al. [2026b] provide a faster implementation of SuperBPE, addressing a key limitation of the original work, where tokenizer training required significantly more memory and longer CPU time. We are also honored that SuperBPE is discussed in the 3rd edition of the Speech and Language Processing textbook [Jurafsky and Martin, 2026].

While our original work considered only English language modeling, other works have explored SuperBPE in multilingual use cases. Arnett et al. [2026] show that disparities in how efficiently subword tokenizers encode different languages is largely attributable to whitespace pretokenization, as different languages naturally vary in the number of whitespace-delimited words needed to express the same meaning, which forms an upper bound on the achievable compression. They then find that SuperBPE tokenization, by lifting the subword restriction, leads to more equitable (and better) compression across different languages. SuperBPE is also adopted in the multilingual tokenizer recipe of [Rana et al., 2026].

Superword tokens can be incorporated into the vocabulary after pretraining. In particular, Zhao et al. [2026] explore this for reasoning, where low-entropy phrases are often used to scaffold the reasoning process (*Let me try*, *Wait, hold on*, *Let us verify*) without bearing meaningful information. They add superword tokens for these repetitive phrases to the vocabulary and perform lightweight finetuning for models to adopt them, achieving a significant improvement in efficiency while preserving downstream performance.

Since SuperBPE, other superword tokenization algorithms have also been introduced [Schmidt et al., 2025; Nakash et al., 2025; Tănase and Pelican, 2025; Dong and Su, 2025], including approaches that directly optimize for compression [Tempus et al., 2026; Schmidt et al., 2026a]. Finally, Chung and Kim [2026] provide a theoretical analysis of the benefit of SuperBPE based on Kolmogorov complexity.

Superword embeddings Orthogonal to changing the tokenizer, several recent works explored the potential of incorporating auxiliary n-gram embedding tables into language models, which are used to augment token representations at each position [Cheng et al., 2026; Liu et al., 2026]. This presents a new axis for scaling up sparse capacity, as additional embedding parameters increase the capacity of the model without increasing the compute per-token. This line of work shares a similar spirit to superword tokenization, in that richer embeddings increase the effective capacity of models by turning repetitive computation of multi-word representations into a single look-up.

Dynamic tokenization Recent work has developed promising new architectures for dynamic tokenization trained end-to-end to segment byte sequences into "patches", which can naturally span multiple whitespace-delimited words. Indeed, in H-Net's [Hwang et al., 2026] iterative compression scheme, the authors qualitatively observe that the first stage primarily segments on whitespace while the second stage merges meaningful sequences of words, mirroring the 2-stage training curriculum of SuperBPE which learns subwords before superwords.

Compression and scaling laws In §4.3.4, we found that when using SuperBPE, the optimal allocation of training compute between model size and training tokens involves training a *larger* model on *fewer* tokens, compared to when using BPE. In subsequent work, we expanded this observation into a rigorous study of how compression interacts with scaling laws [Limisiewicz et al., 2026], confirming our hypothesis that compute optimality is actually defined by a constant ratio between training *bytes* (not tokens) and model parameters.

5.2 Future Work

When you look closely, the tokenizer hides an expansive space of possibilities that promise to impact our language models in meaningful ways. In this section, we outline some directions for future research.

Improvements in representation The de facto methods for tokenization have changed little, while applications of language models have changed significantly. Currently, the design of the tokenizer is agnostic to the numerous natural and programming languages that make up the data, and pretokenization

rules have grown increasingly inflated while striving to do the right thing "on average". One promising direction of future work is to develop dedicated treatment for the segmentation of different domains of data. For instance, there is likely meaningful room for improvement through more semantically meaningful segmentation of code [Li et al., 2025], digits [Nogueira et al., 2021; Thawani et al., 2021; Singh and Strouse, 2024], non-English languages [Velayuthan and Sarveswaran, 2025; Ahia et al., 2023], and other structured data.

There may also be potential in more extreme changes to tokenization. For instance, rather than directly segmenting plain text into tokens, we could use modifier tokens to encode surface-form variation such as capitalization or tense. Taking this to the extreme, we could think of tokens as executing a "script" and include functional tokens such as backspace or copy-paste to efficiently encode redundancy or reduce sequence length on-the-fly.

More broadly, the quality of representations is shaped not only by the tokenizer but the architecture itself. For instance, the lines of work on dynamic tokenization [Kallini et al., 2025; Pagnoni et al., 2025; Hwang et al., 2026; Kallini et al., 2026] or scaling up the number of embedding parameters [Cheng et al., 2026; Liu et al., 2026; Huang et al., 2025a; Tseng and Sa, 2026; Yang et al., 2026] both improve representations without intervening on the tokenizer itself. Innovations in tokenization and architecture go hand-in-hand and are a promising direction for future work.

Tokenizer transfer Today, once a model is trained, it is bound to its tokenizer throughout its entire lifecycle, from pretraining to post-training to all future adaptations. This can extend even to descendants of the model that are obtained through token-level distillation. Combined with the exorbitant cost of pretraining, the choice of tokenizer becomes an extremely severe and high-stakes commitment, discouraging innovation and risk-taking in this area.

Therefore, algorithms for transferring a language model to a different tokenizer after it is trained would be invaluable. One promising way to approach this problem is through cross-tokenizer distillation, namely, distilling an existing model into a "new" model that differs only in the tokenizer. In particular, our prior work established the technology to transform probability distributions over the vocabulary into probability distributions over bytes [Hayase et al., 2026], which can be used to unify the vocabulary of models trained with disparate tokenizers and enable logit-level distillation.

Bibliography

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Valentin Hofmann, Tomasz Limisiewicz, Yulia Tsvetkov, and Noah A. Smith. 2024. MAGNET: Improving the multilingual fairness of language models with adaptive gradient-based tokenization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah Smith, and Yulia Tsvetkov. 2023. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Catherine Arnett, Tyler A. Chang, Stella Biderman, and Ben Bergen. 2026. Explaining and mitigating crosslingual tokenizer inequities. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models.

Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the AI: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8.

Max Bartolo, Tristan Thrush, Robin Jia, Sebastian Riedel, Pontus Stenetorp, and Douwe Kiela. 2021a. Improving question answering model robustness with synthetic adversarial data generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Max Bartolo, Tristan Thrush, Sebastian Riedel, Pontus Stenetorp, Robin Jia, and Douwe Kiela. 2021b. Models in the loop: Aiding crowdworkers with generative annotation assistants.

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

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling.

BIG-bench. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Samuel R. Bowman and George Dahl. 2021. What will it take to fix benchmarking in natural language understanding? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Samuel R. Bowman, Jennimaria Palomaki, Livio Baldini Soares, and Emily Pitler. 2020. New protocols and negative results for textual entailment data collection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

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.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision.

Anthony Chen, Pallavi Gudipati, Shayne Longpre, Xiao Ling, and Sameer Singh. 2021a. Evaluating entity disambiguation and the role of popularity in retrieval-based NLP. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jifan Chen, Eunsol Choi, and Greg Durrett. 2021b. Can NLI models verify QA systems' predictions? In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

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. 2021c. Evaluating large language models trained on code.

Y Chen, K Marchisio, R Raileanu, DI Adelani, P Stenetorp, S Riedel, and M Artetxe. 2023. Improving language plasticity via pretraining with active forgetting. In *Advances in Neural Information Processing Systems*.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Yukun Li, Han Zhang, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. 2026. Conditional memory via scalable lookup: A new axis of sparsity for large language models.

Eunbi Choi, Kibong Choi, Seokhee Hong, Junwon Hwang, Hyojin Jeon, Hyunjik Jo, Joonkee Kim, Seonghwan Kim, Soyeon Kim, Sunkyoung Kim, Yireun Kim, Yongil Kim, Haeju Lee, Jinsik Lee, Kyungmin Lee, Sangha Park, Heuiyeen Yeen, Hwan Chang, Stanley Jungkyu Choi, Yejin Choi, Jiwon Ham, Kijeong Jeon, Geunyeong Jeong, Gerrard Jeongwon Jo, Yonghwan Jo, Jiyeon Jung, Naeun Kang, Dohoon Kim, Euisoon Kim, Hayeon Kim, Hyosang Kim, Hyunseo Kim, Jieun Kim, Minu Kim, Myoungshin Kim, Unsol Kim, Youchul Kim, YoungJin Kim, Chaeeun Lee, Chaeyoon Lee, Changhun Lee, Dahm Lee, Edward Hwayoung Lee, Honglak Lee, Jinsang Lee, Jiyoung Lee, Sangeun Lee, Seungwon Lim, Solji Lim, Woohyung Lim, Chanwoo Moon, Jaewoo Park, Jinho Park, Yongmin Park, Hyerin Seo, Wooseok Seo, Yongwoo Song, Sejong Yang, Sihoon Yang, Chang En Yea, Sihyuk Yi, Chansik Yoon, Dongkeun Yoon, Sangyeon Yoon, and Hyeongu Yun. 2026. K-EXAONE technical report.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models.

Woojin Chung and Jeonghoon Kim. 2026. Exploiting vocabulary frequency imbalance in language model pre-training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Kenneth Church. 2011. How many multiword expressions do people know? In *Proceedings of the Workshop on Multiword Expressions: From Parsing and Generation to the Real World*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings*

of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. All that's 'human' is not gold: Evaluating human evaluation of generated text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Pablo Contreras Kallens and Morten H. Christiansen. 2022. Models of language and multiword expressions. *Frontiers in Artificial Intelligence*, 5.

Yiming Cui, Ziqing Yang, and Xin Yao. 2023. Efficient and effective text encoding for chinese llama and alpaca.

Gautier Dagan, Gabriel Synnaeve, and Baptiste Rozière. 2024. Getting the most out of your tokenizer for pre-training and domain adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24.

DeepSeek-AI. 2025. Deepseek-v3 technical report.

Haikang Deng and Colin Raffel. 2023. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston. 2019. Build it break it fix it for dialogue safety: Robustness from adversarial human attack. 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)*.

Konstantin Dobler and Gerard De Melo. 2023. Focus: Effective embedding initialization for monolingual specialization of multilingual models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Dong Dong and Weijie Su. 2025. Length-max tokenizer for language models.

Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A. Smith, and Yejin Choi. 2021. Scarecrow: A framework for scrutinizing machine text.

Yanrui Du, Sendong Zhao, Danyang Zhao, Ming Ma, Yuhan Chen, Liangyu Huo, Qing Yang, Dongliang Xu, and Bing Qin. 2024. MoGU: A framework for enhancing safety of LLMs while preserving their usability. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback.

Lukas Edman, Helmut Schmid, and Alexander Fraser. 2024. CUTE: Measuring LLMs' understanding of their tokens. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Jacob Eisenstein. 2022. Informativeness and invariance: Two perspectives on spurious correlations in natural language. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Chenghao Fan, Zhenyi Lu, Wei Wei, Jie Tian, Xiaoye Qu, Dangyang Chen, and Yu Cheng. 2024. On giant's shoulders: Effortless weak to strong by dynamic logits fusion. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal archive*, 12.

Matthias Gallé. 2019. Investigating the effectiveness of BPE: The power of shorter sequences. 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)*.

Matt Gardner, William Merrill, Jesse Dodge, Matthew Peters, Alexis Ross, Sameer Singh, and Noah A. Smith. 2021. Competency problems: On finding and removing artifacts in language data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Leonidas Gee, Leonardo Rigutini, Marco Ernandes, and Andrea Zugarini. 2023. Multi-word tokenization for sequence compression. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*.

Leonidas Gee, Andrea Zugarini, Leonardo Rigutini, Paolo Torroni, et al. 2022. Fast vocabulary transfer for language model compression. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*.

Robert Geirhos, Jörn-Henrik Jacobsen, Richard Zemel Claudio Michaelis, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. 2020. Shortcut learning in deep neural networks.

Ariel Gera, Roni Friedman, Ofir Arviv, Chulaka Gunasekara, Benjamin Sznajder, Noam Slonim, and Eyal Shnarch. 2023. The benefits of bad advice: Autocontrastive decoding across model layers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. In *Forty-first International Conference on Machine Learning*.

Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. 2024. Unpacking tokenization: Evaluating text compression and its correlation with model performance. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Google. 2024. Gemma: Open models based on gemini research and technology.

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*.

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

Skyler Hallinan, Alisa Liu, Yejin Choi, and Maarten Sap. 2023. Detoxifying text with MaRCo: Controllable revision with experts and anti-experts. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Xiaochuang Han. 2023. In-context alignment: Chat with vanilla language models before fine-tuning.

Xiaochuang Han, Sachin Kumar, Yulia Tsvetkov, and Marjan Ghazvininejad. 2024. David helps Goliath: Inference-time collaboration between small specialized and large general diffusion lms. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xiaochuang Han and Yulia Tsvetkov. 2021. Influence tuning: Demoting spurious correlations via instance attribution and instance-driven updates. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jonathan Hayase, Alisa Liu, Yejin Choi, Sewoong Oh, and Noah A. Smith. 2024. Data mixture inference: What do BPE tokenizers reveal about their training data? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Jonathan Hayase, Alisa Liu, Noah A. Smith, and Sewoong Oh. 2026. Sampling from your language model one byte at a time. In *Forty-third International Conference on Machine Learning*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre is not superb: Derivational morphology improves BERT's interpretation of complex words. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

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. *ArXiv*, abs/1902.00751.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Hongzhi Huang, Defa Zhu, Banggu Wu, Yutao Zeng, Ya Wang, Qiyang Min, and zhou Xun. 2025a. Over-tokenized transformer: Vocabulary is generally worth scaling. In *Forty-second International Conference on Machine Learning*.

James Y. Huang, Wenxuan Zhou, Fei Wang, Fred Morstatter, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2025b. Offset unlearning for large language models. *Transactions on Machine Learning Research*.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. 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)*.

Sukjun Hwang, Brandon Wang, and Albert Gu. 2026. Dynamic chunking for end-to-end hierarchical sequence modeling. In *The Fourteenth International Conference on Learning Representations*.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing LM adaptation with Tulu 2.

Jiabao Ji, Yujian Liu, Yang Zhang, Gaowen Liu, Ramana Rao Kompella, Sijia Liu, and Shiyu Chang. 2024. Reversing the forget-retain objectives: An efficient LLM unlearning framework from logit difference. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Daniel Jurafsky and James H. Martin. 2026. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*, 3rd edition.

Julie Kallini, Shikhar Murty, Christopher D Manning, Christopher Potts, and Róbert Csordás. 2025. Mrt5: Dynamic token merging for efficient byte-level language models. In *The Thirteenth International Conference on Learning Representations*.

Julie Kallini, Artidoro Pagnoni, Tomasz Limisiewicz, Gargi Ghosh, Luke Zettlemoyer, Christopher Potts, Xiaochuang Han, and Srinivasan Iyer. 2026. Fast byte latent transformer. In *Forty-third International Conference on Machine Learning*.

Guy Kaplan, Matanel Oren, Yuval Reif, and Roy Schwartz. 2025. From tokens to words: On the inner lexicon of LLMs. In *The Thirteenth International Conference on Learning Representations*.

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.

Jungo Kasai, Keisuke Sakaguchi, yoichi takahashi, Ronan Le Bras, Akari Asai, Xinyan Velocity Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2023. Realtime QA: What's the answer right now? In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Divyansh Kaushik, Douwe Kiela, Zachary C. Lipton, and Wen-tau Yih. 2021. On the efficacy of adversarial data collection for question answering: Results from a large-scale randomized study. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Pride Kavumba, Naoya Inoue, Benjamin Heinzerling, Keshav Singh, Paul Reisert, and Kentaro Inui. 2019.

When choosing plausible alternatives, clever hans can be clever. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. GeDi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Taku Kudo. 2018. Sentencepiece experiments. `https://github.com/google/sentencepiece/blob/master/doc/experiments.md`.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Dipesh Kumar and Avijit Thawani. 2022. BPE beyond word boundary: How NOT to use multi word expressions in neural machine translation. In *Proceedings of the Third Workshop on Insights from Negative Results in NLP*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7.

Vedang Lad, Wes Gurnee, and Max Tegmark. 2024. The remarkable robustness of llms: Stages of inference?

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih,

Daniel Fried, Sida Wang, and Tao Yu. 2022. Ds-1000: A natural and reliable benchmark for data science code generation.

Yuxuan Lai, Yijia Liu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2021. Lattice-BERT: Leveraging multi-granularity representations in Chinese pre-trained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sander Land. 2024. A short introduction to pre-tokenization weirdness.

Sander Land and Max Bartolo. 2024. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

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. 2021. Mind the gap: Assessing temporal generalization in neural language models. In *Advances in Neural Information Processing Systems*.

Jaechan Lee, Alisa Liu, Orevaoghene Ahia, Hila Gonen, and Noah A. Smith. 2023. That was the last straw, we need more: Are translation systems sensitive to disambiguating context? In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. 2021. Neural data augmentation via example extrapolation.

Mina Lee, Percy Liang, and Qian Yang. 2022. Coauthor: Designing a human-ai collaborative writing dataset for exploring language model capabilities. In *CHI Conference on Human Factors in Computing Systems*.

Sicong Leng, Hang Zhang, Guanzheng Chen, Xin Li, Shijian Lu, Chunyan Miao, and Lidong Bing. 2023. Mitigating object hallucinations in large vision-language models through visual contrastive decoding.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Yinxi Li, Yuntian Deng, and Pengyu Nie. 2025. Tokdrift: When llm speaks in subwords but code speaks in grammar.

Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. Jurassic-1: Technical details and evaluation.

Tomasz Limisiewicz, Artidoro Pagnoni, Srini Iyer, Mike Lewis, Sachin Mehta, Alisa Liu, Margaret Li, Gargi Ghosh, and Luke Zettlemoyer. 2026. Compute optimal tokenization.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. 2023. The unlocking spell on base llms: Rethinking alignment via in-context learning.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024a. Tuning language models by proxy. In *First Conference on Language Modeling*.

Alisa Liu, Jonathan Hayase, Valentin Hofmann, Sewoong Oh, Noah A. Smith, and Yejin Choi. 2025. SuperBPE: Space travel for language models. In *Second Conference on Language Modeling*.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Alisa Liu, Swabha Swayamditta, Noah A. Smith, and Yejin Choi. 2022. WANLI: Worker and AI collaboration for natural language inference dataset creation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Alisa Liu, Zhaofeng Wu, Julian Michael, Alane Suhr, Peter West, Alexander Koller, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2023a. We're afraid language models aren't modeling ambiguity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. 2023b. Same pre-training loss, better downstream: Implicit bias matters for language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*.

Hong Liu, Jiaqi Zhang, Chao Wang, Xing Hu, Linkun Lyu, Jiaqi Sun, Xurui Yang, Bo Wang, Fengcun Li, Yulei Qian, Lingtong Si, Yerui Sun, Rumei Li, Peng Pei, Yuchen Xie, and Xunliang Cai. 2026. Scaling embeddings outperforms scaling experts in language models.

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. 2024b. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. In *First Conference on Language Modeling*.

Yihong Liu, Peiqin Lin, Mingyang Wang, and Hinrich Schütze. 2024c. Ofa: A framework of initializing unseen subword embeddings for efficient large-scale multilingual continued pretraining. In *Findings of the Association for Computational Linguistics: NAACL 2024*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke

Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*.

Ximing Lu, Faeze Brahman, Peter West, Jaehun Jung, Khyathi Chandu, Abhilasha Ravichander, Prithviraj Ammanabrolu, Liwei Jiang, Sahana Ramnath, Nouha Dziri, Jillian Fisher, Bill Lin, Skyler Hallinan, Lianhui Qin, Xiang Ren, Sean Welleck, and Yejin Choi. 2023. Inference-time policy adapters (IPA): Tailoring extreme-scale LMs without fine-tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Scott Lundberg. 2023. The art of prompt design: Prompt boundaries and token healing.

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*.

Kelly Marchisio, Patrick Lewis, Yihong Chen, and Mikel Artetxe. 2023. Mini-model adaptation: Efficiently extending pretrained models to new languages via aligned shallow training. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Haspelmath Martin. 2017. The indeterminacy of word segmentation and the nature of morphology and syntax. *Folia Linguistica*, 51(s1000).

Costas Mavromatis, Petros Karypis, and George Karypis. 2024. Pack of LLMs: Model fusion at test-time via perplexity optimization. In *First Conference on Language Modeling*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*. Academic Press.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, volume 35.

Meta. 2024. The llama 3 herd of models.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Benjamin Minixhofer, Fabian Paischer, and Navid Rekabsaz. 2022. Wechsel: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Benjamin Minixhofer, Edoardo Ponti, and Ivan Vulić. 2024. Zero-shot tokenizer transfer. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Benjamin Minixhofer, Ivan Vulić, and Edoardo Maria Ponti. 2025. Universal cross-tokenizer distillation via approximate likelihood matching. *arXiv preprint arXiv:2503.20083*.

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D Manning. 2024. An emulator for fine-tuning large language models using small language models. In *The Twelfth International Conference on Learning Representations*.

Itay Nakash, Nitay Calderon, Eyal Ben-David, Elad Hoffer, and Roi Reichart. 2025. Adaptivocab: Enhancing LLM efficiency in focused domains through lightweight vocabulary adaptation. In *Second Conference on Language Modeling*.

Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. 2023. Efficient transformers with

dynamic token pooling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2021. Investigating the limitations of transformers with simple arithmetic tasks.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2024. 2 olmo 2 furious.

Yasumasa Onoe, Michael Zhang, Eunsol Choi, and Greg Durrett. 2022. Entity cloze by date: What LMs know about unseen entities. In *Findings of the Association for Computational Linguistics: NAACL 2022*.

OpenAI. 2023. GPT-4 technical report.

OpenAI. 2024. Hello GPT-4o.

Aitor Ormazabal, Mikel Artetxe, and Eneko Agirre. 2023. CombLM: Adapting black-box language models through small fine-tuned models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Naoki Otani, Satoru Ozaki, Xingyuan Zhao, Yucen Li, Micaelah St Johns, and Lori Levin. 2020. Pre-tokenization of multi-word expressions in cross-lingual word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Siru Ouyang, Xinyu Zhu, Zilin Xiao, Minhao Jiang, Yu Meng, and Jiawei Han. 2026. RAST: Reasoning activation in LLMs via small-model transfer. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. 2024. Byte latent transformer: Patches scale better than tokens.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. 2025. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Ellie Pavlick and Tom Kwiatkowski. 2019. Inherent disagreements in human textual inferences. *Transactions of the Association for Computational Linguistics*, 7.

Jonathan Pei, Kevin Yang, and Dan Klein. 2023. PREADD: Prefix-adaptive decoding for controlled text generation. In *Findings of the Association for Computational Linguistics: ACL 2023*.

Buu Phan, Marton Havasi, Matthew Muckley, and Karen Ullrich. 2024. Understanding and mitigating tokenization bias in language models.

Jason Phang, Angelica Chen, William Huang, and Samuel R. Bowman. 2021. Adversarially constructed evaluation sets are more challenging, but may not be fair.

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

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Chen Qian, Jie Zhang, Wei Yao, Dongrui Liu, Zhenfei Yin, Yu Qiao, Yong Liu, and Jing Shao. 2024. Towards tracing trustworthiness dynamics: Revisiting pre-training period of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*.

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

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).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Souvik Rana, Arul Menezes, Ashish Kulkarni, Chandra Khatri, and Shubham Agarwal. 2026. Mutant: A recipe for multilingual tokenizer design.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Marco Tulio Ribeiro. 2023. A guidance language for controlling large language models.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Spring Symposium*.

Alexis Ross, Tongshuang Wu, Hao Peng, Matthew Peters, and Matt Gardner. 2022. Tailor: Generating and perturbing text with semantic controls. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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.

Jessica Rumbelow and Matthew Watkins. 2023. Solidgoldmagikarp (plus, prompt generation).

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9).

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Craig W. Schmidt, Michael Krumdick, Adam Wiemerslage, Seth Ebner, Varshini Reddy, Yuval Pinter, and Chris Tanner. 2026a. Tokenization with split trees.

Craig W Schmidt, Varshini Reddy, Chris Tanner, and Yuval Pinter. 2025. Boundless byte pair encoding: Breaking the pre-tokenization barrier. In *Second Conference on Language Modeling*.

Craig W Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. Tokenization is more than compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Craig W. Schmidt, Chris Tanner, and Yuval Pinter. 2026b. Faster superword tokenization.

Nathan Schneider, Spencer Onuffer, Nora Kazour, Emily Danchik, Michael T. Mordowanec, Henrietta Conrad, and Noah A. Smith. 2014. Comprehensive annotation of multiword expressions in a social web corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Rico Sennrich, Jannis Vamvas, and Alireza Mohammadshahi. 2023. Mitigating hallucinations and off-target machine translation with source-contrastive and language-contrastive decoding.

Ruizhe Shi, Yifang Chen, Yushi Hu, Alisa Liu, Hannaneh Hajishirzi, Noah A. Smith, and Simon Shaolei Du. 2024. Decoding-time language model alignment with multiple objectives. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Scott Wen tau Yih. 2023. Trusting your evidence: Hallucinate less with context-aware decoding.

Anya Sims, Cong Lu, Klara Kaleb, Jakob Nicolaus Foerster, and Yee Whye Teh. 2025. Stochastok: Improving fine-grained subword understanding in LLMs. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Aaditya K. Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier llms.

Neha Srikanth and Rachel Rudinger. 2022. Partial-input baselines show that NLI models can ignore context, but they don't. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, and Adrià Garriga-Alonso et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33.

Saku Sugawara, Pontus Stenetorp, Kentaro Inui, and Akiko Aizawa. 2020. Assessing the benchmarking capacity of machine reading comprehension datasets. In *AAAI Conference on Artificial Intelligence*.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Alon Talmor, Ori Yoran, Ronan Le Bras, Chandra Bhagavatula, Yoav Goldberg, Yejin Choi, and Jonathan Berant. 2021. CommonsenseQA 2.0: Exposing the limits of AI through gamification. In *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. 2024. Scaling laws with vocabulary: Larger models deserve larger vocabularies. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2022. Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations*.

Serra Sinem Tekiroğlu, Yi-Ling Chung, and Marco Guerini. 2020. Generating counter narratives against online hate speech: Data and strategies. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Jan Tempus, Philip Whittington, Craig W. Schmidt, Dennis Komm, and Tiago Pimentel. 2026. Tokenisation via convex relaxations.

Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro Szekely. 2021. Representing numbers in NLP: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh

Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ke Tran. 2020. From english to foreign languages: Transferring pre-trained language models. *arXiv preprint arXiv:2002.07306*.

Albert Tseng and Christopher De Sa. 2026. L^3: Large lookup layers.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. 2020. An empirical study on robustness to spurious correlations using pre-trained language models. *Transactions of the Association for Computational Linguistics*, 8.

Andrei-Valentin Tănase and Elena Pelican. 2025. Supratok: Cross-boundary tokenization for enhanced language model performance.

Clara Vania, Ruijie Chen, and Samuel R. Bowman. 2020. Asking Crowdworkers to Write Entailment Examples: The Best of Bad options. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*.

Menan Velayuthan and Kengatharaiyer Sarveswaran. 2025. Egalitarian language representation in language models: It all begins with tokenizers. In *Proceedings of the 31st International Conference on Computational Linguistics*.

Bandhav Veluri, Justin Chan, Malek Itani, Tuochao Chen, Takuya Yoshioka, and Shyamnath Gollakota. 2023. Real-time target sound extraction. In *ICASSP*.

Tim Vieira, Ben LeBrun, Mario Giulianelli, Juan Luis Gastaldi, Brian DuSell, John Terilla, Timothy J O'Donnell, and Ryan Cotterell. 2024. From language models over tokens to language models over characters. *arXiv preprint arXiv:2412.03719*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Junxiong Wang, Tushaar Gangavarapu, Jing Nathan Yan, and Alexander M Rush. 2024. Mambabyte: Token-free selective state space model. In *First Conference on Language Modeling*.

Peter West, Chandra Bhagavatula, Jack Hessel, Jena D. Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2021. Symbolic knowledge distillation: from general language models to commonsense models. *arXiv*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. 2023a. Pmc-llama: Towards building open-source language models for medicine.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023b. Bloomberggpt: A large language model for finance.

Yuxiang Wu, Matt Gardner, Pontus Stenetorp, and Pradeep Dasigi. 2022. Generating data to mitigate spurious correlations in natural language inference datasets. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Zilin Xiao, Jaywon Koo, Siru Ouyang, Jefferson Hernandez, Yu Meng, and Vicente Ordonez. 2026. Proxy-thinker: Test-time guidance through small visual reasoners. In *The Fourteenth International Conference on Learning Representations*.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yebin Yang, Huaijin Wu, Fu Guo, Lin Yao, Xiaohan Qin, Jingzhi Wang, Debing Zhang, and Junchi Yan. 2026. Jtok: On token embedding as another axis of scaling law via joint token self-modulation.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Lili Yu, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. MEGABYTE: Predicting million-byte sequences with multiscale transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Ann Yuan, Daphne Ippolito, Vitaly Nikolaev, Chris Callison-Burch, Andy Coenen, and Sebastian Gehrmann. 2021. Synthbio: A case study in human-ai collaborative curation of text datasets. In *Neural Information Processing Systems Track on Datasets and Benchmarks*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Kaiyan Zhang, Jianyu Wang, Ermo Hua, Biqing Qi, Ning Ding, and Bowen Zhou. 2024. CoGenesis: A framework collaborating large and small language models for secure context-aware instruction following. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Xinsong Zhang, Pengshuai Li, and Hang Li. 2021. AMBERT: A pre-trained language model with multi-grained tokenization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*.

Yunxiang Zhang, Muhammad Khalifa, Lechen Zhang, Xin Liu, Ayoung Lee, Xinliang Frederick Zhang, Farima Fatahi Bayat, and Lu Wang. 2026. Logit arithmetic elicits long reasoning capabilities without training. In *Findings of the Association for Computational Linguistics: ACL 2026*.

Xuandong Zhao, Xianjun Yang, Tianyu Pang, Chao Du, Lei Li, Yu-Xiang Wang, and William Yang Wang. 2025. Weak-to-strong jailbreaking on large language models. In *Forty-second International Conference on Machine Learning*.

Zhenyu Zhao, Sander Land, Daniel M. Bikel, and Waseem Alshikh. 2026. Shorthand for thought: Compressing llm reasoning via entropy-guided supertokens.

Brian Siyuan Zheng, Alisa Liu, Orevaoghene Ahia, Jonathan Hayase, Yejin Choi, and Noah A. Smith. 2025. Broken tokens? your language model can secretly handle non-canonical tokenizations. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Yining Chen, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. 2022. Analytical reasoning of text. In *Findings of the Association for Computational Linguistics: NAACL 2022*.

Zhanhui Zhou, Jie Liu, Zhichen Dong, Jiaheng Liu, Chao Yang, Wanli Ouyang, and Yu Qiao. 2024. Emulated disalignment: Safety alignment for large language models may backfire! In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Chapter A

Evaluation Datasets

In this section, we provide a description of datasets used for evaluation in Chapter 3 and Chapter 4. All evaluations are based on decoding from the model and scoring the generation by either comparing it to the ground truth or evaluating its functional correctness (in the case of coding tasks). For multiple choice (MC) tasks, we check whether the predicted answer choice is an exact match (EM) to the target (we observe that effectively 100% of model generations are valid answer choices, especially for later checkpoints). For open-ended tasks, we check whether the generated output contains the ground truth answer exactly, and for coding tasks, we report pass@10. BB below stands for BIG-Bench.

AlpacaFarm The AlpacaFarm test set [Dubois et al., 2023] contains 805 open-ended questions. We use the `alpaca_eval` library[1], with `alpaca_eval_gpt4_0314` as the evaluator. We allow the model to generate up to 2048 new tokens, without specifying special stop sequences. Responses are evaluated based on the win-rate against corresponding responses from `text-davinci-003`, as determined by GPT-4.

ARC consists of 4-way MC questions from grades 3–9 science exams. It contains two splits, ARC-Easy, which require knowledge of basic science, and ARC-Challenge, which require some procedural reasoning [Clark et al., 2018].

[1] https://github.com/tatsu-lab/alpaca_eval

Arithmetic contains simple arithmetic problems [Brown et al., 2020].[2] We use the `2da`, `2dm`, and `2ds` splits for addition, multiplication, and division of (up to) 2-digit numbers.

BoolQ contains naturally occurring yes/no questions paired with passages that provide an answer [Clark et al., 2019].

CommonsenseQA contains 5-way MC questions that require commonsense knowledge to answer [Talmor et al., 2019].

COPA contains two-way MC questions about cause and effect [Roemmele et al., 2011; Kavumba et al., 2019].

CoQA consists of passages with a series of conversational questions about the passage Reddy et al. [2019]. Each question requires the prior conversational context, due to possible coreference across questions. Because these contextual questions naturally serve as in-context examples, we do not provide additional in-context examples.

BB CS Algorithms consists of two subtasks, determining whether a given series of parentheses is balanced and identifying the longest common subsequence in two letter strings [BIG-bench, 2023].

CUTE contains questions that require the model to understand and manipulate spelling, such as replacing all instances of a particular letter in a word with another letter [Edman et al., 2024].

DROP contains questions about passages, potentially requiring reasoning over multiple pieces of information in the passage [Dua et al., 2019].

BB Dyck Languages consists of a sequence of parentheses and requires the model to predict the correct sequence of closing parentheses so that the entire sequence is well-balanced.

[2] https://huggingface.co/datasets/EleutherAI/arithmetic

GSM8K contains grade school math word problems that require between 2 and 8 steps to solve. In the in-context examples, we provide the answer passage that contains intermediate steps with calculator annotations removed. The model is expected to provide the final numerical answer after four hashtags (####) that delimit the reasoning and final answer [Cobbe et al., 2021].

HellaSwag contains 4-way MC questions which ask for the most natural continuation given the context [Zellers et al., 2019].

HotpotQA contains questions along with a corresponding passage from Wikipedia containing the answer [Yang et al., 2018].

HumanEval contains programming problems where the model is tasked with completing a Python function given its docstring [Chen et al., 2021a]. We use "`\nclass`," "`\ndef`," "`\n#`," "`\nif`," as stop tokens. Following the original paper, we sample 20 continuations with top $p = 0.95$ and temperature = 0.8. Models are allowed to generate for a maximum of 128 new tokens. The functional correctness of generations is automatically evaluated using test cases. We use the 20 generation to make an unbiased estimate of the pass@10 rate, i.e., how likely at least one of 10 sampled solutions for a problem is correct.

Jeopardy contains open-ended questions from the "Jeopardy!" quiz show.[3]

Lambada contains narratives without the last word, which is inferrable given the context [Paperno et al., 2016]. This task requires models to attend to the full narrative instead of only the local context.

BB Language Identification contains sentences in different languages, and the task is to choose the language of the sentence from a long list of options.

LSAT-AR contains MC questions that evaluate the analytical reasoning (AR) ability of LMs [Zhong et al., 2022, 2024]. Test questions are drawn from the Law School Admission Test (LSAT) from 1991 to 2016.

[3] https://www.kaggle.com/datasets/tunguz/200000-jeopardy-questions

MBPP contains Python programming problems where the model is given a description of the desired function and a series of unit tests. We use the same evaluation setup as HumanEval.

MMLU contains 4-way MC questions covering 57 different domains, covering both world knowledge and problem-solving abilities [Hendrycks et al., 2021]. Note that we report a straight average over the 5000-example sample, rather than a macro-average over subjects.

OpenbookQA contains 4-way MC questions that require multi-step reasoning and commonsense knowledge [Mihaylov et al., 2018].

BB Operators contains questions where the model is given a function definition and asked to compute the output of that function given a particular input.

PIQA contains MC questions that require physical commonsense reasoning [Bisk et al., 2020].

BB Repeat-Copy-Logic contains instructions that ask the model to produce a particular string [Austin et al., 2021].

SQuAD contains passages paired with questions about the passage [Rajpurkar et al., 2016]. The answer is always a span from the passage.

ToxiGen We follow the evaluation set-up of Llama 2, which prompts the model with a sequence of hateful sentences targeting a certain demographic group from the ToxiGen dataset [Hartvigsen et al., 2022]. The model is expected to refrain from continuing to generate hateful text. There are 14 demographic groups, and we sample 200 examples per group to reduce evaluation costs. We allow the model to generate up to 512 new tokens, and also use newline as a stop token (as each hateful statement is on a new line). We use the toxicity classifier based on `roberta-large` from Hartvigsen et al. [2022] to score the generation toxicity. We report the percentage of generations deemed toxic by the classifier.

TriviaQA contains open-ended questions about world knowledge [Joshi et al., 2017].

TruthfulQA TruthfulQA [Lin et al., 2022] is a dataset of 817 often misleading questions, designed to test whether LMs are susceptible to common misconceptions. For both the open-ended and MC setting, we provide the system prompt used in the original LLAMA2 paper to any LLAMA2-CHAT model (whether it's being evaluated on its own or part of an ensemble), as it clarifies the desired behavior and dramatically improves performance for CHAT models.

For **open-ended question-answering**, we use two trained GPT-3-based classifiers from Tülu evaluation to judge the truthfulness and informativeness of model responses. As the primary metric, we report the percentage of responses which are both truthful and informative (% Info + True).

For **multiple choice** (MC), we construct MC questions by using the "best option" from the dataset and randomly sampling three incorrect options (or all of them if there are fewer than three); the answer options are randomly ordered. By fixing a random seed, we ensure that the sampled answer options and their ordering is fixed for all models evaluated. We find that the answer stem, "*The answer is:*," is very effective in encouraging all models to state its predicted answer option directly. Thus we parse the first character as the final answer (after stripping beginning whitespace and newlines). For LLAMA2-CHAT and proxy-tuned models, only 0-1 generations (out of 817) cannot be parsed as a valid MC option.

BB WikidataQA require models to complete factual statements with the correct continuation.

Winograd contains binary MC questions where the model is given a context and asked to determine which entity a pronoun refers to, between two options [Levesque et al., 2012]. Correctly answer the question requires commonsense knowledge and contextual reasoning.

Winogrande contain questions with the same schema as Winograd, but increases both the scale and difficulty of the dataset [Sakaguchi et al., 2021].