UNIVERSITY OF CALIFORNIA SAN DIEGO

Training Data Curation for Language Models with Weak Supervision

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Dheeraj Mekala

Committee in charge:

Professor Jingbo Shang, Chair
Professor Taylor Berg-Kirkpatrick
Professor Zhiting Hu
Professor Julian McAuley

2025

University of California San Diego

2025

DEDICATION

To my mom, dad, & beloved wife for their love and support. To Mother Nature, whose beauty captivates me and fuels my curiosity to understand and unravel its wonders.

EPIGRAPH

The simplest explanation is usually the correct one.

William of Ockham

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Jingbo Shang for his support as my advisor and the chair of my committee. His expertise in the domain and insightful discussions about the research were instrumental in completing this thesis. I am also immensely thankful to him for his support, flexibility in allowing me to pursue my interests and kindness during the challenging times of the pandemic. I am deeply grateful to him for providing an enriching environment for the past five years which has helped me gain valuable research experience. His advice to *keep it simple* will always stay with me.

I also extend my sincere thanks to Prof. Taylor Berg-Kirkpatrick, Prof. Julian McAuley, and Prof. Zhiting Hu for serving on my thesis committee. Their insightful advice and thoughtful feedback have been essential in refining my thesis and preparing me for a successful defense.

I am sincerely thankful to my mentors and collaborators from Amazon Product Graph, Microsoft Semantic Machines, the Meta FAIR team, and the Meta GenAI Llama Agents team – particularly Nasser Zalmout, Chenwei Zhang, Xian Li, Subhro Roy, Jane Yu, Jason Weston, Roberta Raileanu, Jack Lanchantin, Maria Lomeli, Grégoire Mialon, Virginie Do, Thomas Scialom, Amine Benhalloum, Romain Froger, Mathieu Rita, Pierre Ménard, Tatiana Shavrina. I had the privilege of interning with these incredible teams, where I gained valuable insights that significantly contributed to my research.

I would like to express my heartfelt appreciation to my labmates at the Shang Data Lab, especially Chengyu, Zihan, Zilong, Zi, Weitang, Alex, and Feng, for numerous brainstorming sessions that sparked many ideas and helped me explore several new directions in my research. I am also grateful to my fellow graduate students, Amanda, Erik, Harsh Jhamtani, and Eric, whose company I thoroughly enjoyed.

I would like to express my sincere gratitude to IIT Kanpur for providing me with an exceptional environment to grow both academically and personally. During my undergraduate there, I gained valuable knowledge in Computer Science that has been crucial to my development as a researcher. Additionally, I would like to express my gratitude to Professor Harish Karnick

(CSE, IIT Kanpur) who kindled my interest in the fields of Machine learning and Natural Language Processing, and my first collaborator Vivek Gupta, for motivating me to work on important problems. Beyond academics, IIT Kanpur also gifted me lifelong friendships – Manpreet, Varun, Yash, Shiva, Himanshu, Vinod, Ajay, Harsh, Shivam, Akhil, Keerti, and many more, and helped me develop essential life skills that continue to shape my journey. I am truly grateful for the experiences and opportunities that have had such a profound impact on my life.

I am eternally grateful to my school, SPR School of Excellence, for fostering my curiosity and never discouraging a relentless student like me who constantly asked questions. This encouragement instilled a lifelong sense of curiosity, which ultimately led me to explore important questions during my Ph.D. It was here that I truly learned *how to learn*.

I would like to express my heartfelt gratitude to my parents, grandparents, and uncle for their unwavering support throughout my life. They have consistently motivated me to aim high and never settle for less. Whenever I have been tempted to aim lower, they have acted as catalysts, reminding me that I am capable of more. In a sense, they've always encouraged me to reach for greater heights, providing a safety net to catch me if I fall.

I am grateful to my friends Palash, Amulya, Shreyas, Gautham, Winny, Joseph, Goutham, Varshita, Prashanti, Sai Kishan, Keerti Anand, Akhil Vemula, Rachita, Aodhagan, Sahana, Devashish, Amit, Balu, Mengqian for their friendship. You have been a constant presence, providing enjoyable moments and support when necessary.

Finally, I am forever thankful to my amazing wife, Sanjana, for her endless love, support, and patience. You've been my rock, offering encouragement and strength through every challenge and triumph. Your unwavering belief in me has been a constant source of motivation, and I'm so thankful to have you by my side. Thank you for being my partner, my best friend, my sounding board, and my greatest supporter throughout this journey.

Chapter 1, in full, is a reprint of the material as it appears in Mekala, Dheeraj; Shang, Jingbo. "Contextualized weak supervision for text classification," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 323–333, 2020. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 2, in full, is a reprint of the material as it appears in Mekala, Dheeraj; Zhang, Xinyang; Shang, Jingbo. “Meta: Metadata-empowered weak supervision for text classification,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 8351–8361, 2020. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in Dheeraj Mekala, Chengyu Dong, and Jingbo Shang. 2022. "LOPS: Learning Order Inspired Pseudo-Label Selection for Weakly Supervised Text Classification", in Findings of the Association for Computational Linguistics: EMNLP 2022, pages 4894–4908, Association for Computational Linguistics. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 4, in full, is a reprint of the material as it appears in Dheeraj Mekala, Tu Vu, Timo Schick, and Jingbo Shang. 2022. Leveraging QA Datasets to Improve Generative Data Augmentation, in Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 9737–9750, Association for Computational Linguistics. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 5, in full, is a reprint of the material as it appears in Dheeraj Mekala, Jason E Weston, Jack Lanchantin, Roberta Raileanu, Maria Lomeli, Jingbo Shang, and Jane Dwivedi-Yu. 2024. TOOLVERIFIER: Generalization to New Tools via Self-Verification, in Findings of the Association for Computational Linguistics: EMNLP 2024, pages 5026–5041, Association for Computational Linguistics. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 6, in full, is a reprint of the material as it appears in Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. 2024. Smaller Language Models are capable of selecting Instruction-Tuning Training Data for Larger Language Models, in Findings of the Association for Computational Linguistics: ACL 2024, pages 10456–10470, Association for Computational Linguistics. The dissertation/thesis author was the primary investigator and author of this paper.

VITA

2017	Bachelor of Technology in Computer Science and Engineering, Indian Institute of Technology Kanpur
2021	Master of Science in Computer Science, University of California San Diego
2025	Doctor of Philosophy in Computer Science, University of California San Diego

PUBLICATIONS

[1] Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. Smaller Language Models are capable of selecting Instruction-Tuning Training Data for Larger Language Models. ACL 2024.

[2] Dheeraj Mekala, Jason E Weston, Jack Lanchantin, Roberta Raileanu, Maria Lomeli, Jingbo Shang, and Jane Dwivedi-Yu. TOOLVERIFIER: Generalization to New Tools via Self-Verification. EMNLP 2024.

[3] Dheeraj Mekala, Chengyu Dong, and Jingbo Shang. LOPS: Learning Order Inspired Pseudo-Label Selection for Weakly Supervised Text Classification. EMNLP 2022.

[4] Dheeraj Mekala, Tu Vu, Timo Schick, and Jingbo Shang. Leveraging QA Datasets to Improve Generative Data Augmentation. EMNLP 2022.

[5] Alex Nguyen, Dheeraj Mekala, Chengyu Dong, Jingbo Shang. When is the Consistent Prediction Likely to Be A Correct Prediction?

[6] Dheeraj Mekala, Xinyang Zhang, and Jingbo Shang. META: Metadata-Empowered Weak Supervision for Text Classification. EMNLP 2020.

[7] Dheeraj Mekala and Jingbo Shang. Contextualized Weak Supervision for Text Classification. ACL 2020.

[8] Dheeraj Mekala, Jason Wolfe, and Subhro Roy. ZEROTOP: Zero-Shot Task-Oriented Semantic Parsing using Large Language Models. EMNLP 2023.

[9] Dheeraj Mekala, Varun Gangal, and Jingbo Shang. Coarse2Fine: Fine-grained Text Classification on Coarsely-grained Annotated Data. EMNLP 2021.

[10] Dheeraj Mekala, Vivek Gupta, Bhargavi Paranjape, and Harish Karnick. SCDV : Sparse Composite Document Vectors using soft clustering over distributional representations. EMNLP 2017 (Oral).

[11] Zihan Wang, Tianle Wang, Dheeraj Mekala, and Jingbo Shang. A Benchmark on Extremely Weakly Supervised Text Classification: Reconcile Seed Matching and Prompting Approaches. ACL 2023.

[12] Zihan Wang, Dheeraj Mekala, and Jingbo Shang. X-Class: Text Classification with Extremely Weak Supervision. NAACL 2021.

[13] Yasaman Jafari, Dheeraj Mekala, Rose Yu, and Taylor Berg-Kirkpatrick. MORL-Prompt: An Empirical Analysis of Multi-Objective Reinforcement Learning for Discrete Prompt Optimization. EMNLP 2024.

[14] Alex Nguyen, Zilong Wang, Jingbo Shang, and Dheeraj Mekala. DOCMASTER: A Unified Platform for Annotation, Training, & Inference in Document Question-Answering. NAACL 2024.

[15] Zichao Li, Dheeraj Mekala, Chengyu Dong, and Jingbo Shang. BFClass: A Backdoor-free Text Classification Framework. EMNLP 2021.

[16] Dheeraj Mekala, Adithya Samavedhi, Chengyu Dong, and Jingbo Shang. SELFOOD: Self-Supervised Out-Of-Distribution Detection via Learning to Rank. EMNLP 2023.

[17] Xiuwen Zheng, Dheeraj Mekala, Amarnath Gupta, Jingbo Shang. News Meets Microblog: A Retriever-Generator Hashtag Annotation Framework.

[18] Dawei Li, Yaxuan Li, Dheeraj Mekala, Shuyao Li, Xueqi Wang, William Hogan, Jingbo Shang. DAIL: Data Augmentation for In-Context Learning via Self-Paraphrase

[19] Sudhanshu Ranjan, Dheeraj Mekala, and Jingbo Shang. Progressive Sentiment Analysis for Code-Switched Text Data. EMNLP 2022.

[20] Rahul Wadbude, Vivek Gupta, Dheeraj Mekala, Harish Karnick. User bias removal in review score prediction. CODS/COMAD 2018.

ABSTRACT OF THE DISSERTATION

Training Data Curation for Language Models with Weak Supervision

by

Dheeraj Mekala

Doctor of Philosophy in Computer Science

University of California San Diego, 2025

Professor Jingbo Shang, Chair

Large language models (LLMs) are typically trained on millions of annotated samples, incurring substantial annotation costs and computational resources. While neural scaling laws demonstrate that test error decreases as a power law with training data volume, we are approaching the limits of feasibly collectible public data. This thesis investigates efficient alternatives through weak supervision. We explore two kinds of weak supervision: extractive and generative. Extractive weak supervision curate training data from unsupervised pools of data using weak supervision sources, while generative weak supervision leverages pre-trained language models to create synthetic data. We also propose assessing data quality through three key dimensions: diversity, difficulty, and correctness. Additionally, we empirically show that the training dynamics of LLMs provide valuable insights into data quality.

In the extractive weak supervision domain, we present a contextualized weakly supervised text classification framework that utilizes contextualized representations and user-provided seed words to interpret the corpus and derive labeled data. We also demonstrate that metadata can serve as an additional supervision source in our metadata-empowered weakly supervised classification framework.

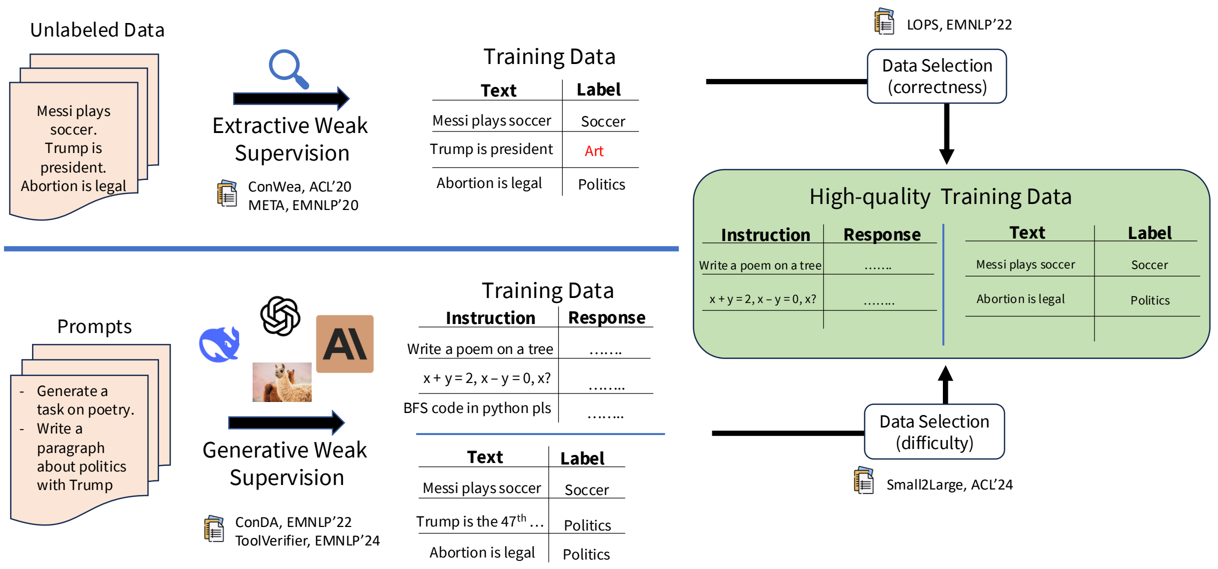
For generative weak supervision, we showcase how language models and publicly-available question-answering datasets can be leveraged to generate text-classification data. Additionally, we also discuss our work on generating synthetic tool-usage data from scratch with minimal human supervision.

Finally, we analyze training dynamics to understand the data quality and investigate whether quality improvements can reduce quantity requirements. Through the lens of difficulty, diversity, and noise, we observe that extractive weak supervision tends to produce noisy data while generative weak supervision creates less challenging data. To address these limitations, we propose learning-order-based selection to filter noisy data and learning-percentage-based selection to identify difficult examples. We also empirically observe that smaller language models are capable of curating training data for larger language models.

Together, this body of work advances our understanding of what constitutes "high-quality" training data while providing cost-effective solutions for data curation. Our theoretical analyses and empirical evaluations demonstrate significant performance improvements achieved through these weak supervision approaches and targeted data selection methods. These approaches not only reduce resource requirements but also establish a framework for quantifying data quality metrics across different weak supervision paradigms. By strategically addressing the inherent limitations of both extractive and generative weak supervision, we provide a comprehensive methodology for optimizing training data that balances quality considerations with practical constraints, ultimately creating more efficient pathways for training robust language models.

Introduction

Large language models (LLMs) have transformed the artificial intelligence landscape with their remarkable capabilities. At the heart of these advancements lies a critical factor: training data. These models achieve their prowess through exposure to vast quantities of annotated examples — for instance, Llama-3 (Dubey et al., 2024) was instruction-tuned on more than 10 million manually annotated data points. This massive data requirement represents a significant bottleneck in the development pipeline.



Unlabeled Data
Messi plays soccer. Trump is president. Abortion is legal
Extractive Weak Supervision
ConWea, ACL'20
META, EMNLP'20
Training Data
Text | Label
Messi plays soccer | Soccer
Trump is president | Art
Abortion is legal | Politics
Prompts
- Generate a task on poetry.
- Write a paragraph about politics with Trump
Generative Weak Supervision
ConDA, EMNLP'22
ToolVerifier, EMNLP'24
Training Data
Instruction | Response
Write a poem on a tree | …….
x + y = 2, x – y = 0, x? | ……..
BFS code in python pls | ……..
Text | Label
Messi plays soccer | Soccer
Trump is the 47th … | Politics
Abortion is legal | Politics
LOPS, EMNLP'22
Data Selection (correctness)
High-quality Training Data
Instruction | Response
Write a poem on a tree | …….
x + y = 2, x – y = 0, x? | ……..
Text | Label
Messi plays soccer | Soccer
Abortion is legal | Politics
Data Selection (difficulty)
Small2Large, ACL'24

Figure 1. Overview of the dissertation. We introduce data curation methods leveraging extractive and generative weak supervision. Moreover, we also propose data selection works focused on correctness and difficulty of the data.

Neural scaling laws (Kaplan et al., 2020; Henighan et al., 2020; Gordon et al., 2021) empirically demonstrate that test error decreases as a power law relative to training data volume.

However, as we approach the limits of feasibly collectible public data, exploring efficient alternatives becomes imperative. This dissertation investigates how weak supervision techniques can address this challenge by enabling the curation of high-quality training data with minimal human intervention. We introduce cost-effective data curation methods by efficiently leveraging different forms of weak supervision.

We explore two fundamental types of weak supervision: extractive and generative, as illustrated in Figure 1. Extractive weak supervision involves curating training data from unsupervised sources using limited supervision signals such as seed words (Meng et al., 2018, 2019), label names (Meng et al., 2020; Wang et al., 2021), label descriptions (Mekala et al., 2021), and few labeled samples (Mekala et al., 2022b). This approach is particularly useful for fine-tuning LLMs on discriminative tasks. In contrast, generative weak supervision employs pre-trained LLMs (Radford & Narasimhan, 2018; Radford et al., 2019; Brown et al., 2020; Dubey et al., 2024) to synthesize training data (Kumar et al., 2020; Anaby-Tavor et al., 2020), making it applicable to both discriminative tasks like text classification and generative tasks such as instruction-following (Wang et al., 2022a). However, both extractive and generative weak supervision often produce low-quality training data. To address this limitation, we investigate what constitutes "high-quality" data through the lens of model training dynamics. Our analysis examines three critical dimensions: diversity, difficulty, and correctness. We observe that extractive weak supervision tends to result in noisy data with low correctness, while generative weak supervision often produces simplistic data with low difficulty (Peng et al., 2024). To enhance model performance, we propose two data selection methods for improving the data quality tailored to extractive and generative weak supervision, independently optimizing for correctness and difficulty. A comparative analysis of these approaches is presented in Table 1.

A common source of extractive weak supervision includes small sets of user-provided label-indicative seed words for each class. However, the immense volume of text data and the inherent complexity of natural language create significant challenges. Seed words often have multiple interpretations depending on context, requiring identification of user-intended meanings

Table 1. Extractive vs Generative Weak Supervision comparison

	Extractive Weak Supervision	**Generative Weak Supervision**
Unlabeled data	Necessary	No
LLMs for generation	No	Necessary
Tasks	Mainly discriminative	Both discriminative & generative
Data Selection Criteria	Correctness	Difficulty

for accurate classification. Traditional methods generate pseudo-labels through context-free approaches (such as string matching), overlooking the ambiguous, context-dependent nature of human language. In our first work, we explore contextualized weak supervision where we leverage contextualized representations of word occurrences and seed word information to automatically differentiate multiple interpretations of a word. The "contextualized" aspect manifests in two ways: in the corpus, where each word occurrence may be interpreted differently according to its context; and in seed words, where ambiguous terms must be resolved according to their user-specified class. This approach aims to improve the accuracy of the final text classifier. Additionally, widely available metadata—including author information, publication date, and geographic location—represents an untapped, complementary supervision source. Our second work organizes text and metadata into a text-rich network and employs network motifs to capture appropriate metadata combinations. This network structure provides a holistic view of the corpus and enables ranking and selection of useful metadata entities.

Generative weak supervision typically involves priming an LLM with a task description and relevant contextual information to generate corresponding data. We present work that reformulates the data generation task as a context generation task for given question-answer pairs, utilizing publicly-available question-answering datasets to train a data generator. With this generator, we create text classification datasets that significantly improve classifier performance. Additionally, we present our work on generating tool-usage data for training tool-calling LLMs from scratch with minimal human input. Starting with weak human supervision signals—consisting only of domain specifications and a few examples—we employ a hierarchical approach to synthetically generate a comprehensive suite of tools, user instructions, and the underlying reasoning processes that guide tool selection.

The data obtained through either extractive or generative weak supervision is rarely perfect. To overcome these limitations, we introduce our two training dynamics-inspired data selection methods. These approaches filter out noisy and less challenging samples based on a key observation: models typically learn clean, easy samples first, followed by difficult samples, with noisy samples processed last. Our first method introduces learning-order based data selection for discriminative tasks, empirically demonstrating that learning order reflects the probability of wrong annotation, thereby eliminating noise by selecting samples learned earlier. We generalize this approach to generative tasks with learning-percentage based selection, eliminating less challenging data for instruction-following tasks, which improves training time without compromising performance.

We explore extractive weak supervision in Chapters 1, 2, 3 and generative weak supervision in Chapters 4, 5, 6. Chapter 1 introduces contextualized weak supervision (Mekala & Shang, 2020), while Chapter 2 examines the use of metadata as auxiliary weak supervision (Mekala et al., 2020). Chapter 3 introduces the learning-order inspired data selection method focused on correctness tailored for extractive weak supervision (Mekala et al., 2022a). Our works leveraging generative weak supervision begins with Chapter 4 where we explore utilizing QA datasets to train a data generator for text classification tasks (Mekala et al., 2022b). Chapter 5 investigates generating tool-selection data to enhance tool generalization (Mekala et al., 2024b). Finally, Chapter 6 presents the learning-percentage based data selection strategy centered on difficulty of the data tailored for generative weak supervision (Mekala et al., 2024a).

The ultimate aim of our research is to develop practical data curation methods that enable the creation of high-quality training datasets with minimal human effort (Nguyen et al., 2024), thereby advancing the capabilities and accessibility of large language models.

Open source tools

Our developed methods are made public and has attracted attention from the open-source community. The list of our open-sourced methods are as follows:

- ConWea: `https://github.com/dheeraj7596/ConWea`
- META: `https://github.com/dheeraj7596/META`
- LOPS: `https://github.com/dheeraj7596/LOPS`
- Conda: `https://github.com/dheeraj7596/CONDA`
- ToolVerifier:
  - Code: `https://github.com/facebookresearch/ToolVerifier`
  - Third-party blogs:
    - MarktechPost: Researchers from Meta AI and UCSD Present ToolVerifier
    - DailyAI: Meta, UCSD Introduce ToolVerifier
    - FavTutor: Meta UCSD ToolVerifier LLM Tool Calls
  - Dataset: `https://huggingface.co/datasets/facebook/toolverifier`
- Small2Large: `https://github.com/dheeraj7596/Small2Large`

Overall Impact

Our methods on contextualized weak supervision for text classification (ConWea (Mekala & Shang, 2020)), metadata-empowered weak supervision for text classification (META (Mekala et al., 2020)), and learning-order based pseudo-label selection (LOPS (Mekala et al., 2022a)) are being taught in graduate courses, e.g. *University of California San Diego* (CSE291-Advanced Data-Driven Text Mining). Our insights from Chapter 6 (Mekala et al., 2024a) and Chapter 5 (Mekala et al., 2024b) are shared & cited in the influential Llama-3 paper (Dubey et al., 2024).

Chapter 1

Contextualized Weak Supervision for Text Classification

In this chapter, we present our proposed method ConWea, that contextualizes weak supervision thereby resolving the interpretation of seed words and performs text classification.

1.1 Motivation & Overview

Weak supervision in text classification has recently attracted much attention from researchers, because it alleviates the burden of human experts on annotating massive documents. One of the popular forms of weak supervision is a small set of user-provided seed words for each class. Typical seed-driven methods follow an iterative framework — generate pseudo-labels using some heuristics, learn the mapping between documents and classes, and expand the seed set (Agichtein & Gravano, 2000; Riloff et al., 2003; Kuipers et al., 2006; Tao et al., 2015; Meng et al., 2018).

Most of, if not all, existing methods generate pseudo-labels in a context-free manner, therefore, the ambiguous, context-dependent nature of human languages has been long overlooked. Suppose the user gives "penalty" as a seed word for the *sports* class, as shown in Figure 1.1. The word "penalty" has at least two different meanings: the penalty in *sports*-related documents and the fine or death penalty in *law*-related documents. If the pseudo-label of a document is decided based only on the frequency of seed words, some documents about *law* may

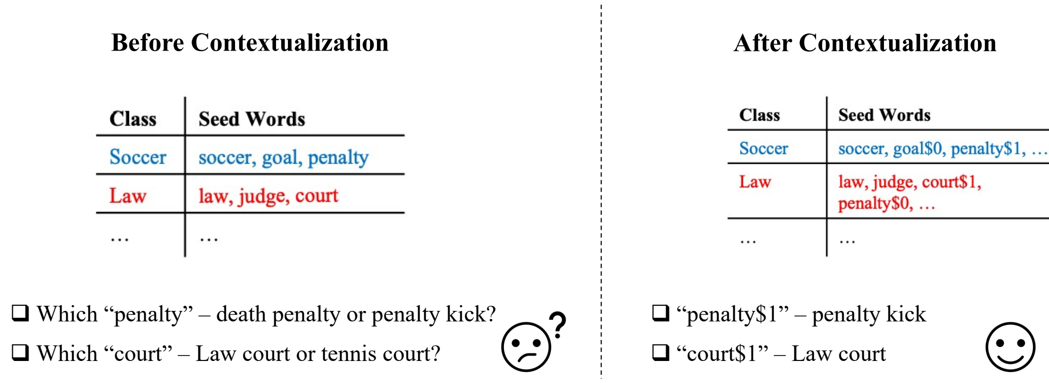

Before Contextualization
Class | Seed Words
Soccer | soccer, goal, penalty
Law | law, judge, court
… | …
Which "penalty" – death penalty or penalty kick?
Which "court" – Law court or tennis court?
?
After Contextualization
Class | Seed Words
Soccer | soccer, goal$0, penalty$1, …
Law | law, judge, court$1, penalty$0, …
… | …
"penalty$1" – penalty kick
"court$1" – Law court

Figure 1.1. Why contextualization?

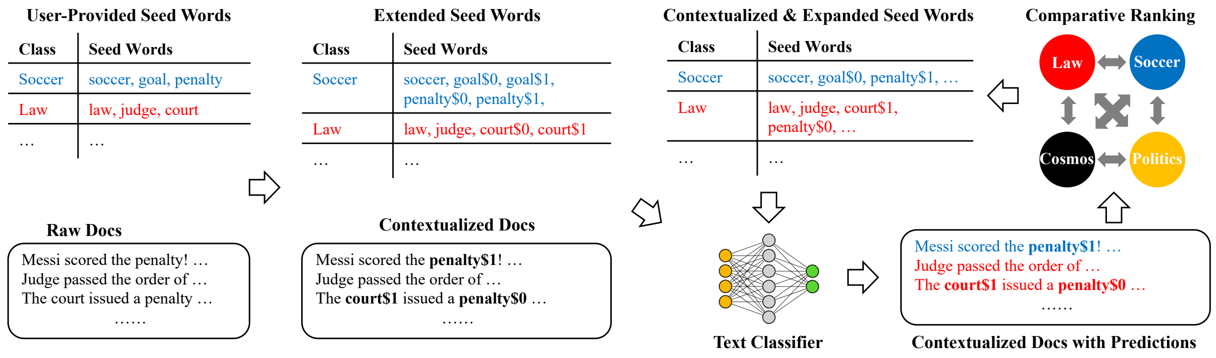

User-Provided Seed Words
Class | Seed Words
Soccer | soccer, goal, penalty
Law | law, judge, court
Extended Seed Words
Soccer | soccer, goal$0, goal$1, penalty$0, penalty$1,
Law | law, judge, court$0, court$1
Contextualized & Expanded Seed Words
Soccer | soccer, goal$0, penalty$1, …
Law | law, judge, court$1, penalty$0, …
Comparative Ranking
Law
Soccer
Cosmos
Politics
Raw Docs
Messi scored the penalty! …
Judge passed the order of …
The court issued a penalty …
Contextualized Docs
Messi scored the penalty$1! …
Judge passed the order of …
The court$1 issued a penalty$0 …
Text Classifier
Contextualized Docs with Predictions

Figure 1.2. Our proposed contextualized weakly supervised method leverages BERT to create a contextualized corpus. This contextualized corpus is further utilized to resolve interpretations of seed words, generate pseudo-labels, train a classifier and expand the seed set in an iterative fashion.

be mislabelled as *sports*. More importantly, such errors will further introduce wrong seed words, thus being propagated and amplified over the iterations.

Bearing these challenges in mind, we propose ConWea, a **Con**textualized **Wea**kly supervised text classification framework. This framework introduces contextualized weak supervision to train a text classifier based on user-provided seed words. The "contextualized" here is reflected in two places: the corpus and seed words. Every word occurrence in the corpus may be interpreted differently according to its context; Every seed word, if ambiguous, must be resolved according to its user-specified class. In this way, we aim to improve the performance of the final text classifier.

As illustrated in Figure 1.2, it leverages contextualized representation learning techniques, such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), together with user-provided seed information to first create a *contextualized corpus*. This contextualized corpus is further utilized to train the classifier and expand seed words in an iterative manner. During this process, *contextualized seed words* are introduced by expanding and disambiguating the initial seed words. Specifically, for each word, we develop an unsupervised method to adaptively decide its number of interpretations, and accordingly, group all its occurrences based on their contextualized representations. We design a principled comparative ranking method to select highly label-indicative keywords from the contextualized corpus, leading to contextualized seed words. We will repeat the iterative classification and seed word expansion process until the convergence.

To the best of our knowledge, this is the first work on contextualized weak supervision for text classification. It is also worth mentioning that our proposed framework is compatible with almost any contextualized representation learning models and text classification models. Our contributions are summarized as follows:

- We propose a novel framework enabling contextualized weak supervision for text classification.
- We develop an unsupervised method to automatically group word occurrences of the same word into an adaptive number of interpretations based on contextualized representations and user-provided seed information.
- We design a principled ranking mechanism to identify words that are discriminative and highly label-indicative.
- We have performed experiments on real-world datasets for both coarse- and fine-grained text classification tasks. The results demonstrate the superiority of using contextualized weak supervision, especially when the labels are fine-grained.

1.2 Related Work

In this section, we review the literature about (1) weak supervision for text classification methods, (2) contextualized representation learning techniques, (3) document classifiers, and (4) word sense disambiguation.

1.2.1 Weak Supervision for Text Classification

Weak supervision has been studied for building document classifiers in various of forms, including hundreds of labeled training documents (Tang et al., 2015b; Miyato et al., 2017; Xu et al., 2017), class/category names (Song & Roth, 2014; Tao et al., 2015; Li et al., 2018), and user-provided seed words (Meng et al., 2018; Tao et al., 2015). Our method focuses on user-provided seed words as the source of weak supervision, Along this line, Doc2Cube (Tao et al., 2015) expands label keywords from label surface names and performs multi-dimensional document classification by learning dimension-aware embedding; PTE (Tang et al., 2015b) utilizes both labeled and unlabeled documents to learn text embeddings specifically for a task, which are later fed to logistic regression classifiers for classification; WeSTClass (Meng et al., 2018) leverages seed information to generate pseudo documents and introduces a self-training module that bootstraps on real unlabeled data for model refining. This method is later extended to handle hierarchical classifications based on a pre-defined label taxonomy (Meng et al., 2019). However, all these weak supervisions follow a context-free manner. Here, we propose to use contextualized weak supervision.

1.2.2 Contextualized Word Representations

Contextualized word representation is originated from machine translation (MT). CoVe (McCann et al., 2017) generates contextualized representations for a word based on pre-trained MT models, More recently, ELMo (Peters et al., 2018) leverages neural language models to replace MT models, which removes the dependency on massive parallel texts and takes advantages of nearly unlimited raw corpora. Many models leveraging language modeling to build sentence representations (Howard & Ruder, 2018; Radford & Narasimhan, 2018; Devlin et al., 2019) emerge almost at the same time. Language models have also been extended to the character level (Liu et al., 2018; Akbik et al., 2018), which can generate contextualized representations for character spans.

Our proposed framework is compatible with all the above contextualized representation techniques. In our implementation, we choose to use BERT to demonstrate the power of using contextualized supervision.

1.2.3 Word Sense Disambiguation

Word Sense Disambiguation (WSD) is one of the challenging problems in natural language processing. Typical WSD models (Lesk, 1986; Zhong & Ng, 2010; Yuan et al., 2016; Raganato et al., 2017; Le et al., 2018; Tripodi & Navigli, 2019) are trained for a general domain. Recent works (Li & Jurafsky, 2015; Mekala et al., 2017; Gupta et al., 2019; Li et al., 2021) also showed that machine-interpretable representations of words considering its senses, improve document classification. However, if one wants to apply WSD to some specific corpus, additional annotated training data might be required to meet the similar performance as ours, which defeats the purpose of a weakly supervised setting.

In contrast, our contextualization, building upon (Devlin et al., 2019), is adaptive to the input corpus, without requiring any additional human annotations. Therefore, our framework is more suitable than WSD under the weakly supervised setting. Our experimental results have verified this reasoning and showed the superiority of our contextualization module over WSD in weakly supervised document classification tasks.

1.2.4 Document Classifier

Document classification problem has been long studied. In our implementation of the proposed framework, we used HAN (Yang et al., 2016), which considers the hierarchical structure of documents and includes attention mechanisms to find the most important words and sentences in a document. CNN-based text classifiers(Kim, 2014; Zhang et al., 2015; Lai et al., 2015) are also popular and can achieve inspiring performance.

Our framework is compatible with all the above text classifiers. We choose HAN just for a demonstration purpose.

1.3 Preliminaries

In this section, we introduce seed-driven weakly supervised text classification problem and provide an overview of our proposed framework.

1.3.1 Problem Formulation

The input of our problem contains (1) a collection of n text documents $\mathscr{D} = \{\mathscr{D}_1, \mathscr{D}_2, \ldots, \mathscr{D}_n\}$ and (2) m target classes $\mathscr{C} = \{\mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_m\}$ and their seed words $\mathscr{S} = \{\mathscr{S}_1, \mathscr{S}_2, \ldots, \mathscr{S}_m\}$. We aim to build a high-quality document classifier from these inputs, assigning class label $\mathscr{C}_j \in \mathscr{C}$ to each document $\mathscr{D}_i \in \mathscr{D}$.

Note that, all these words could be upgraded to phrases if phrase mining techniques (Liu et al., 2015; Shang et al., 2018) were applied as pre-processing. In this chapter, we stick to the words.

1.3.2 Framework Overview

We propose a framework, ConWea, enabling contextualized weak supervision. Here, “contextualized" is reflected in two places: the corpus and seed words. Therefore, we have developed two novel techniques accordingly to make both contextualizations happen.

First, we leverage contextualized representation learning techniques (Peters et al., 2018; Devlin et al., 2019) to create a contextualized corpus. We choose BERT (Devlin et al., 2019) as an example in our implementation to generate a contextualized vector of every word occurrence. We assume the user-provided seed words are of reasonable quality — the majority of the seed words are not ambiguous, and the majority of the occurrences of the seed words are about the semantics of the user-specified class. Based on these two assumptions, we are able to develop an unsupervised method to automatically group word occurrences of the same word into an adaptive number of interpretations, harvesting the contextualized corpus.

Second, we design a principled comparative ranking method to select highly label-indicative keywords from the contextualized corpus, leading to contextualized seed words. Specifically, we start with all possible interpretations of seed words and train a neural classifier. Based on the predictions, we compare and contrast the documents belonging to different classes, and rank contextualized words based on how label-indicative, frequent, and unusual these words are. During this process, we eliminate the wrong interpretations of initial seed words and also add more highly label-indicative contextualized words.

This entire process is visualized in Figure 1.2. We denote the number of iterations between classifier training and seed word expansion as T, which is the only hyper-parameter in our framework. We discuss these two novel techniques in detail in the following sections. To make this chapter self-contained, we will also brief the pseudo-label generation and document classifiers.

1.4 Document Contextualization

We leverage contextualized representation techniques to create a contextualized corpus. The key objective of this contextualization is to disambiguate different occurrences of the same word into several interpretations. We treat every word separately, so in the rest of this section, we focus on a given word w. Specifically, given a word w, we denote all its occurrences as $w_1, \ldots, w_n$, where n is its total number of occurrences in the corpus.

1.4.1 Contextualized Representation

First, we obtain a contextualized vector representation $\mathbf{b}_{w_i}$ for each w_i. Our proposed method is compatible with almost any contextualized representation learning model. We choose

BERT (Devlin et al., 2019) as an example in our implementation to generate a contextualized vector for each word occurrence. In this contextualized vector space, we use the cosine similarity to measure the similarity between two vectors. Two word occurrences w_i and w_j of the same interpretation are expected to have a high cosine similarity between their vectors $\mathbf{b}_{w_i}$ and $\mathbf{b}_{w_j}$. For the ease of computation, we normalize all contextualized representations into unit vectors.

1.4.2 Choice of Clustering Methods

We model the word occurrence disambiguation problem as a clustering problem. Specifically, we propose to use the K-Means algorithm (Jain & Dubes, 1988) to cluster all contextualized representations $\mathbf{b}_{w_i}$ into K clusters, where K is the number of interpretations. We prefer K-Means because (1) the cosine similarity and Euclidean distance are equivalent for unit vectors and (2) it is fast and we are clustering a significant number of times.

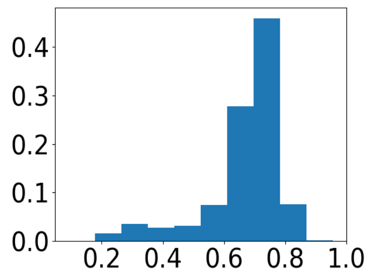
1.4.3 Automated Parameter Setting

We choose the value of K purely based on a similarity threshold τ. τ is introduced to decide whether two clusters belong to the same interpretation by checking if the cosine similarity between two cluster center vectors is greater than τ. Intuitively, we should keep increasing K until there exist no two clusters with the same interpretation. Therefore, we choose K to be the largest number such that the similarity between any two cluster centers is no more than τ.

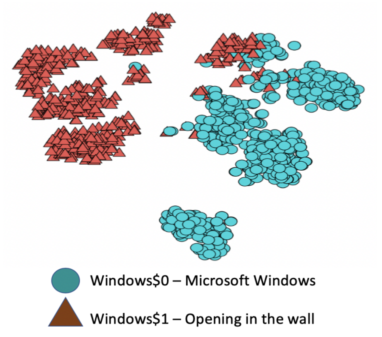
$$K = \arg\max_{K} \{\cos(\mathbf{c}_i, \mathbf{c}_j) < \tau \; \forall \, i, j\} \tag{1.1}$$

where $\mathbf{c}_i$ refers to the i-th cluster center vector after clustering all contextualized representations into K clusters. In practice, K is usually no more than 10. So we increase K gradually until the constraint is violated.

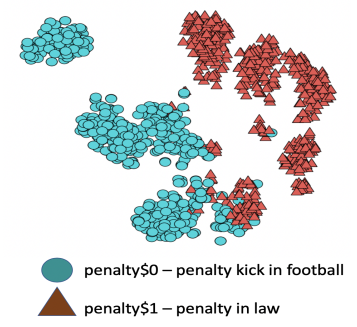
We pick τ based on user-provided seed information instead of hand-tuning, As mentioned, we make two "majority" assumptions: (1) For any seed word, the majority of its occurrences



(a) Similarity Distribution: Windows



(b) Cluster Visualisation: Windows



(c) Cluster Visualisation: Penalty

Figure 1.3. Document contextualization examples using word 'windows' and 'penalty'. τ is decided based on the similarity distributions of all seed word occurrences. Two clusters are discovered for both words, respectively.

follow the intended interpretation by the user; and (2) The majority of the seed words are not ambiguous — they only have one interpretation. Therefore, for each seed word s, we take the median of pairwise cosine similarities between its occurrences.

$$\tau(s) = \text{median}(\{\text{sim}(\mathbf{b}_{s_i}, \mathbf{b}_{s_j}) \mid \forall\, i, j\}) \tag{1.2}$$

Then, we take the median of these medians over all seed words as τ. Mathematically,

$$\tau = \text{median}(\{\tau(s) | \forall s\}) \tag{1.3}$$

The nested median solution makes the choice of τ safe and robust to outliers. For example, consider the word "windows" in the 20Newsgroup corpus. In fact, the word *windows* has two interpretations in the 20Newsgroup corpus — one represents an opening in the wall and the other is an operating system. We first compute the pairwise similarities between all its occurrences and plot the histogram as shown in Figure 1.3(a). From this plot, we can see that its median value is about 0.7. We apply the same for all seed words and obtain τ following Equation 1.3. τ is calculated to be 0.82. Based on this value, we gradually increase K for "windows" and it ends up with $K = 2$. We visualize its K-Means clustering results using t-SNE (Maaten & Hinton, 2008)

Algorithm 1: Corpus Contextualization

```
Input: Word occurrences $w_1, w_2, \ldots, w_n$ of the word $w$, Seed words $s_1, s_2, \ldots, s_m$ and
  their occurrences $s_{i,j}$.
Output: Contextualized word occurrences $\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_n$
Obtain $\mathbf{b}_{w_i}$ and $\mathbf{b}_{s_{i,j}}$ using BERT.
Compute $\tau$ follow Equation 1.3.
K ← 1
while True do
    Run K-Means on $\{b_{w_i}\}$ for (K+1) clusters.
    Obtain cluster centers $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{K+1}$.
    if $\max_{i,j} \cos(\mathbf{c}_i, \mathbf{c_j}) > \tau$ then
        Break
    K ← K + 1
Run K-Means on $\{b_{w_i}\}$ for K clusters.
Obtain cluster centers $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K$.
for each occurrence $w_i$ do
    Compute $\hat{w}_i$ following Equation 1.4.
Return $\hat{w}_i$.
```

in Figure 1.3(b). Similar results can be observed for the word *penalty*, as shown in Figure 1.3(c). These examples demonstrate how our document contextualization works for each word.

In practice, to make it more efficient, one can subsample the occurrences instead of enumerating all pairs in a brute-force manner.

1.4.4 Contextualized Corpus

The interpretation of each occurrence of w is decided by the cluster-ID to which its contextualized representation belongs. Specifically, given each occurrence w_i, the word w is replaced by $\hat{w}_i$ in the corpus as follows:

$$\hat{w}_i = \begin{cases} w & \text{if K = 1} \\ w\$j^* & \text{otherwise} \end{cases} \tag{1.4}$$

where

$$j^* = \arg\max_{j=1}^{K} \cos(\mathbf{b}_{w_i}, \mathbf{c}_j)$$

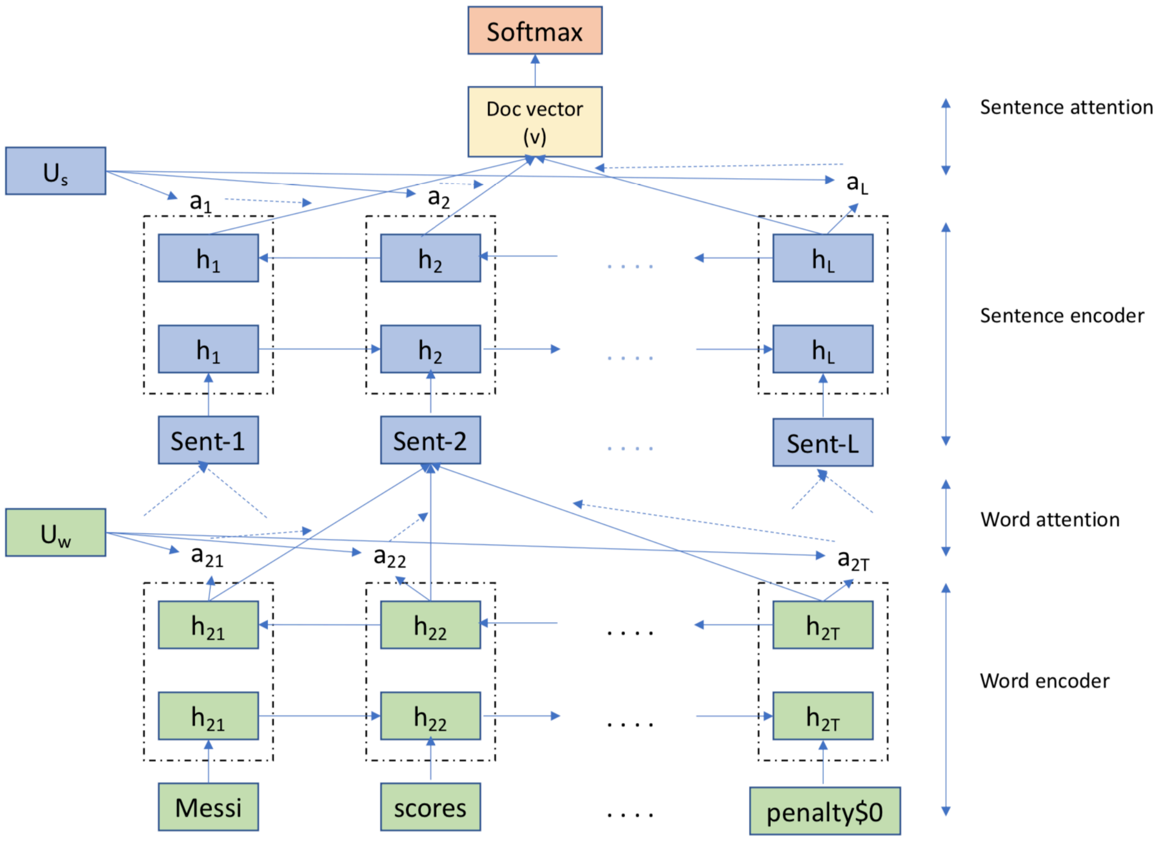


Figure 1.4. The HAN classifier used in our ConWea framework. It is trained on our contextualized corpus with the generated pseudo-labels.

By applying this to all words and their occurrences, the corpus is contextualized. The pseudo-code for corpus contextualization is shown in Algorithm 1.

1.5 Pseudo-Label and Text Classifier

In this section, we discuss pseudo-label generation and text classifier. These two parts are not the focus of the proposed method. We briefly introduce them to make the chapter self-contained.

We generate pseudo-labels for unlabeled contextualized documents and train a classifier based on these pseudo-labels, similar to many other weakly supervised methods (Agichtein & Gravano, 2000; Riloff et al., 2003; Kuipers et al., 2006; Tao et al., 2015; Meng et al., 2018).

1.5.1 Pseudo-Label Generation

There are several ways to generate pseudo-labels from seed words. As proof-of-concept, we employ a simple but effective method based on counting. Each document is assigned a label whose aggregated term frequency of seed words is maximum. Let $\mathrm{tf}(\hat{w}, d)$ denote term-frequency of a contextualized word w in the contextualized document d and $\mathscr{S}_c$ represents set of seed words of class c, the document d is assigned a label $l(d)$ as follows:

$$l(d) = \arg\max_{l} \{\sum_{i} tf(s_i, d) | \forall s_i \in \mathscr{S}_l\} \tag{1.5}$$

1.5.2 Document Classifier

Our framework is compatible with any text classification model. We use Hierarchical Attention Networks (HAN) (Yang et al., 2016) as an example in our implementation. HAN considers the hierarchical structure of documents (document – sentences – words) and includes an attention mechanism that finds the most important words and sentences in a document while taking the context into consideration. There are two levels of attention: word-level attention identifies the important words in a sentence and sentence level attention identifies the important sentences in a document. The overall architecture of HAN is shown in Figure 1.4. We train a HAN model on contextualized corpus with the generated pseudo-labels. The predicted labels are used in seed expansion and disambiguation.

1.6 Seed Expansion and Disambiguation

1.6.1 Seed Expansion

Given contextualized documents and their predicted class labels, we propose to rank contextualized words and add the top few words into the seed word sets. The core element of this process is the ranking function. An ideal seed word s of label l, is an unusual word that appears only in the documents belonging to label l with significant frequency. Hence, for a given class $\mathscr{C}_j$ and a word w, we measure its ranking score based on the following three aspects:

- **Label-Indicative.** Since our pseudo-label generation follows the presence of seed words in the document, ideally, the posterior probability of a document belonging to the class $\mathscr{C}_j$ after observing the presence of word w (i.e., $P(\mathscr{C}_j|w)$) should be very close to 100%. Therefore, we use $P(\mathscr{C}_j|w)$ as our label-indicative measure:

$$\mathbf{LI}(\mathscr{C}_j, w) = P(\mathscr{C}_j|w) = \frac{f_{\mathscr{C}_j,w}}{f_{\mathscr{C}_j}}$$

  where $f_{\mathscr{C}_j}$ refers to the total number of documents that are predicted as class $\mathscr{C}_j$, and among them, $f_{\mathscr{C}_j,w}$ documents contain the word w. All these counts are based on the prediction results on the input unlabeled documents.

- **Frequent.** Ideally, a seed word s of label l appears in the documents belonging to label l with significant frequency. To measure the frequency score, we first compute the average frequency of seed word s in all the documents belonging to label l. Since average frequency is unbounded, we apply *tanh* function to scale it, resulting in the frequency score,

$$\mathbf{F}(\mathscr{C}_j, w) = \tanh\left(\frac{f_{\mathscr{C}_j}(w)}{f_{C_j}}\right)$$

  Here, different from $f_{\mathscr{C}_j,w}$ defined earlier, $f_{\mathscr{C}_j}(w)$ is the frequency of word w in documents that are predicted as class $\mathscr{C}_j$.

- **Unusual.** We want our highly label-indicative and frequent words to be unusual. To incorporate this, we consider inverse document frequency (IDF). Let n be the number of documents in the corpus $\mathscr{D}$ and $f_{\mathscr{D},w}$ represents the document frequency of word w, the IDF of a word w is computed as follows:

$$\mathbf{IDF}(w) = \log\left(\frac{n}{f_{\mathscr{D},w}}\right)$$

Similar to previous work (Tao et al., 2015), we combine these three measures using the geometric mean, resulting in the ranking score $R(\mathscr{C}_j, w)$ of a word w for a class $\mathscr{C}_j$.

$$R(\mathscr{C}_j, w) = \big(\mathbf{LI}(\mathscr{C}_j, w) \times \mathbf{F}(\mathscr{C}_j, w) \times \mathbf{IDF}(w)\big)^{1/3}$$

Based on this aggregated score, we add top words to expand the seed word set of the class $\mathscr{C}_j$.

1.6.2 Seed Disambiguation

While the majority of user-provided seed words are nice and clean, some of them may have multiple interpretations in the given corpus. We propose to disambiguate them based on the ranking. We first consider all possible interpretations of an initial seed word, generate the pseudo-labels, and train a classifier. Using the classified documents and the ranking function, we rank all possible interpretations of the same initial seed word. Because the majority occurrences of a seed word are assumed to belong to the user-specified class, the intended interpretation shall be ranked the highest. Therefore, we retain only the top-ranked interpretation of this seed word.

After this step, we have fully contextualized our weak supervision, including the initial user-provided seeds.

1.7 Experiments

In this section, we evaluate our framework and many compared methods on coarse- and fine-grained text classification tasks under the weakly supervised setting.

1.7.1 Datasets

Following previous work (Tao et al., 2015), (Meng et al., 2018), we use two news datasets in our experiments. The dataset statistics are provided in Table 1.1. Here are some details.

- **The New York Times (NYT):** The NYT dataset contains news articles written and

Table 1.1. Dataset statistics.

Dataset	# Docs	# Coarse	# Fine	Avg Doc Len
NYT	13,081	5	25	778
20News	18,846	6	20	400

published by The New York Times. These articles are classified into 5 wide genres (e.g., arts, sports) and 25 fine-grained categories (e.g., dance, music, hockey, basketball).

- **The 20 Newsgroups (20News):** The 20News dataset[1] is a collection of newsgroup documents partitioned widely into 6 groups (e.g., recreation, computers) and 20 fine-grained classes (e.g., graphics, windows, baseball, hockey).

We perform coarse- and fine-grained classifications on the NYT and 20News datasets. NYT dataset is imbalanced in both fine-grained and coarse-grained classifications. 20News is nearly balanced in fine-grained classification but imbalanced in coarse-grained classification. Being aware of these facts, we adopt micro- and macro-F_1 scores as evaluation metrics.

1.7.2 Compared Methods

We compare our framework with a wide range of methods described below:

- **IR-TF-IDF** treats the seed word set for each class as a query. The relevance of a document to a label is computed by aggregated TF-IDF values of its respective seed words. The label with the highest relevance is assigned to each document.

- **Dataless** (Chang et al., 2008) uses only label surface names as supervision and leverages Wikipedia to derive vector representations of labels and documents. Each document is labeled based on the document-label similarity.

- **Word2Vec** first learns word vector representations (Mikolov et al., 2013a) for all terms in the corpus and derive label representations by aggregating the vectors of its respective

[1]http://qwone.com/~jason/20Newsgroups/

seed words. Finally, each document is labeled with the most similar label based on cosine similarity.

- **Doc2Cube** (Tao et al., 2015) considers label surface names as seed set and performs multi-dimensional document classification by learning dimension-aware embedding.
- **WeSTClass** (Meng et al., 2018) leverages seed information to generate pseudo documents and refines the model through a self-training module that bootstraps on real unlabeled documents.

We denote our framework as **ConWea**, which includes contextualizing corpus, disambiguating seed words, and iterative classification & key words expansion. Besides, we have three ablated versions. **ConWea-NoCon** refers to the variant of ConWea trained without the contextualization of corpus. **ConWea-NoSeedExp** is the variant of ConWea without the seed expansion module. **ConWea-WSD** refers to the variant of ConWea, with the contextualization module replaced by Lesk algorithm (Lesk, 1986), a classic Word-sense disambiguation algorithm (WSD).

We also present the results of **HAN-Supervised** under the supervised setting for reference. We use 80-10-10 for train-validation-test splitting and report the test set results for it. All weakly supervised methods are evaluated on the entire datasets.

1.7.3 Experiment Settings

We use pre-trained `BERT-base-uncased`[2] to obtain contextualized word representations. We follow (Devlin et al., 2019) and concatenate the averaged word-piece vectors of the last four layers.

The seed words are obtained as follows: we asked 5 human experts to nominate 5 seed words per class, and then considered the majority words (i.e., > 3 nominations) as our final set of seed words. For every class, we mainly use the label surface name as seed words. For some

[2]https://github.com/google-research/bert

Table 1.2. Evaluation Results for All Methods on Fine-Grained and Coarse-Grained Labels. Both micro-F_1(mic-F_1) and macro-F_1(mac-F_1) scores are presented. Ablation and supervised results are also included.

	NYT				**20 Newsgroup**			
	5-Class (Coarse)		**25-Class** (Fine)		**6-Class** (Coarse)		**20-Class** (Fine)	
Methods	Mic-F_1	Mac-F_1	Mic-F_1	Mac-F_1	Mic-F_1	Mac-F_1	Mic-F_1	Mac-F_1
IR-TF-IDF	0.65	0.58	0.56	0.54	0.49	0.48	0.53	0.52
Dataless	0.71	0.48	0.59	0.37	0.50	0.47	0.61	0.53
Word2Vec	0.92	0.83	0.69	0.47	0.51	0.45	0.33	0.33
Doc2Cube	0.71	0.38	0.67	0.34	0.40	0.35	0.23	0.23
WeSTClass	0.91	0.84	0.50	0.36	0.53	0.43	0.49	0.46
ConWea	**0.95**	**0.89**	**0.91**	**0.79**	**0.62**	**0.57**	**0.65**	**0.64**
ConWea-NoCon	0.91	0.83	0.89	0.74	0.53	0.50	0.58	0.57
ConWea-NoExpan	0.92	0.85	0.76	0.66	0.58	0.53	0.58	0.57
ConWea-WSD	0.83	0.78	0.72	0.64	0.52	0.46	0.49	0.47
HAN-Supervised	0.96	0.92	0.94	0.82	0.90	0.88	0.83	0.83

multi-word class labels (e.g., "international business"), we have multiple seed words, but never exceeds four per each class. The same seed words are utilized for all compared methods for fair comparisons.

For ConWea, we set $T = 10$. For any method using word embedding, we set its dimension to be 100. We use the public implementations of WeSTClass[3] and Dataless[4] with the hyper-parameters mentioned in their original papers.

1.7.4 Performance Comparison

We summarize the evaluation results of all methods in Table 1.2. As one can observe that our proposed framework achieves the best performance among all the compared weakly supervised methods. We discuss the effectiveness of ConWea as follows:

- Our proposed framework ConWea outperforms all the other methods with significant margins. By contextualizing the corpus and resolving the interpretation of seed words, ConWea achieves inspiring performance, demonstrating the necessity and the importance

[3]https://github.com/yumeng5/WeSTClass
[4]https://cogcomp.org/page/software_view/Descartes

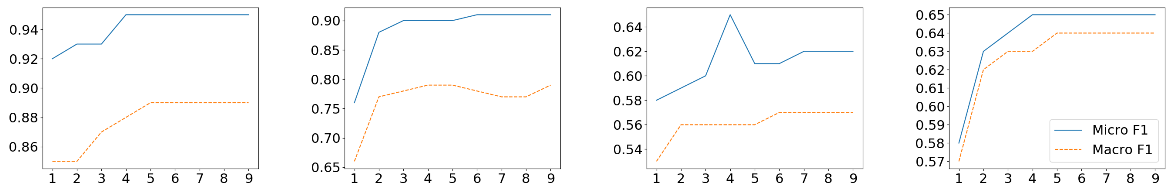
of using contextualized weak supervision.

- We observe that in the fine-grained classification, the advantages of ConWea over other methods are even more significant. This can be attributed to the contextualization of corpus and seed words. Once the corpus is contextualized properly, the subtle ambiguity between words is a drawback to other methods, whereas ConWea can distinguish them and predict them correctly.

- The comparison between ConWea and the ablation method ConWea-NoExpan demonstrates the effectiveness of our Seed Expansion. For example, for fine-grained labels on the 20News dataset, the seed expansion improves the micro-F1 score from 0.58 to 0.65.

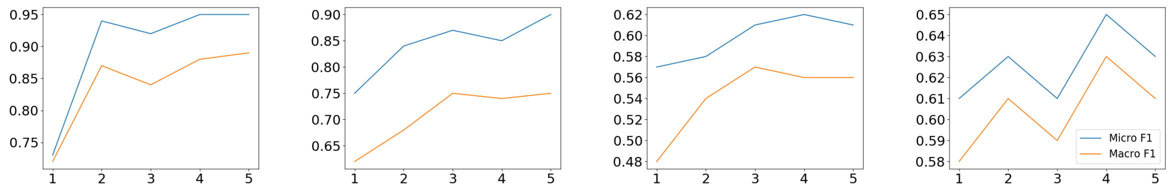
- The comparison between ConWea and the two ablation methods ConWea-NoCon and ConWea-WSD demonstrates the effectiveness of our Contextualization. Our contextualization, building upon (Devlin et al., 2019), is adaptive to the input corpus, without requiring any additional human annotations. However, WSD methods(e.g., (Lesk, 1986)) are typically trained for a general domain. If one wants to apply WSD to some specific corpus, additional annotated training data might be required to meet the similar performance as ours, which defeats the purpose of a weakly supervised setting. Therefore, we believe that our contextualization module has its unique advantages. Our experimental results further confirm the above reasoning empirically. For example, for coarse-grained labels on the 20News dataset, the contextualization improves the micro-F1 score from 0.53 to 0.62.

- We observe that ConWea performs quite close to supervised methods, for example, on the NYT dataset. This demonstrates that ConWea is quite effective in closing the performance gap between the weakly supervised and supervised settings.



(a) NYT Coarse (b) NYT Fine (c) 20News Coarse (d) 20News Fine

Figure 1.5. Micro- and Macro-F_1 scores w.r.t. the number of iterations.



(a) NYT Coarse (b) NYT Fine (c) 20News Coarse (d) 20News Fine

Figure 1.6. Micro- and Macro-F_1 scores w.r.t. the number of seed words.

1.7.5 Parameter Study

The only hyper-parameter in our algorithm is T, the number of iterations of iterative expansion & classification. We conduct experiments to study the effect of the number of iterations on the performance. The plot of performance w.r.t. the number of iterations is shown in Figure 1.5. We observe that the performance increases initially and gradually converges after 4 or 5 iterations. We observe that after the convergence point, the expanded seed words have become almost unchanged. While there is some fluctuation, a reasonably large T, such as $T = 10$, is a good choice.

1.7.6 Number of Seed Words

We vary the number of seed words per class and plot the F_1 score in Figure 1.6. One can observe that in general, the performance increases as the number of seed words increase. There is a slightly different pattern on the 20News dataset when the labels are fine-grained. We conjecture that it is caused by the subtlety of seed words in fine-grained cases – additional seed words may bring some noise. Overall, three seed words per class are enough for reasonable performance.

Table 1.3. Case Study: Seed word expansion of the *For Sale* class in context-free and contextualized corpora. The *For Sale* class contains documents advertising goods for sale. Blue bold words are potentially wrong seeds.

	Seed Words for *For Sale* class	
Iter	**Plain Corpus**	**Contextualized Corpus**
1	sale, offer, forsale	sale, offer, forsale
2	**space**, price, shipping, sale, offer	shipping, forsale, offer$0, condition$0, sale
3	**space**, price, shipping, sale, **nasa**, offer, package, email	price, shipping, sale, forsale, condition$0, offer$0, package, email
4	**space**, price, **moon**, shipping, sale, **nasa**, offer, **shuttle**, package, email	price, shipping, sale, forsale, condition$0, offer$0, package, email, offers$0, obo$0

1.7.7 Case Study

We present a case study to showcase the power of contextualized weak supervision. Specifically, we investigate the differences between the expanded seed words in the plain corpus and contextualized corpus over iterations. Table 1.3 shows a column-by-column comparison for the class *For Sale* on the 20News dataset. The class *For Sale* refers to documents advertising goods for sale. Starting with the same seed sets in both types of corpora, from Table 1.3, in the second iteration, we observe that "space" becomes a part of expanded seed set in the plain corpus. Here "space" has two interpretations, one stands for the physical universe beyond the Earth and the other is for an area of land. This error gets propagated and amplified over the iterations, further introducing wrong seed words like "nasa", "shuttle" and "moon", related to its first interpretation. The seed set for contextualized corpus addresses this problem and adds only the words with appropriate interpretations. Also, one can see that the initial seed word "offer" has been disambiguated as "offer$0".

1.8 Summary

In this chapter, we proposed ConWea, a novel contextualized weakly supervised classification framework. Our method leverages contextualized representation techniques and initial user-provided seed words to contextualize the corpus. This contextualized corpus is further used to resolve the interpretation of seed words through iterative seed word expansion and document classifier training. Experimental results demonstrate that our model outperforms previous methods significantly, thereby signifying the superiority of contextualized weak supervision, especially when labels are fine-grained.

In the future, we are interested in generalizing contextualized weak supervision to hierarchical text classification problems. Currently, we perform coarse- and fine-grained classifications separately. There should be more useful information embedded in the tree-structure of the label hierarchy. Also, extending our method for other types of textual data, such as short texts, multi-lingual data, and code-switched data is a potential direction.

Chapter 1, in full, is a reprint of the material as it appears in Mekala, Dheeraj; Shang, Jingbo. "Contextualized weak supervision for text classification," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 323–333, 2020. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 2

META: Metadata-Empowered Weak Supervision for Text Classification

In this chapter, we present our proposed method META, that leverages metadata information as an additional source of weak supervision and incorporates it into the classification framework.

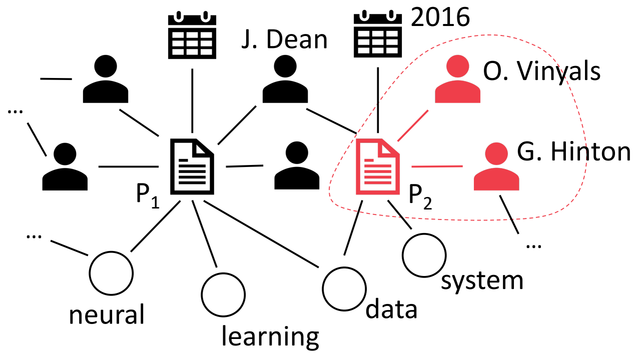
2.1 Motivation & Overview

Weakly supervised text classification has recently gained much attention from the researchers because it reduces the burden of annotating the data. So far, the major source of weak supervision lies in text data itself (Agichtein & Gravano, 2000; Kuipers et al., 2006; Riloff et al., 2003; Tao et al., 2015; Meng et al., 2018; Mekala & Shang, 2020). These methods typically require a few user-provided seed words for each class as weak supervision. They expand seed words with generated pseudo labels and improve their text classifier in an iterative fashion.

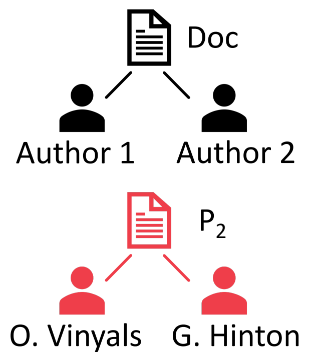
Metadata information (e.g., author, published year) in addition to textual information, is widely available across various domains (e.g., news articles, social media posts, and scientific papers) and it could serve as a strong, complementary weak supervision source. Take a look at the research papers in Figure 2.1(a) as an example. It shall be learned in a data-driven manner that *G. Hinton* is a highly-reputed machine learning researcher, thus his presence is a strong indicator of a paper belonging to the *Machine Learning* category.

Paper	Authors	Year	Category
P_1	G. Hinton, S. Osindero, YW. Teh	2006	ML
P_2	G. Hinton, O. Vinyals, J. Dean	2015	ML
P_3	J. Dean, S.Ghemawat	2008	Sys

(a) Examples of research papers with metadata.



(b) A text-rich network view of the papers.



(c) A motif pattern and a motif instance

Figure 2.1. Text corpus, text-rich network, and motif.

Distilling effective metadata for weak supervision faces several major challenges. Metadata is often multi-typed, each type and the type combinations could have very different semantics and may not be equally important. Moreover, even entities within a single metadata type could be noisy. Continuing our example in Figure 2.1(a), we shall notice that *year* is less helpful than an *author* to do classification. Among the authors, *J. Dean* might be an important figure but has research interests spanning across different domains. However, if we join the *author* with *year*, it carries more accurate semantics, and we may discover *J. Dean* has more interest in machine learning in recent years, thus becoming highly label-indicative.

Bearing the challenges in mind, we propose META, a principled framework for metadata-empowered weakly-supervised text classification. As illustrated in Figure 2.1 and Figure 2.2, we first organize the text data and metadata together into a text-rich network. The network structure gives us a holistic view of the corpus and enables us to rank and select useful metadata entities. We leverage motif patterns (Benson et al., 2016; Milo et al., 2002; Shang et al., 2020) to model typed metadata as well as their combinations. A motif pattern is a subgraph pattern at

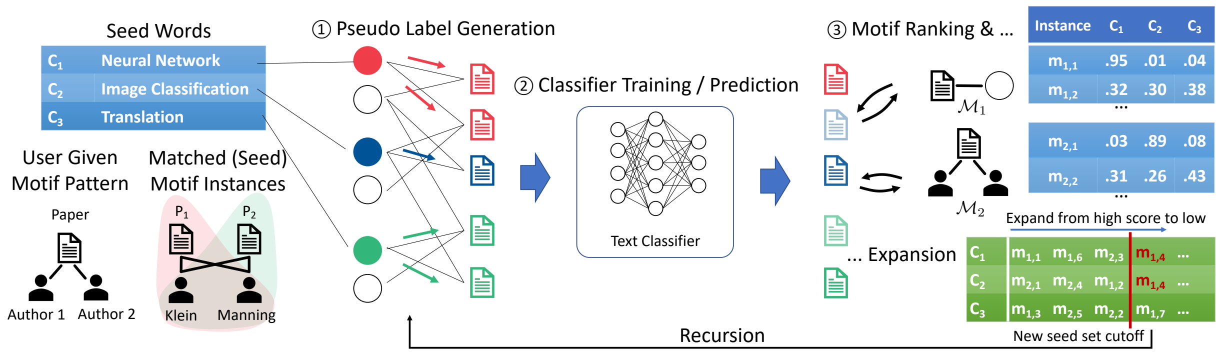


Figure 2.2. Our META framework. In each iteration, we generate pseudo labels for documents, train the text classifier, and rank all words and motif instances in a unified ranking framework. We then expand seed sets until an automatic cutoff is reached.

the meta-level that captures higher-order connections and the semantics represented by these connections. It serves as a useful tool to model typed edges, typed paths (a.k.a. meta-paths) (Sun et al., 2011), and higher-order structures in the network. With little effort, users can specify a few possibly useful motif patterns as input to our model. We develop a unified, principled ranking mechanism to select label-indicative motif instances and words, forming expanded weak supervision. Note that, such instance-level selection process also implicitly refines the motif patterns, ensuring the robust performance of META even when irrelevant motif patterns exist in input. It is worth a mention that META is compatible with any text classifiers.

Our contributions are summarized as follows:

- We explore to incorporate metadata information as an additional source of weak supervision for text classification along with seed words.
- We propose a novel framework META, which introduces motif patterns to capture the high-order combinations among different types of metadata and conducts a unified ranking and selection of label-indicative motif instances and words.
- We conduct experiments on two real-world datasets. The results and case studies demonstrate the superiority of incorporating metadata as parts of weak supervision and verify the effectiveness of META.

2.2 Related Work

In this section, we review the literature about (1) weakly supervised text classification methods, (2) text classification with metadata, and (3) document classifiers.

2.2.1 Weakly Supervised Text Classification

Due to the training data bottleneck in supervised classification, weakly supervised classification has recently attracted much attention from researchers. The majority of weakly supervised classification techniques require seeds in various forms, including label surface names (Li et al., 2018; Song & Roth, 2014; Tao et al., 2015), label-indicative words (Chang et al., 2008; Meng et al., 2018; Tao et al., 2015; Mekala & Shang, 2020), and labeled-documents (Tang et al., 2015b; Xu et al., 2017; Miyato et al., 2017; Meng et al., 2018).

Dataless (Song & Roth, 2014) considers label surface names as seeds and classifies documents by embedding both labels and documents in a semantic space and computing semantic similarity between a document and a potential label; Along similar lines, Doc2Cube (Tao et al., 2015) expands label-indicative words using label surface names and performs multi-dimensional document classification by learning dimension-aware embedding; WeSTClass (Meng et al., 2018) considers both word-level and document level supervision sources. It first generates bag-of-words pseudo documents for neural model pre-training, then bootstraps the model on unlabeled data. This method is later extended to a hierarchical setting with a pre-defined hierarchy (Meng et al., 2019); ConWea (Mekala & Shang, 2020) leverages contextualized representation techniques to provide contextualized weak supervision for text classification.

However, all these techniques consider only the text data and don't leverage metadata information for classification. In this chapter, we focus on user-provided seed words and mine label-indicative words and metadata in an iterative manner.

2.2.2 Text Classification with Metadata

Previous studies try to incorporate metadata information to improve the performance of the classifier. (Tang et al., 2015a) and (Chen et al., 2016) consider the user and product information as metadata for document-level sentiment classification; (Rosen-Zvi et al., 2012) use author information for paper classification; (Zhang et al., 2017) employ user biography data for tweet localization. However, all these frameworks are in a supervised setting and use fixed metadata types for each task whereas our method is generalized for different metadata types and multiple metadata combinations.

Another way to leverage metadata for text understanding is to organize the corpus into a heterogeneous information network. A straightforward approach is to obtain document representations using their respective meta-path guided node embeddings (Dong et al., 2017; Shang et al., 2016) and train a classifier. However, higher-order connectivity cannot be captured by meta-paths and this approach can't handle new documents directly without re-training the embeddings. Recently, (Zhang et al., 2020b) proposed a minimally supervised framework to categorize text with metadata. However, they require labeled documents as supervision and they only consider typed edges in the model. Network motifs (Milo et al., 2002) can capture higher-order connectivity and have been proved fundamental in complex real-word networks across various domains (Benson et al., 2016). (Shang et al., 2020) leveraged motifs for topic taxonomy construction in an unsupervised setting. Our proposed method mines highly label-indicative metadata information with a unified motif and word ranking framework, and effectively expands weak supervision to improve document classification. (Wang et al., 2023) introduced a benchmark on weakly supervised text classification.

2.2.3 Document classifier

Document classification has been a long-studied problem in Natural Language Processing. CNN-based classifiers (Kim, 2014; Johnson & Zhang, 2014; Lai et al., 2015), RNN-based classifiers (Socher et al., 2013) achieve competitive performance. (Yang et al., 2016) proposed Hierarchical Attention Network (HAN) for document classification that performs attention first on the sentences in the document, and on the words in the sentence to find the most important sentences and words in a document. Though our framework uses HAN as the document classifier, it is also compatible with all the above-mentioned text classifiers. We choose HAN for the demonstration purpose.

2.3 Preliminaries

In this section, we briefly discuss the concepts that are essential to understand our framework such as Text-rich network, motif pattern and motif instances. In the end, we formally formulate the problem and provide a brief description of our proposed framework.

2.3.1 Documents as Text-rich Network

Given a collection of n text documents $\mathscr{D} = \{\mathscr{D}_1, \mathscr{D}_2, \ldots, \mathscr{D}_n\}$, and their corresponding metadata, we propose to organize them into a *text-rich network*, as illustrated in Figure 2.1(b). A text-rich network is a heterogeneous network with documents, words, different types of metadata as nodes, and their associations as edges. For example, our text-rich network for research papers has papers, words, authors, and publication years as nodes. Each paper is connected to its associated words and metadata nodes. Such a network provides a holistic and structured representation of the input.

2.3.2 Seed Words and Motif Patterns

Users are asked to provide a few seed words $\mathscr{S} = \{\mathscr{S}_1^w, \mathscr{S}_2^w, \ldots, \mathscr{S}_l^w\}$ for each of l classes (i.e., $\mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_l$) in our classification problem, as well as k motif patterns $\{\mathscr{M}_1, \mathscr{M}_2, \ldots, \mathscr{M}_k\}$. Motif patterns are sub-graph patterns at the meta-level (i.e., every node is abstracted by its type). They are able to capture semantics and higher-order inter-connections among nodes. A motif instance is a sub-graph instance in the graph that follows a motif pattern. Figure 2.1(c) presents an example of a motif pattern that captures co-authorship and a motif instance following this motif pattern. In this framework, we discover seed motif instances for each class label, denoted as $\{\mathscr{S}_1^m, \mathscr{S}_2^m, \ldots, \mathscr{S}_l^m\}$.

2.3.3 Problem Formulation

Given the text-rich network and user-provided seed words and motif patterns as input, we aim to build a document classifier, assigning a class label $\mathscr{C}_j$ to each document $\mathscr{D}_i$.

2.3.4 Framework Overview

As shown in Figure 2.2, META is an iterative framework, generating pseudo labels and training the text classifier alternatively, similar to many other weakly supervised text classification methods (Kuipers et al., 2006; Tao et al., 2015; Meng et al., 2018). One iteration in META consists of the following steps:

- Generate pseudo labels based on the seeds.
- Train a text classifier based on pseudo labels.
- Rank and select words and motif instances to expand the seeds.

We repeat these steps iteratively. We denote the number of iterations as T, which is the only hyper-parameter in our framework.

The novelty of META mainly lies in integrating two sources of weak supervisions, seed motif instances, and seed words. Given each motif instance m or each word w, for each label l, we estimate a *ranking score* $\mathscr{R}_{m,l}$ or $\mathscr{R}_{w,l}$ ranging between 0 and 1, measuring how label-indicative it is to the particular label l. Such ranking scores are utilized to select new seed motif instances and seed words. Note that, while this selection is conducted at the instance level, it also selects motif patterns implicitly and therefore ensures robust performance when users provide some irrelevant motif patterns.

2.4 Pseudo Labels and Text Classifier

In this section, we discuss pseudo-label generation and text classifier. Based on seed words, seed motif instances, and their respective ranking scores for each class, we generate pseudo labels for unlabeled text documents and train a classifier based on these pseudo labels. In the first iteration, we have no seed motif instances and the ranking score is 1 for all seed words.

2.4.1 Pseudo-Label Generation

Suppose we have seed word sets $\mathscr{S}^w_{1..l}$ and seed motif instance sets $\mathscr{S}^m_{1..l}$ for all l labels, we generate pseudo labels using a simple yet effective count-based technique. Specifically, given a document $\mathscr{D}_i$, the probability that it belongs to the class l is proportional to the aggregated ranking scores of its respective seed words and seed motif instances.

$$P(l|\mathscr{D}_i) \propto \sum_{w \in \mathscr{D}_i \cap \mathscr{S}^w_l} f_{\mathscr{D}_i,w} \cdot \mathscr{R}_{w,l} + \sum_{m \in \mathscr{D}_i \cap \mathscr{S}^m_l} \mathscr{R}_{m,l}$$

where $f_{\mathscr{D}_i,w}$ is the term frequency of the word w in document $\mathscr{D}_i$. The pseudo label of document $\mathscr{D}_i$ is then assigned as follows:

$$l(\mathscr{D}_i) = \arg\max_l P(l|\mathscr{D}_i)$$

2.4.2 Document Classifier

Our framework is compatible with any text classification model as a classifier. We use Hierarchical Attention Networks (HAN) (Yang et al., 2016) as the classifier. HAN is designed to capture the hierarchical document structure i.e. words – sentences – documents. As illustrated in Figure 2.3, HAN performs attention first on the sentences in the document to find the important sentence in a document and on the words in the sentence to identify important words in a sentence. We train a HAN model on unlabeled documents with the generated pseudo-labels. For

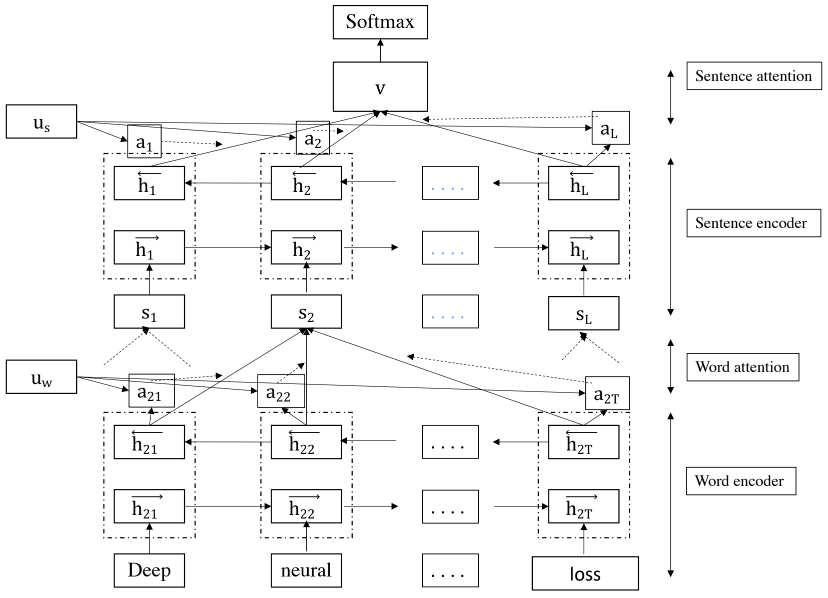


Figure 2.3. HAN Classifier used in our META.

the document $\mathscr{D}_i$, it estimates the probability $\hat{Y}_{i,l}$ for each class l. Such predicted distributions are used in the expansion of seed words and motifs.

2.5 Unified Seed Ranking and Expansion

In this section, we describe the ranking score and seed expansion technique. Once the text classifier is trained, we rank words and motif instances together for each class. Then, we expand the seed sets by adding top-ranked words and motif instances. This improves the quality of the weak supervision over iterations, thereby improving the text classifier. We present our design of the unified ranking and expansion as follows.

2.5.1 Ranking Score Design

An ideal seed word or motif instance for a particular class should be highly relevant and highly exclusive to this class. So an effective ranking score must quantify relevance and exclusiveness. Such a ranking score for words alone has been explored by previous studies (Tao et al., 2015; Mekala & Shang, 2020), typically based on similarity and frequency-based metrics. In this framework design, we have motif instances in addition to words, therefore, we build upon

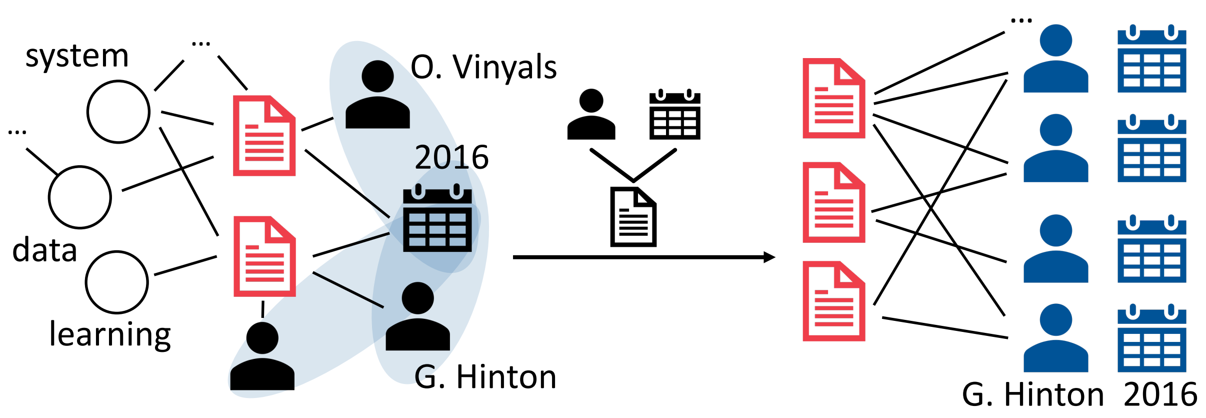


Figure 2.4. Using motif patterns, we construct bipartite graphs from the text-rich network linking documents to their respective motif instances.

the text-rich network to unify the ranking process.

Given k user-provided motif patterns $\mathscr{M}_1$, …, $\mathscr{M}_k$ and the text-rich network $\mathscr{G}$, we construct k bipartite graphs $\mathscr{G}_1^B, \ldots, \mathscr{G}_k^B$, one for each motif pattern (see Figure 2.4). In the i-th bipartite graph $\mathscr{G}_i^B$, the node set contains two parts: (1) all documents and (2) all motif instances following the motif pattern $\mathscr{M}_i$ in the text-rich network $\mathscr{G}$; The edges in the graph $\mathscr{G}_i^B$ connect the documents to the motif instances which are subsets of the metadata associated with the documents.

For the sake of simplicity, we introduce one more motif pattern, document–word. It makes words a special case of motif instances, and one can easily construct a similar bipartite graph for words. Therefore, in the rest of this section, we use motif instances to explain our ranking score design.

For each motif pattern $\mathscr{M}$, we conduct one personalized random walk on its corresponding bipartite graph $\mathscr{G}^B$ for each label l. Specifically, we normalize each column of the adjacency matrix of the bipartite graph $\mathscr{G}^B$ by the degree of its respective node, resulting in the transition matrix $\mathbf{W}$. Suppose $\mathbf{p}_{l,u}$ represents the personalized PageRank (PPR) score of each node u for each label l, we initialize the PPR score of each document node to $\hat{Y}_{i,l}$ and PPR score of each motif instance node to 0. This initialization ensures that a random walk starts from a document node and since $\mathscr{G}^B$ is bipartite, it ends at a motif instance node. We iteratively update the PPR scores as follows:

$$\mathbf{p}_l^{(t+1)} \leftarrow \mathbf{W}\mathbf{p}_l^{(t)}$$

Since each document node is initialized with probabilities corresponding to l and the random walk starts from a document node and ends at a motif instance node, this can be viewed as a label propagation problem. Based on the previous work in label propagation (Hensley et al., 2015), similar nodes are more likely to form edges and the PPR score is used to measure the similarity. Therefore, we believe that $\mathbf{p}_{l,m}$ reflects the *relevance* of a motif instance m to the particular class label l.

Though the absolute values of PPR scores are quite small, their relative magnitude conveys their affinity towards a label. Therefore, we normalize these PPR scores into a distribution, resulting in the ranking scores. Mathematically, for a label l, the ranking score of a motif instance m is:

$$\mathscr{R}_{m,l} = \frac{\mathbf{p}_{l,m}}{\sum_{l' \in \mathscr{C}} \mathbf{p}_{l',m}}$$

If a motif instance has similar relevance to multiple labels, the ranking score distribution becomes flat irrespective of the magnitude of its respective PPR scores. From this, we realize that our ranking score also quantifies *exclusiveness*, which is an essential characteristic of a highly label-indicative term.

Based on this ranking score, we rank words and motif instances in a unified manner and expand the seed word set and seed motifs set.

2.5.2 Expansion

Given the ranking scores of all words and motif instances for every label, we expand the seed words and seed motifs simultaneously for all labels. Intuitively, a highly label-indicative motif instance would not belong to the seed sets of multiple labels. Therefore, when any motif instance is expanded to seed sets of multiple classes, we stop the expansion of motif instances of the corresponding motif pattern. Also, we set a hard threshold of $\frac{1}{|\mathscr{C}|}$, where $|\mathscr{C}|$ is the number of

Table 2.1. Dataset statistics.

Dataset	# Docs	# Classes	Avg Doc Len
DBLP	38,128	9	893
Book Graph	33,594	8	620

classes, on ranking scores for those added motif instances. In this way, the number of new seed words and seed motif instances is decided by the method automatically. It is worth mentioning that our expansion here is adaptive and every label may have a different number of seeds. Note that, in the first iteration, pseudo labels are generated using only seed words but ranking scores are obtained for all words and motif instances. The highly ranked motif instances and words are used as seeds in further iterations.

After expanding the seed sets for every label, we generate pseudo labels and train the classifier. This process is repeated iteratively for T iterations.

2.6 Experiments

In this section, we evaluate META and compare it with existing techniques on two real-world datasets in a weakly supervised classification setting.

2.6.1 Datasets

We conduct experiments on the DBLP dataset (Tang et al., 2008) and the Book Graph dataset (Wan & McAuley, 2018; Wan et al., 2019). The dataset statistics are shown in Table 2.1. The details of the datasets are mentioned below.

- **DBLP dataset:** The DBLP dataset contains a comprehensive set of research papers in computer science. We select $38,128$ papers published in flagship venues. In addition to text data, it has information about authors, published year, and venue for each paper. There are $9,300$ distinct authors and 42 distinct years. For each paper, we annotate its research area largely based on its venue as the classification objective[1]. Therefore, in our

[1]Classes in DBLP: (1) computer vision, (2) computational linguistics, (3) biomedical engineering, (4) software

experiments, we drop the venue information to ensure a fair comparison.

- **Book Graph dataset:** The Book Graph dataset is a collection of the description of books, user-book interactions, and users' book reviews collected from a popular online book review website named Goodreads[2]. We select books belonging to eight popular genres[3]. The genre of a book is viewed as the label to be predicted. The total number of books selected is $33,594$. We use the title and description of a book as text data and author, publisher, and year as metadata. In total, there are $22,145$ distinct authors, $5,186$ distinct publishers, and 136 distinct years.

2.6.2 Motif Patterns

The motif patterns we used as metadata information for DBLP and Book Graph datasets are shown in Figure 2.5.

2.6.3 Seed Words

The seed words are obtained as follows: we asked 5 human experts to recommend 5 seed words for each class and selected the final seed words based on majority voting i.e. (> 3 recommendations).

2.6.4 Evaluation Metrics

Both datasets are imbalanced with respect to the label distribution. Being aware of this fact, we adopt micro- and macro-F_1 scores as evaluation metrics.

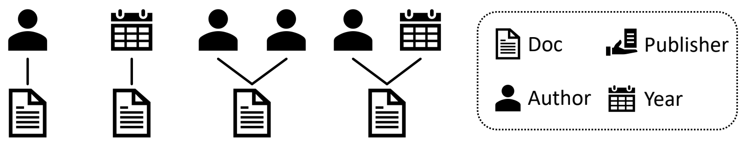
2.6.5 Implementation Details

To make the model robust to multi-word phrases as supervision, we extract phrases using Autophrase (Liu et al., 2015; Shang et al., 2018). We set the word vector dimension to be 100

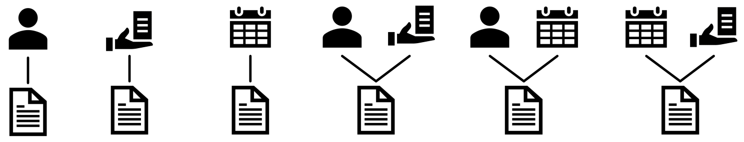
engineering, (5) graphics, (6) data mining, (7) security and cryptography, (8) signal processing, (9) robotics, and (10) theory.

[2]https://www.goodreads.com/

[3]Classes in Book Graph: (1) children, (2) graphic comics, (3) paranormal fantasy, (4) history & biography, (5) crime, mystery thriller, (6) poetry, (7) romance, and (8) young adult.



(a) Motif patterns: DBLP



(b) Motif patterns: Book Graph

Figure 2.5. Motif Patterns used in Experiments.

for all the methods that use word embeddings. We set the number of iterations parameter for META to 9.

2.6.6 Compared Methods

We compare our proposed method with a wide range of methods described below:

- **IR-TF-IDF** treats seed words as a query. It computes the relevance of a document to a class by aggregating the TF-IDF values of its seed words. Each document is assigned the label which is the most relevant to this document.
- **Word2Vec** learns word vector representations (Mikolov et al., 2013b) for all words in the corpus. It computes label representations by aggregating the word vectors of all its seed words. Each document is assigned the label whose cosine similarity with this document is maximum.
- **Doc2Cube** (Tao et al., 2015) considers label surface names as seed set and performs multi-dimensional document classification by learning dimension-aware embedding.
- **WeSTClass** (Meng et al., 2018) leverages seed words to generate bag-of-words pseudo documents for neural model pre-training and then bootstraps the model on unlabeled data. Specifically, we compare with WeSTClass-CNN which is the best configuration under

our setting. We use the public implementations of WeSTClass[4] with the hyperparameters mentioned in the paper.

- **Metapath2Vec** (Dong et al., 2017) learns node representations in the text-rich network using meta-path-guided random walks by capturing the structural and semantic correlations of differently typed nodes. We use the first two motif patterns in Figure 2.5(a) and the first three motif patterns in Figure 2.5(b) as meta-paths because the rest cannot be represented as meta-paths. We generate pseudo-labels using the seed words and train a logistic regression classifier with document nodes representations as input to predict the labels.

We denote our framework with HAN classifier as **META**, with CNN classifier as **META-CNN**, and with BERT(bert-base-uncased) classifier as **META-BERT**. We also compare with their respective ablated versions **META-NoMeta**, **META-CNN-NoMeta**, **META-BERT-NoMeta** where metadata information is not expanded and not considered while generating pseudo labels.

For a fair comparison, we also present results of all the baselines on the **metadata-augmented** datasets, where a token for every relevant motif instance is appended to the text data of a document. This is denoted by **++** in Table 2.2, e.g., WeSTClass++ represents the performance of WeSTClass on metadata-augmented datasets.

We also present the performance of HAN in a supervised setting which is denoted as **HAN-Sup**. The results of HAN-Sup reported are on the test set which follows an 80-10-10 train-dev-test split.

2.6.7 Performance Comparison

The evaluation results of all methods are summarized in Table 2.2. We can observe that our proposed framework outperforms all the compared weakly supervised methods. We discuss the effectiveness of our proposed META as follows:

- META achieves the best performance among all the compared weakly supervised methods

[4]https://github.com/yumeng5/WeSTClass

Table 2.2. Evaluation Results on Two Datasets. **++** represents that the input is metadata-augmented.

	DBLP		**Books Graph**	
Methods	Mi-F_1	Ma-F_1	Mi-F_1	Ma-F_1
IR-TF-IDF	0.19	0.20	0.24	0.29
Word2Vec	0.23	0.22	0.28	0.26
Doc2Cube	0.37	0.36	0.33	0.31
WeSTClass	0.58	0.53	0.42	0.41
Metapath2Vec	0.64	0.61	0.47	0.48
IR-TF-IDF++	0.19	0.20	0.24	0.29
Word2Vec++	0.24	0.21	0.26	0.25
Doc2Cube++	0.40	0.38	0.36	0.33
WeSTClass++	0.60	0.55	0.47	0.43
META	**0.66**	**0.63**	0.62	**0.63**
META-CNN	0.61	0.58	0.54	0.55
META-BERT	0.64	0.61	**0.63**	**0.63**
META-NoMeta	0.61	0.58	0.58	0.58
META-CNN-NoMeta	0.56	0.53	0.53	0.53
META-BERT-NoMeta	0.58	0.57	0.60	0.60
HAN-Sup	0.75	0.72	0.77	0.76
HAN-Sup++	0.79	0.77	0.81	0.81

with significant margins. By extracting the highly label-indicative motif instances along with words and using them together in pseudo label generation, META successfully leverages metadata information and achieves superior performance.

- We observe that the performance of META is better than all the compared weakly supervised models on metadata-augmented datasets. By comparing those **++** methods with their text-only counterparts, one can easily observe that adding metadata in text classification is indeed helpful. However, META does not restrict to single metadata types and goes beyond by employing motif patterns to capture the metadata information. It is successful in identifying the appropriate label-indicative metadata combinations and therefore achieves even better performance.

- The comparison between META and Metapath2Vec demonstrates the advantages of motif patterns over the meta-paths. For example, on the Book Graph dataset, the last three motif

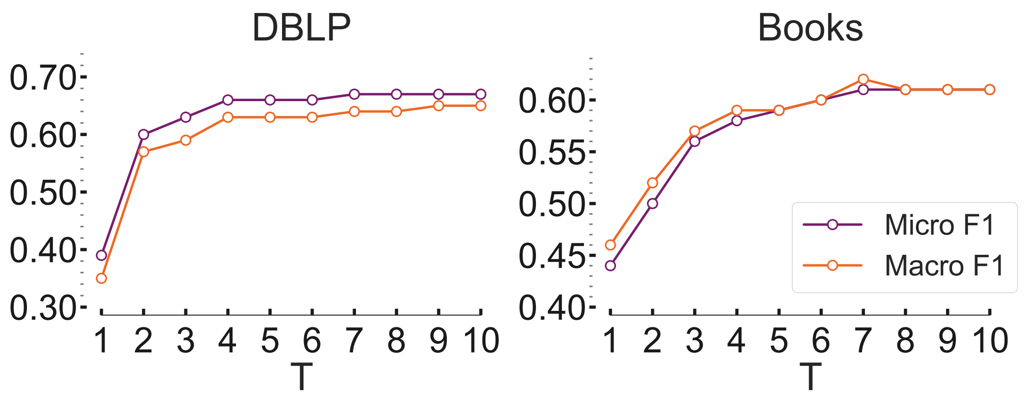


Figure 2.6. Micro- and Macro-F_1 scores w.r.t. the number of iterations.

patterns in Figure 2.5(b) cannot be represented through meta-paths and this significantly affects the performance. It's also worth mentioning that Metapath2Vec cannot handle new documents directly without re-training the embedding whereas our framework can directly predict without any additional effort.

- The comparison between META and the ablation method META-NoMeta demonstrates the effectiveness of our motif instance expansion. For example, on the Book Graph dataset, the motif instance expansion improves the micro-F1 score from 0.58 to 0.62 and macro-F1 score from 0.58 to 0.63, which are quite significant.
- The comparison between META-CNN, META-BERT, and their respective ablated versions META-CNN-NoMeta, META-BERT-NoMeta demonstrate that our proposed approach provides significant additive gains to different classifiers and thereby showing the effectiveness of leveraging metadata information as an additional source of weak supervision.
- The comparison between META and HAN-Sup demonstrates that META is effective in decreasing the gap between the performance of the weakly supervised and supervised settings.

2.6.8 Parameter Study

The only hyper-parameter in our framework META is T, the number of iterations. We experiment on both datasets to study the effect of the number of iterations on the performance.

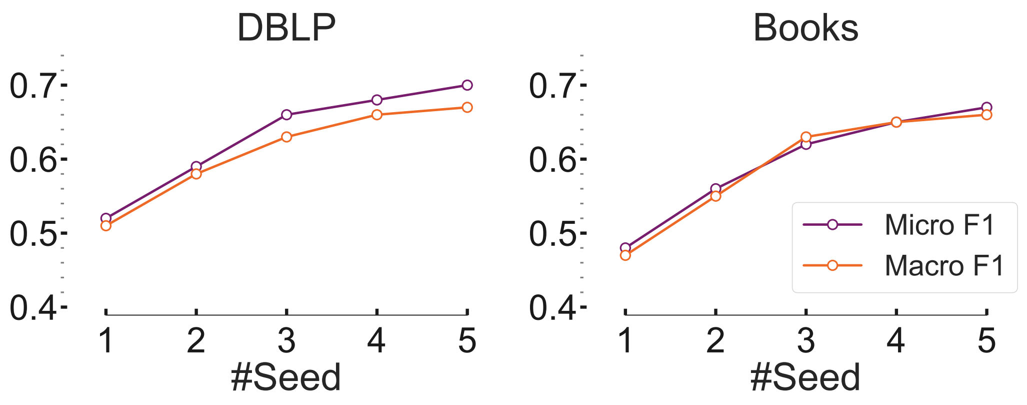


Figure 2.7. Micro- and Macro-F_1 scores w.r.t. the number of seed words.

The plots of micro-F1 score and macro-F1 score with respect to the number of iterations are shown in Figure 2.6. We observe that the performance increases initially and gets gradually converged by 6 or 7 iterations. We also observe that the expanded seed words and seed motifs have become almost unchanged. While there is some fluctuation, a reasonably large T, such as $T = 9$ or $T = 10$, is recommended.

2.6.9 Number of Seed Words

We vary the number of seed words per class and plot the performance in Figure 2.7. We observe that the performance increases as the number of seed words increase, which is generally intuitive. For reasonable performance, we observe that three seed words are sufficient.

2.6.10 Case Studies

We present case studies to showcase the effectiveness of our framework in addressing the challenges of leveraging metadata.

Leveraging Metadata Combinations

Table 2.3 shows a few samples of expanded motif instances. First, let's take a look at motif instances related to authors and publishers. We can observe that strong label-indicative authors and publishers are mined accurately. For example, *Marvel*, a widely known comics publisher, is present in the expanded publishers for *comics* genre; A classic American poet *E.*

Table 2.3. Case Study: Expanded motif instances.

Expanded motif instances of Book Graph dataset					
Class	**Author**	**Publisher**	**Author-Publisher**	**Year**	**Author-Year**
children	Z. Fraillon, K. Argent	Brighter Child, HarperCollins Children's Books	(**N. Gaiman**, Bloomsbury UK) (M. Fox, Penguin Australia)	N/A	(**N. Gaiman**, 2004) (S. Blackall, 2010)
comics	F. Teran, B. Kane	Marvel, Titan Books Ltd	(**N. Gaiman**, Marvel) (T. McFarlane, Marvel Comics)	N/A	(T. Hairsine, 2013) (A. Sinclair, 2009)
fantasy	J. Barne, S. Dubbin	DAW Books, Inc., Edge Publishing	(W. King, Titan Books Ltd) (G.J. Grant, Prime Books)	N/A	(G.J. Grant, 2012) (M. Lingen, 2012)
poetry	B. Guest, E. Dickinson	Shearsman Books, Souvenir Press	(**N. Gaiman**, MagicPress) (R. Browning, Wordsworth Editions)	1692, 1914	(E. Dickinson, 1959) (J. McCrae, 1929)

Dickinson is successfully identified as label-indicative for *poetry* genre.

Note that, the author *N. Gaiman* (in blue) who has written books in multiple genres including comic books, graphic novels, etc., is not a label-indicative author for any of these categories, because he is not exclusive to any one category, which is accurately captured by our framework. However, his works in various genres together with their respective publisher information form a unique label-indicative pattern which is reflected by the "Author-Publisher" motif pattern.

Now, adding *year* metadata into the loop, although "Year-Document" is a user-provided motif pattern, META identifies that *year* information alone is not much helpful in classification. This demonstrates the robustness of our framework when users provide some irrelevant motif patterns. However, if we combine author information with year, it then carries more accurate semantics, and we may discover that *N.Gaiman* had authored more children's books in early 2000, thus becoming highly label-indicative.

Eliminating Noise in Metadata

Table 2.4 presents the percentage of motif instances expanded out of the total motif instances following a motif pattern, for every label. One can observe that META actually prunes out many motif instances, as the final selection ratio is far less than 100%.

For the "Year-Document" motif pattern, we observe that its motif instances are only expanded for a few genres, which is generally intuitive. For example, one can see that a significant

Table 2.4. Case Study: Percentage of motif instances expanded for Book Graph dataset. **A** stands for author, **P** for publisher and **Y** for year.

	Percentage of motif instances expanded					
Label	**A**	**P**	**Y**	**A-P**	**P-Y**	**A-Y**
children	5.12	9.42	0	9.21	12.73	9.68
comics	4.91	1.33	0	9.52	1.48	14.11
fantasy	6.2	2.8	0	13.1	2.95	10.97
history	4.31	10.5	6.12	8.1	11.8	7.94
mystery	4.11	8.6	3.67	9.8	11.04	9.59
poetry	6.8	9.2	15.4	10	8.17	9.11
romance	5.6	13.5	1.47	9.6	12.28	9.19
y. adult	3.52	13.7	2.2	9.1	15.04	9.32

Table 2.5. Expanded seed words of *comics*, *history*, and *mystery* classes in Books dataset.

	Expanded seed words
Label	**Seed words**
comics	batman, superman, marvel, mary-jane, general zod
history	history, world war, world war ii, political science
mystery	serial killer, sherlock holmes, inspector lestrade

percentage of “Year-Document" motif instances expanded for *history* and *poetry*. After a closer inspection, we find that the expanded years were concentrated between the late 1800 and early 1900, thus developing an affinity for this time period.

One can also observe that the percentage of motif instances following the “Publisher-Document” motif pattern expanded varies for different labels, ranging from 1 to 13.5. This illustrates that our expansion is adaptive.

Seed words Expansion

Figure 2.8 shows the number of seed words expanded after each iteration for *comics*, *hystory*, and *mystery* classes in Books dataset. We observe that the number varies for each label because of our data-driven, adaptive thresholds, which is different for each label.

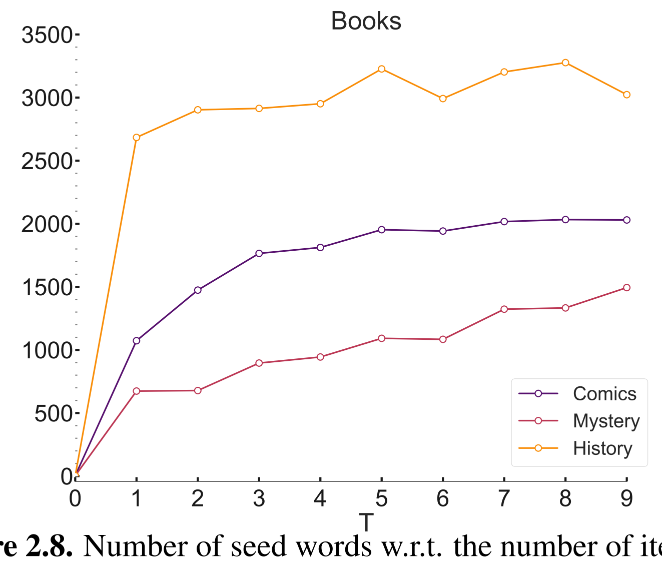


Figure 2.8. Number of seed words w.r.t. the number of iterations

One can also observe that the the number increases over iterations and gets almost stagnated at the end, indicating that the seed sets are getting refined and converged. A few examples of expanded seed words are shown in Table 2.5.

2.7 Summary

In this chapter, we propose META, a novel framework that leverages metadata information as an additional source of weak supervision and incorporates it into the classification framework. Our method organizes the text data and metadata together into a text-rich network and employs motif patterns to capture appropriate metadata combinations. Using the initial user-provided seed words and motif patterns, our method generates pseudo labels, trains classifier, and ranks and filters highly label-indicative words, motifs in a unified manner and adds them to their respective seed set. Experimental results and case studies demonstrate that our model outperforms previous methods significantly, thereby signifying the advantages of leveraging metadata as weak supervision.

In the future, we are interested in effectively integrating different forms of supervision including annotated documents. Also, we only consider positively label-indicative metadata combinations currently. There should be negatively label-indicative combinations as well which can eliminate some classes from potential labels. This is another potential direction for the extension of our method.

Chapter 2, in full, is a reprint of the material as it appears in Mekala, Dheeraj; Zhang, Xinyang; Shang, Jingbo. “Meta: Metadata-empowered weak supervision for text classification,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 8351–8361, 2020. The dissertation/thesis author was the primary investigator and author of this paper.

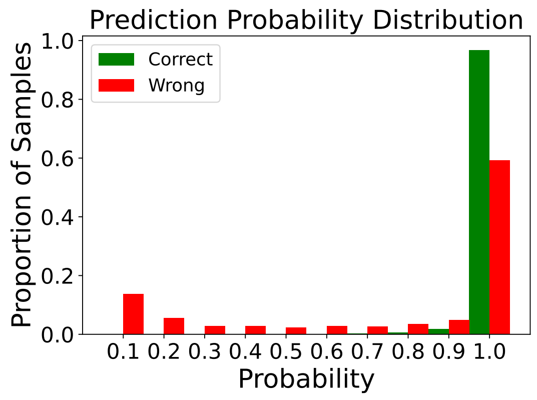
Chapter 3

Learning Order Inspired Pseudo-Label Selection for Weakly Supervised Text Classification

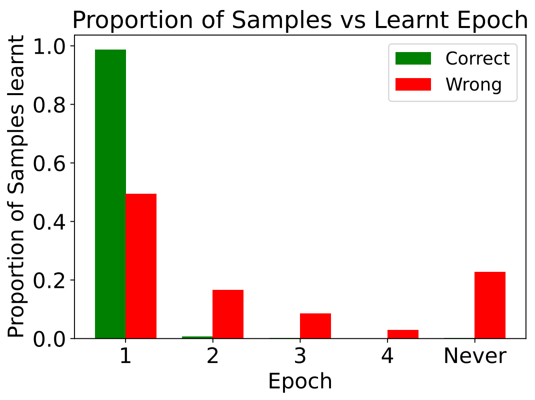
3.1 Introduction

Weakly supervised text classification methods (Agichtein & Gravano, 2000; Riloff et al., 2003; Tao et al., 2015; Meng et al., 2018; Mekala & Shang, 2020; Mekala et al., 2020, 2021) typically start with generating pseudo-labels, and train a deep neural classifier to learn the mapping between documents and classes. There is no doubt that the quality of pseudo-labels plays a fundamental role in the final classification accuracy, however, they are inevitably noisy due to their heuristic nature. Pseudo-labels are typically generated by some heuristic, for example, through string matching between the documents and user-provided seed words (Mekala & Shang, 2020). Deep neural networks (DNNs) trained on such noisy labels have a high risk of making erroneous predictions. More importantly, when self-training is employed, such error can be further amplified upon boostrapping.

To address this problem, in this paper, we study the pseudo-label selection in weakly supervised text classification, aiming to select a high quality subset of the pseudo-labeled documents (in every iteration when using self-training) that can potentially achieve a higher classification accuracy.



(a)



(b)

Figure 3.1. Distributions of correct and wrong instances using different pseudo-label selection strategies on the NYT-Coarse dataset for its initial pseudo-labels. The base classifier is BERT. (a) is based on the softmax probability of samples' pseudo-labels and (b) is based on the earliest epochs at which samples are learnt.

A straightforward solution is to first train a deep neural classifier based on the pseudo-labeled documents and then threshold the documents by the predicted probability scores corresponding to their pseudo-labels. However, DNNs usually have a poor calibration and generate overconfident predicted probability scores (Guo et al., 2017). For example, on New York Times (NYT) coarse-grained dataset, as shown in Figure 3.1(a), 60% of wrong instances in the pseudo-labeled documents have a predicted probability by BERT greater than 0.9 for their wrong pseudo-labels.

Recent studies on the memorization effects of DNNs show that they memorize easy and clean instances first, and gradually learn hard instances and eventually memorize the wrong annotations (Arpit et al., 2017; Geifman et al., 2019; Zhang et al., 2021). We have confirmed this in our experiments for different classifiers. For example, as shown in Figure 3.1(b), BERT classifier learns most of the clean instances in the first epoch and learns wrong instances across all epochs. Although it also learns good number of wrong instances in the first epoch, it is significantly less than the probability-based selection in Figure 3.1(a). Therefore, we define the learning order of a pseudo-labeled document as the epoch when it is learnt during training i.e. when the training model's prediction is the same as its given pseudo-label. Since correct

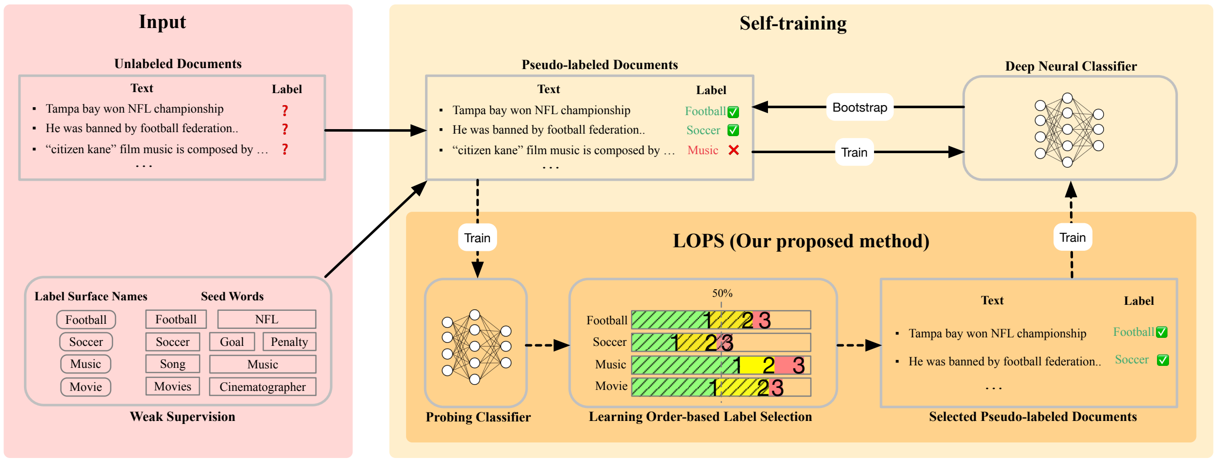


Figure 3.2. An overview of our proposed LOPS and how it plugs into self-training frameworks to replace the conventional training step. Given pseudo-labeled samples, LOPS trains a probing classifier to obtain their learning order and we stop the training when at least $\tau\%$ of samples corresponding to each class are learnt and select the learnt samples. The numbers shown are learnt epochs and the samples in the shaded part are selected. A text classifier is trained on selected pseudo-labeled documents that is further used for inference and bootstrapping.

samples are learnt first, we hypothesize that learning order-based selection will be able to filter out wrongly labeled samples.

Inspired by our observation, we propose a novel learning order inspired pseudo-label selection method LOPS, as shown in Figure 3.2. Specifically, LOPS involves training a probing classifier on pseudo-labeled data and tracking the learning order of samples. We define a sample is learnt if and only if the classifier trained on pseudo-labels gives the same argmax prediction as its pseudo-label at the end of an epoch. We stop the training when at least $\tau\%$ of samples corresponding to each class are learnt and select all the learnt samples. Then, we train a text classifier on these selected pseudo-labeled documents that is further used for inference. We empirically show that LOPS can boost the accuracy of various weakly supervised text classification methods and it is much more effective and stable than probability score-based selections.

Our contributions are summarized as follows:

- We propose a novel pseudo-label selection method LOPS that takes learning order of samples into consideration.

- We show that selection based on learning order is much stable and effective than selection based on probability scores.
- Extensive experiments and case studies on real-world datasets with different classifiers and weakly supervised text classification methods demonstrate significant performance gains upon using LOPS. It can be viewed as a solid performance-boost plug-in for weak supervision.

Reproducibility. We release the code and datasets on Github[1].

3.2 Related Work

Pseudo-Labels in Weakly Supervised Text Classification. Since the weakly supervised text classification methods lack gold annotations, pseudo-labeling has been a common phenomenon to generate initial supervision. Pseudo-labeling depends on the type of weak supervision. Mekala & Shang (2020) and Mekala et al. (2020) have a few label-indicative seed words as supervision and they generate pseudo-labels using string-matching where a document is assigned a label whose aggregated term frequency of seed words is maximum. (Meng et al., 2018) generates pseudo-documents using the seed information corresponding to a label. (Wang et al., 2021) takes only label names as supervision and generates class-oriented document representations, and cluster them to create a pseudo-training set. Under the same scenario, (Mekala et al., 2021) consider samples that exclusively contain the label surface name as its respective weak supervision. In (Karamanolakis et al., 2021), pseudo-labels are created from the predictions of a trained neural network. (Arachie & Huang, 2021) combines different weak signals to produce soft labels.

Label Selection. There are different lines of work aiming to select true-labeled examples from a noisy training set. One line of work involves training multiple networks to guide the learning process. Along this direction, (Malach & Shalev-Shwartz, 2017) maintains two DNNs and update them based on their disagreement. (Jiang et al., 2018b) learns another neural network that provides data-driven curriculum. (Han et al., 2018; Yu et al., 2019) use co-training where

[1]https://github.com/dheeraj7596/LOPS

Table 3.1. Noise ratios of different pseudo-label heuristics on NYT-Fine dataset.

Pseudo-label Heuristic	**Noise Ratio**
vMF distribution modeling (Meng et al., 2018)	46.17%
String-Match (Mekala et al., 2020)	31.80%
Contextualized String-Match (Mekala & Shang, 2020)	31.24%
Exclusive String-Match (Mekala et al., 2021)	52.13%
Clustering (Wang et al., 2021)	15.64%

they select instances based on small loss criteria and cross-train two networks simultaneously. (Huang et al., 2019a) considers the training loss as the metric to filter out noise. (Swayamdipta et al., 2020) uses model's confidence and its variability across epochs to identify wrongly labeled samples. Another line of work learns weights for the training data. Along this line, (Ren et al., 2018) propose a meta-learning algorithm that learns weights corresponding to training examples based on their gradient directions. (Fang et al., 2020) learns dynamic importance weighting that iterates between weight estimation and weighted classification. Recently, (Rizve et al., 2021) propose utilizing uncertainty to perform label selection.

Training dynamics. In deep learning regime, models with large capacity are typically more robust to outliers. Nevertheless, data examples can still exhibit diverse levels of difficulties. Arpit et al. (2017) finds that data examples are not learned equally when injecting noisy data into training. Easy examples are often learned first. Hacohen et al. (2020) further shows such order of learning examples is shared by different random initializations and neural architectures. Toneva et al. (2019) shows that certain examples are forgotten frequently during training, which means that they can be first classified correctly, then incorrectly. Model performance can be largely maintained when removing those least forgettable examples from training.

3.3 Problem and Motivation

Weakly supervised classification refers to the problem with inputs (1) a set of unlabeled text documents $\mathscr{S} = \{x\}$, where $x \in \mathscr{X}$. (2) and M target labels $\mathscr{C} = \{1, \ldots, M\}$. Our goal is to find a labeling function $f : \mathscr{X} \rightarrow \mathscr{C}$ that maps every document x to its true label. Here we denote y^* as the *unknown* true label of a document x. To cold start the classification of unlabeled documents, a source of weak supervision has to be introduced, which can come from various sources such as label surface names (Wang et al., 2021), label-indicative seed words (Mekala & Shang, 2020), or rules (Karamanolakis et al., 2021). Given a "weak" labeling function $w : \mathscr{X} \to \mathscr{C}$, pseudo-labels are then generated on a subset of the unlabeled documents, which yields a labeled subset $\mathscr{D} = \{(x, w(x))\}$. For convenience, we denote $\mathscr{D}[j]$ to be the set of all documents that are pseudo-labeled as class j in $\mathscr{D}$, namely $\mathscr{D}[j] = \{(x, w(x)) \in \mathscr{D} | w(x) = j\}$.

Pseudo-labels are noisy due to their heuristic nature. For example, on the NYT fine-grained dataset, we generate pseudo-labels using five different strategies (Meng et al., 2018; Mekala & Shang, 2020; Mekala et al., 2020, 2021; Wang et al., 2021) and compute their noise ratios. As expected, no strategy is perfect and all of them generate noisy labels, ranging from 15% to 50% (see Table 3.1).

When a classifier is trained on such noisy training data, it can make some high confident erroneous predictions. And, upon bootstrapping the classifier on unlabeled data, it has a snowball effect where such high confident erroneous predictions are added to the training data, and thus corrupting it more. As this process repeats for a few iterations, it adds more noise and significantly affects the final performance. Therefore, identifying and selecting the correctly labeled samples is necessary and has a huge potential for a boost in performance. Note that, if the labels are not selected carefully, it could instead hurt the performance.

Our pseudo-label selection problem. The weak supervision is likely to generate a noisy labeled set, which means $w(x) \neq y^*$ for some documents x. We denote $\mathscr{D}_{\checkmark}$ as the set of correctly labeled documents and $\mathscr{D}_{\times} = \mathscr{D} \setminus \mathscr{D}_{\checkmark}$ as the set of wrongly labeled documents, where $\mathscr{D}_{\checkmark} = \{(x, w(x)) | w(x) = y^*\}$. The problem of pseudo-label selection is thus to identify $\mathscr{D}_{\checkmark}$.

Note that pseudo-label selection is conceptually related to failure prediction (Hecker et al., 2018; Jiang et al., 2018a; Corbière et al., 2019) and out-of-distribution detection (Hendrycks & Gimpel, 2017; Devries & Taylor, 2018; Liang et al., 2018; Lee et al., 2018). However, the major difference here is for pseudo-label selection we have to detect wrong annotations in the training phase instead of inference phase.

3.4 Our LOPS Framework

In this section, we explain our framework LOPS in detail. First, we give an overview of confidence function-based pseudo-label selection and discuss probability score as confidence function. Then, we explain learning order as confidence function. Finally, we show our algorithm that performs selection based on learning order.

3.4.1 Overview: Confidence function-based Pseudo-label Selection

In this section, we briefly introduce confidence function and discuss commonly-used probability score as confidence function.

Confidence function $\kappa : \mathscr{X} \times \mathscr{C} \to [0,1]$, assigns a value to each labeled document, which represents our confidence of its pseudo-label being correct. Then, we can perform the selection by choosing a threshold γ on confidence function. We denote the set of labeled documents selected based on κ and γ as $\hat{\mathscr{D}}_{\checkmark}(\kappa,\gamma)$, namely

$$\hat{\mathscr{D}}_{\checkmark}(\kappa,\gamma) = \{(x, w(x)) \in \mathscr{D} \mid \kappa(x, w(x)) > \gamma\}$$

An optimal confidence function κ^* should be able to perfectly distinguish the correctly labeled documents from wrongly labeled ones, namely there exists a threshold γ^* such that $\hat{\mathscr{D}}_{\checkmark}(\kappa^*,\gamma^*) = \mathscr{D}_{\checkmark}$.

Probability score as confidence function. One commonly-used intuitive confidence function for pseudo-label selection is the model's prediction probability scores corresponding to the pseudo-labels. Probability scores have been used as confidence functions to select samples for bootstrapping (Meng et al., 2018, 2019; Mekala & Shang, 2020). Specifically, let $\mathbf{f} : \mathscr{X} \to [0,1]^{|\mathscr{C}|}$ be a probabilistic classifier trained on pseudo-labeled documents and $\mathbf{f}(x)[j]$ represents the predicted probability of document x belonging to class j, $\mathbf{f}(x)[w(x)]$ is used as the confidence function. However, due to the poor calibration of DNNs (Guo et al., 2017), probability scores of wrongly labeled documents are usually high. As a result, it might be difficult to distinguish correctly- and wrongly-labeled documents based on probability scores.

3.4.2 LOPS: Learning Order as Confidence Function

Learning order. Learning order of a pseudo-labeled document is the epoch when it is learnt during training, or more specifically when its label predicted by the model matches its given pseudo-label. Recent studies show that a DNN learns clean samples first and then gradually memorizes the noisy samples (Arpit et al., 2017). We thus hypothesize that learning order can reflect the probability of wrong pseudo-label in terms of ranking.

We now utilize learning order to define a confidence function. Specifically, let $\mathbf{f}^t(\cdot)$ be the classifier being trained at epoch t, and T as the total number of epochs, the learning order of document x can be defined as

$$\eta(x, w(x)) = 1 - \frac{1}{T}\min\{t \mid \arg\max_j \mathbf{f}^t(x)[j] = w(x)\}, \tag{3.1}$$

where $t \in \{1, \ldots, T\}$. Here we have negated and scaled the learning order to be complied with the convention of confidence function i.e. higher confidence implies higher probability of a correct label. We calculate the learning order at the granularity of epoch because the model would have seen all the training data by the end of an epoch, and hence, the learning order computed would be fair for all documents. In case when the epoch number is not sufficient to distinguish the documents, one can increase the granularity of the learning order, for example, the batch number at which the document is learnt. Granularity higher than the epoch incurs extra training cost as a document will be examined more than once in each epoch.

Algorithm 2: LOPS Method

```
Input: A set of documents $\mathscr{D}$ pseudo-labeled by $w$, Probing Classifier $\mathbf{f}$.
Output: Selected documents $\hat{\mathscr{D}}_{\checkmark}$
for epoch $t = 1, 2, \ldots, T$ do
    Train $\mathbf{f}$ on $\mathscr{D}$
    for $(x, w(x)) \in \mathscr{D}$ do
        if $\arg\max_j \mathbf{f}(x)[j] = w(x)$ then
            if $|\hat{\mathscr{D}}_{\checkmark}[w(x)]| / |\mathscr{D}[w(x)]| < \tau\%$ then
                $\hat{\mathscr{D}}_{\checkmark} = \hat{\mathscr{D}}_{\checkmark} \cup \{(x, w(x))\}$
    if $|\hat{\mathscr{D}}_{\checkmark}[j]| / |\mathscr{D}[j]| \geq \tau\%$ for all $j$ then
        Break
Return $\hat{\mathscr{D}}_{\checkmark}$
```

Algorithm 3: Self-training with LOPS

```
Input: Unlabeled data $\mathscr{D}$, Classifier $C$, Weak Supervision $w$.
Output: Prediction labels $predLabs$
$\hat{\mathscr{D}}$ = Generate Pseudo-labels for $\mathscr{D}$, $w$
for iteration $it = 1, 2, \ldots, n_{its}$ do
    $\mathscr{D}_{sel}$ = LOPS ($\hat{\mathscr{D}}$, $C$)
    Train $C$ on $\mathscr{D}_{sel}$
    $predLabs$, $predProbs$ = Predict($C$, $\mathscr{D}$)
    $\hat{\mathscr{D}}$ = $\hat{\mathscr{D}} \cup \{x \mid predProbs(x) > \delta\}$
Return $predLabs$
```

3.4.3 LOPS: Putting it all together

Motivated by previous analyses, we utilize learning order to select pseudo-labels. We train a probing classifier on all pseudo-labeled documents and track their first learnt epoch during training. The confidence function can then be calculated based on Equation (3.1). Finally, we rank the documents based on their confidence and select the top-$\tau\%$ for each label independently.

To maximize the efficiency of LOPS, we utilize the fact that the top-ranked documents are learned earlier, and conduct the confidence calculation and pseudo-label selection simultaneously during training. Specifically, for each label, a document is selected once it is learnt, until the fraction of selected documents $\geq \tau\%$ exceeds $\tau\%$ in this label. Whenever the fractions of selected documents exceeds $\tau\%$ for all labels, we stop the training. The pseudo-code is shown in Algorithm 2. Note that LOPS can be plugged to any self-training based weakly-supervised classification framework as shown in Algorithm 3.

Table 3.2. Dataset statistics.

Dataset	# Docs	# labels	Noise Ratio(%)
NYT-Coarse	13,081	5	11.47
NYT-Fine	13,081	26	31.80
20News-Coarse	17,871	5	12.50
20News-Fine	17,871	17	25.67
AGNews	120,000	4	16.26
Books	33,594	8	37.32

3.5 Experiments

We evaluate our label selection method based on end-to-end classification performance using different state-of-the-art classifiers and weakly supervised text classification frameworks. And also, we evaluate learning order as a confidence function and provide a comparison with probability score as confidence function.

3.5.1 Datasets

We experiment on four datasets: New York Times (**NYT**), 20 Newsgroups (**20News**), **AGNews** (Zhang et al., 2015), **Books** (Wan & McAuley, 2018; Wan et al., 2019). NYT and 20News datasets also have fine-grained labels which are also used for evaluation. Initial pseudo-labels are generated using String-Match (Mekala & Shang, 2020). The dataset statistics and corresponding noise ratios of initial pseudo-labels are provided in Table 3.2 and more details are provided in Appendix A.1.

3.5.2 Compared Methods

We compare with several label selection methods mentioned below:

- **O2U-Net:** (Huang et al., 2019a) trains a classifier cyclically to make its status transfer from overfitting to underfitting and records losses of each sample. They consider the normalized loss as the metric to filter out the noise.
- **MC-Dropout:** (Mukherjee & Awadallah, 2020) performs pseudo-label selection based on uncertainty estimates computed using probability scores.

Table 3.3. Evaluation results on six datasets using different combinations of classifiers and pseudo-label selection methods. Initial pseudo-labels are generated using String-Match. Micro- and Macro-F1 scores are used as evaluation metrics. Each experiment is repeated with three random seeds, mean and their respective standard deviations are presented in percentages. For a fair comparison, we consider the same number of samples for all baselines as LOPS in each iteration. Abnormally high standard deviations are highlighted in *blue* and low performances are highlighted in *red*. LOPS outperforming *Standard* is made bold and baselines performing better than our method are made bold. Statistical significance results are in Appendix A.5.

		Coarse-grained Datasets								Fine-grained Datasets			
		NYT-Coarse		20News-Coarse		AGNews		Books		NYT-Fine		20News-Fine	
Classifier	Method	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
BERT	Standard	90.1(0.17)	80.3(0.91)	77.3(0.27)	76.4(0.76)	75.4(0.64)	75.4(0.47)	55.7(0.54)	57.9(0.82)	77.2(0.36)	71.6(0.43)	70.0(0.30)	69.6(0.25)
	LOPS	**94.6(0.36)**	**88.4(0.50)**	**81.7(1.00)**	**80.7(0.43)**	**79.5(0.86)**	**79.5(0.58)**	**57.7(0.87)**	**59.5(0.46)**	**84.3(0.54)**	**81.6(0.34)**	**73.8(0.61)**	**72.7(1.00)**
	MC-Dropout	89.3(0.41)	79.3(0.45)	80.7(0.17)	77.7(0.24)	75.8(0.34)	75.0(0.41)	55.1(0.15)	56.7(0.61)	72.1(0.74)	69.0(0.41)	68.0(0.21)	68.7(0.26)
	Entropy	91.2(0.41)	83.1(0.47)	80.4(0.23)	78.0(0.54)	**80.4(0.47)**	**80.0(0.42)**	55.2(0.74)	56.7(0.42)	43.4(9.84)	18.1(6.98)	64.3(0.74)	63.6(0.83)
	O2U-Net	92.9(0.41)	85.9(0.69)	80.9(0.28)	78.5(0.19)	79.8(0.47)	79.8(0.53)	55.8(0.27)	56.8(0.36)	14.7(10.24)	8.70(7.31)	71.1(0.36)	71.2(0.75)
	Random	90.3(0.47)	80.9(0.47)	79.0(1.00)	76.8(1.50)	76.3(0.35)	76.3(0.65)	56.1(0.18)	58.2(0.35)	78.4(0.94)	71.7(0.47)	71.4(0.50)	70.6(1.00)
	Probability	92.3(1.50)	85.1(2.00)	78.6(2.50)	77.5(3.00)	77.4(1.25)	77.6(1.34)	54.3(1.12)	56.5(1.43)	46.6(2.50)	22.3(0.50)	47.8(23.50)	47.9(23.50)
	Stability	93.3(0.50)	86.5(0.50)	76.7(5.00)	75.4(5.00)	79.3(0.75)	79.5(0.35)	55.0(0.43)	57.0(0.19)	48.1(29.50)	35.5(33.50)	73.5(0.50)	72.5(1.00)
	OptimalFilter	98.3(0.27)	96.4(0.37)	94.7(0.37)	94.9(0.61)	89.4(0.46)	89.3(0.76)	76.2(0.21)	76.7(0.19)	97.4(0.71)	92.2(0.62)	87.6(0.37)	86.5(0.36)
XLNet	Standard	89.2(0.74)	80.1(0.64)	77.6(0.39)	75.4(0.68)	72.7(0.97)	72.4(0.53)	57.6(0.31)	58.7(0.46)	77.4(0.34)	71.3(0.75)	60.7(0.74)	66.5(0.61)
	LOPS	**89.5(0.17)**	**81.4(0.90)**	**82.5(0.50)**	**81.2(0.20)**	**77.7(0.57)**	**77.7(0.54)**	**58.5(0.65)**	**59.4(0.67)**	**80.7(0.22)**	**77.4(0.83)**	**70.6(0.31)**	**70.4(0.27)**
	MC-Dropout	88.5(0.41)	80.4(0.38)	77.1(0.28)	73.2(0.53)	74.7(0.71)	73.2(0.36)	56.4(0.41)	58.0(0.74)	74.9(0.96)	68.9(0.84)	66.9(0.45)	68.5(0.62)
	Entropy	**92.4(0.42)**	**85.4(0.51)**	78.2(0.36)	74.4(0.45)	72.9(0.67)	72.0(0.51)	54.5(0.74)	56.6(0.65)	77.9(0.67)	70.7(0.38)	68.8(0.65)	69.5(0.74)
	O2U-Net	92.2(0.37)	84.6(0.24)	80.5(0.93)	77.4(0.57)	71.6(0.69)	68.8(0.61)	58.1(0.17)	**59.9(0.52)**	79.6(0.47)	76.8(0.59)	67.2(0.64)	69.0(0.26)
	Random	90.7(0.03)	80.5(0.51)	78.6(0.50)	75.4(1.00)	67.5(0.22)	67.4(0.63)	57.5(0.43)	58.3(0.45)	76.6(0.94)	72.7(0.70)	67.3(0.49)	67.2(0.32)
	Probability	91.3(0.29)	83.4(0.50)	77.4(1.00)	75.2(0.30)	70.1(1.09)	70.4(1.14)	54.6(1.42)	56.3(1.26)	38.2(6.50)	36.5(1.00)	69.5(0.82)	69.2(0.12)
	Stability	91.4(1.00)	82.3(1.50)	79.7(1.50)	77.6(1.50)	74.3(1.10)	74.5(0.87)	56.3(0.88)	58.1(0.97)	79.5(0.50)	76.3(1.10)	68.5(0.49)	68.4(1.00)
	OptimalFilter	98.3(0.12)	96.5(0.21)	94.5(0.23)	94.4(0.29)	89.3(0.28)	89.7(0.39)	76.4(0.44)	76.3(0.43)	97.4(0.32)	93.6(0.38)	86.6(0.43)	86.4(0.35)
GPT-2	Standard	91.1(0.24)	82.3(0.28)	78.4(0.26)	76.3(0.38)	61.3(0.28)	61.2(0.43)	51.6(0.41)	53.3(0.37)	76.2(0.41)	69.5(0.38)	70.5(0.46)	70.4(0.38)
	LOPS	**95.2(0.49)**	**89.1(0.51)**	**82.5(0.57)**	**80.3(0.63)**	**75.7(0.52)**	**75.3(0.31)**	**56.8(0.89)**	**58.6(0.63)**	**80.4(0.09)**	**76.3(0.21)**	**70.6(0.76)**	**70.5(0.48)**
	MC-Dropout	89.2(0.14)	79.8(0.51)	80.2(0.63)	77.1(0.57)	65.5(0.34)	65.1(0.94)	49.5(0.74)	51.5(0.54)	74.1(0.62)	68.2(0.21)	70.4(0.47)	70.8(0.65)
	Entropy	93.1(0.32)	85.9(0.36)	80.8(0.65)	77.9(0.84)	65.4(0.85)	65.3(0.54)	54.3(0.32)	55.5(0.47)	77.4(0.42)	75.3(0.65)	69.1(0.62)	69.6(0.21)
	O2U-Net	93.8(0.89)	87.5(0.24)	81.2(0.76)	77.9(0.37)	72.0(0.38)	70.7(0.75)	55.1(0.27)	57.2(0.67)	80.2(0.41)	**79.4(0.58)**	70.3(0.24)	**71.4(0.16)**
	Random	90.2(0.42)	80.2(0.56)	79.7(0.46)	78.4(0.32)	68.2(0.18)	68.1(0.19)	53.4(0.46)	55.3(0.42)	77.5(0.52)	70.4(1.02)	69.4(0.21)	69.3(0.29)
	Probability	93.3(1.04)	85.5(1.13)	80.4(1.49)	78.5(1.50)	66.2(0.69)	66.6(0.89)	51.7(1.11)	54.5(1.09)	76.7(0.57)	71.3(0.69)	69.4(1.21)	69.3(1.18)
	Stability	94.4(0.56)	88.6(0.59)	81.4(1.02)	78.6(1.50)	72.4(0.58)	72.3(0.53)	53.6(1.02)	55.3(1.13)	79.4(0.62)	75.3(0.65)	**70.6(0.68)**	70.4(0.63)
	OptimalFilter	98.3(0.24)	96.2(0.21)	94.2(0.23)	93.3(0.27)	88.7(0.26)	88.4(0.28)	72.3(0.19)	73.7(0.22)	97.3(0.18)	92.4(0.19)	86.1(0.35)	85.5(0.38)

- **Entropy:** is similar to MC-Dropout, however uses entropy to compute uncertainty scores.
- **Probability:** We sort the prediction probabilities corresponding to pseudo-labels in descending order and select the same number of samples as LOPS in each iteration of bootstrapping.
- **Random:** We randomly select the same number of samples as LOPS in each iteration of bootstrapping. To avoid skewed selection, we sample in a stratified fashion based on class labels.
- **Learning Stability (stability):** (Dong et al., 2021) introduced a metric to measure the data quality based on the frequency of events that an example is predicted correctly throughout the

Table 3.4. Evaluation results of weakly supervised text classification frameworks with LOPS. This demonstrates that LOPS can be easily plugged in and improves the performance.

	NYT-Coarse		NYT-Fine		20News-Coarse		20News-Fine		AGNews		Books	
Method	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
						ConWea						
Standard	93.1	87.2	87.4	77.4	74.3	74.6	68.7	68.7	73.4	73.4	52.3	52.6
LOPS	**94.2**	**90.1**	**87.5**	**78.6**	**79.7**	**78.4**	**70.4**	**70.6**	**79.2**	**79.2**	**57.5**	**58.7**
						X-Class						
Standard	**96.3**	**93.3**	86.6	**74.7**	58.2	61.1	70.4	70.4	82.4	82.3	53.6	54.2
LOPS	96.2	**93.3**	**86.8**	73.8	**60.7**	**62.3**	**71.2**	**71.2**	**83.6**	**82.7**	**54.2**	**56.3**
						WeSTClass						
Standard	92.3	86.0	67.1	60.4	53.2	49.4	54.9	54.9	80.4	80.1	49.7	48.1
LOPS	**93.4**	**88.1**	**68.4**	**63.8**	**53.3**	**51.5**	**61.1**	**60.5**	**81.4**	**81.3**	**51.2**	**49.8**
						LOTClass						
Standard	**70.1**	**30.3**	**5.3**	**4.1**	**47.0**	**35.0**	**12.3**	**10.6**	84.9	84.7	**19.9**	**16.1**
LOPS	**70.1**	**30.3**	3.5	2.9	45.7	32.6	7.8	4.1	**86.2**	**86.1**	15.8	10.3

training. We sort the samples based on learning stability in descending order i.e. most stable to least stable and select the same number of samples as LOPS in each iteration of bootstrapping.

To perform controlled experiments with a fair comparison, we consider the same number of samples as LOPS in each iteration for all above baselines because we cannot tune individual thresholds for each dataset since there is no human-annotated data under the weakly supervised setting and one fixed threshold for all datasets doesn't work as distribution of prediction probability varies across datasets.

We also present experimental results without any label selection in addition to the probability threshold δ while bootstrapping (denoted by ***Standard***) as lower bound and with all the wrongly annotated samples removed as ***OptimalFilter***.

3.5.3 Experimental Settings

For all our experiments, we consider seed words used in (Mekala & Shang, 2020; Wang et al., 2021) as weak supervision and generate initial pseudo-labels using String-Match (Mekala et al., 2020) unless specified. The average number of seeds are 4 per class. We experiment on three state-of-the-art text classifiers: (1) **BERT** (`bert-base-uncased`) (Devlin et al., 2019), (2) **XLNet** (`xlnet-base-cased`) (Yang et al., 2019), and (3) **GPT-2** (Radford et al., 2019). We follow the same self-training method for all classifiers that starts with generating pseudo-labels, training a classifier on pseudo-labeled data, and bootstrap it on unlabelled data by adding samples whose prediction probabilities are greater than δ. Following (Mekala & Shang, 2020), we assume that weak supervision $\mathscr{W}$ is of reasonable quality i.e. majority of pseudo-labels are good. Therefore, we set τ to 50%. While training the classifiers, we fine-tune BERT, XLNet, GPT-2 for 4 epochs. We bootstrap all the classifiers for 5 iterations with the probability threshold δ as 0.6. We also experiment on state-of-the-art weakly supervised text classification methods: **ConWea** (Mekala & Shang, 2020), **X-Class** (Wang et al., 2021), **WeSTClass** (Meng et al., 2018), and **LOTClass** (Meng et al., 2020). Three of them are self-training-based methods and more details are mentioned in Appendix A.2.

3.5.4 End-to-End Classification Performance

Results: Different Classifiers

We summarize the evaluation results with different combinations of classifiers and selection methods in Table 3.3. All experiments are run on three random seeds and mean, standard deviations are reported.

As shown in Table 3.3, upon plugging our proposed method LOPS, we observe a significant boost in performance consistently over ***Standard*** with all the classifiers. We observe that LOPS always outperforms random selection which shows that the selection in LOPS is strategic and principled. LOPS performs better than probability and stability based selection methods in most of the cases. This shows that LOPS is very effective in removing wrongly labeled and preserving correctly labeled samples. LOPS also performs better than O2U-Net (Huang et al., 2019a) and MC-Dropout (Mukherjee & Awadallah, 2020) in most of the datasets demonstrating the effectiveness of learning order as confidence function.

We also observe a significant boost in performance over ***Standard*** with all the classifiers in the case of fine-grained datasets as well. In some cases like BERT on NYT-Fine, the improvement is as high as 7 points on micro-f1 and 10 points on macro-f1. We observe abnormally low performances of probability and stability based selection methods in some scenarios (highlighted in *red*). This is because the number of noisy labels are more in fine-grained datasets and gets amplified with self-training and resulting in high noise. Moreover, we also observe that probability and stability based selections are biased towards majority labels and select wrong majority labels over correct minority labels. For example, the precision of pseudo-labels belonging to minority classes like *cosmos*, *gun control*, and *abortion* in NYT-Fine before selection is 100% and it selected almost none of these whereas it selected 700 wrong documents belonging to a majority labels like, *international business*. Although stratified selection can be employed to address this problem, this ends up having a same threshold and selecting a fixed ratio of samples for every dataset, which might not be optimal for every dataset.

We have to note unusually high standard deviation for probability and stability in some cases (highlighted in *blue*). This demonstrates that these selection methods are unstable. LOPS is comparatively more stable and its effectiveness is largely due to its invariance. Although these methods outperform LOPS in a few cases, their unstable nature makes them unreliable. Therefore, we believe LOPS is superior than compared methods.

Results: Different Weakly-Supervised Text Classification Methods

We summarize the evaluation results with different weakly supervised methods in Table 3.4. The results demonstrate that LOPS improves the performance of ConWea and WeST-Class significantly on all datasets and X-Class sometimes. Note that, X-Class sets a confidence threshold and selects only top-50% instances, which provides a hidden advantage and LOPS improves the performance on top of it for some datasets. We have to note the significantly low performance of LOTClass. It is observed that LOTClass requires a wide variety of contexts of label surface names from the input corpus to generate high quality category vocabulary,

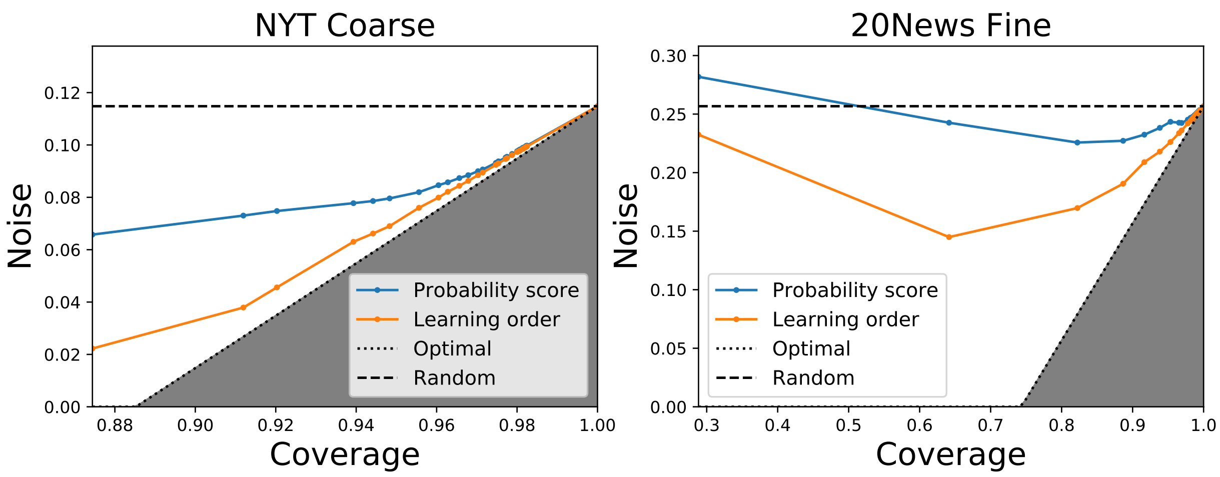
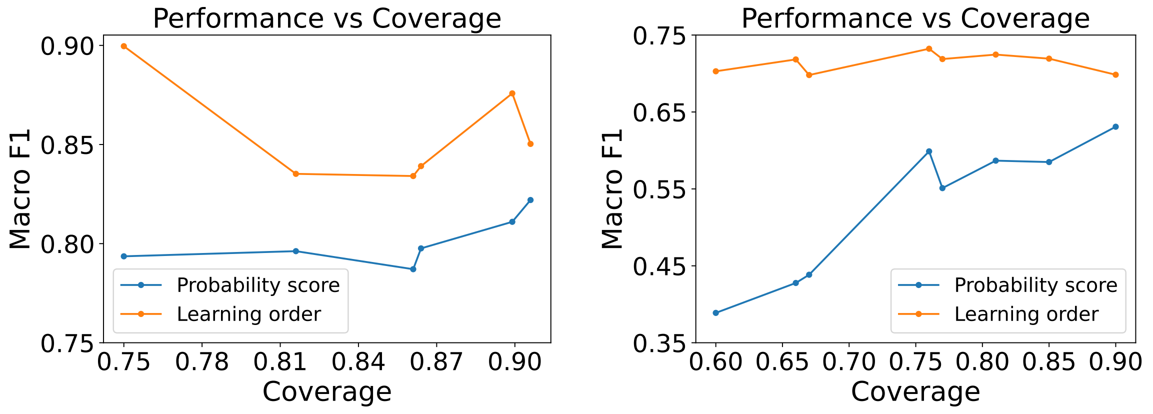


Figure 3.3. NC-curves of learning order and probability score with BERT as the classifier.



(a) NYT Coarse (b) 20News Fine

Figure 3.4. Macro-F$_1$ vs Coverage on NYT-Coarse & 20News-Fine using BERT with LOPS and Probability score based selection.

which plays a key role in performance (Wang et al., 2021). The performance is comparitively worse in fine-grained classes than coarse-grained classes because LOTClass assumes that the replacements of label surface names are indicative of its respective label. However, this might not be a valid assumption for fine-grained classes (Mekala et al., 2021). Among the datasets we experimented on, these requirements are satisfied only by AGNews dataset where there are many documents(120000) classified broadly into 4 categories and we observe a performance boost using LOPS on this dataset. Due to poor quality of pseudo-labels for other datasets, there is no increment in performance with LOPS.

3.5.5 Learning Order vs Probability Score: Evaluating Confidence Functions

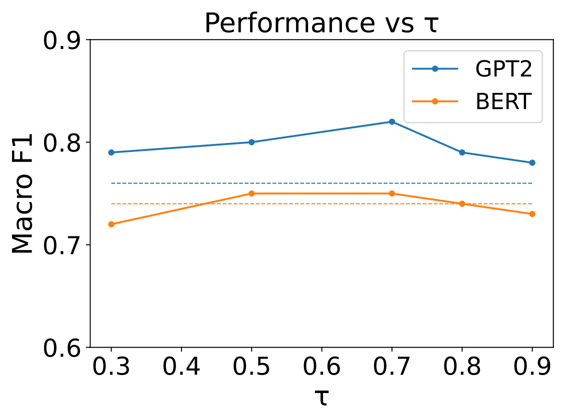
In this section, we define evaluating a confidence function and compare learning order and probability score as confidence functions.

Evaluation of a confidence function. Ideally, there exists a threshold for a given confidence function that perfectly distinguishes correctly and wrongly labeled samples. However, in practice, confidence functions may not suffice such ideal condition. There always exists a trade-off between *noise* $\varepsilon(\kappa,\gamma)$ and *coverage* $\phi(\kappa,\gamma)$, defined as:

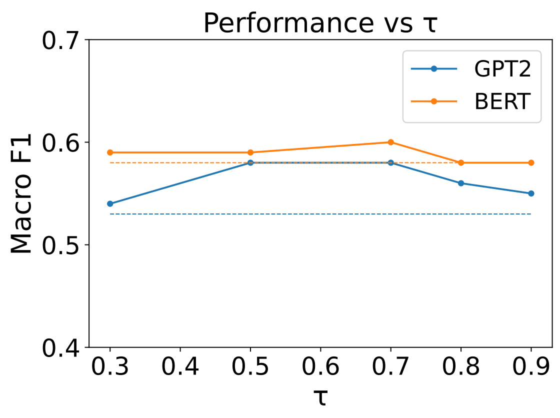
$$\varepsilon(\kappa,\gamma) = \frac{|\hat{\mathscr{D}}_{\checkmark}(\kappa,\gamma) \cap \mathscr{D}_{\times}|}{|\hat{\mathscr{D}}_{\checkmark}(\kappa,\gamma)|}, \quad \phi(\kappa,\gamma) \quad = \frac{|\hat{\mathscr{D}}_{\checkmark}(\kappa,\gamma)|}{|\mathscr{D}|}.$$

The coverage is the fraction of labeled documents being selected and the noise is the fraction of wrongly labeled documents within selected documents. A small threshold leads to high coverage i.e. most labeled documents will be selected, thus being more noisy. And a high threshold leads to an opposite situation. Therefore, to evaluate a confidence function, we plot noise and coverage at various thresholds, which we refer as the *noise-coverage curve* (NC-curve) and compute the *area under the noise-coverage curve* (AUNC). As shown in figure 3.3, an optimal confidence function selects wrongly labeled documents only after selecting all the correctly labeled documents, hence generates a NC-curve in the shape of a rectifier, namely $\varepsilon = \max(0, \phi - |\mathscr{D}_{\checkmark}|/|\mathscr{D}|)$. A random confidence function always selects the same fraction of wrongly labeled documents, hence an NC-curve with a constant value. An ideal confidence function should minimize AUNC.

Learning Order vs Probability Score. We plot NC-curves of learning order and probability scores in Figure 3.3 with BERT classifier on NYT-Coarse, 20News-Fine datasets. To isolate them from the effects of bootstrapping, we don't perform any bootstrapping. We also plot the end-to-end performance vs coverage in Figure 3.4. From Figure 3.3, we observe that learning order has significantly smaller AUNC compared to the probability score. In some datasets such as NYT-Coarse, it even approaches optimal confidence function. In fine-grained datasets like

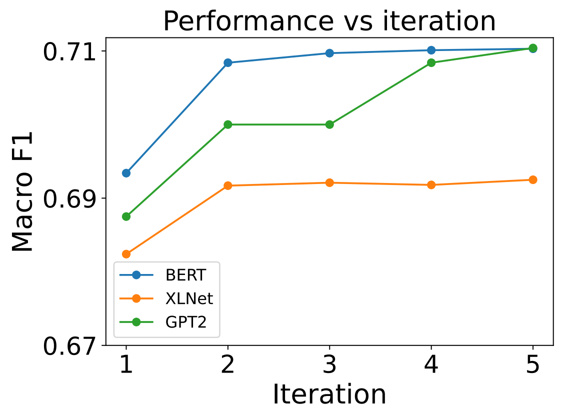


(a) 20News Coarse

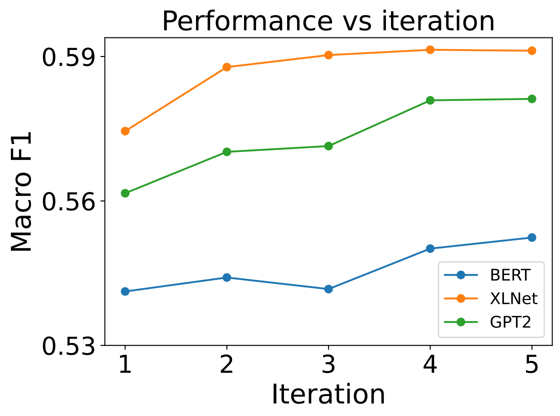


(b) Books

Figure 3.5. Macro-F$_1$ vs τ on 20News-Coarse & Books using GPT2 and BERT with LOPS. The dashed lines represent performance with no label selection.



(a) 20News Fine



(b) Books

Figure 3.6. Macro-F$_1$ vs iteration on 20News-Fine & Books using BERT, XLNet, GPT2 with LOPS.

20News-Fine, the calibration is so poor that the probability score is even worse than random, which explains poor empirical results of Probability-based selection on fine-grained datasets. From Figure 3.4, we observe that the performance with LOPS is significantly better and more stable than Probability.

3.5.6 Performance vs τ

To study the effect of τ on performance, we plot macro-f1 vs τ on 20News-coarse and Books datasets using GPT2 and BERT classifiers, shown in Figure 3.5. We observe that the performance increases initially and gradually drops down at higher τ values. The lower τ values imply being highly selective and thus the few number of selected samples are not enough for the model to generalize. The higher τ values imply poor selection with many noisy labels, making the performance to drop. From the plot, we can observe that the performance is robust for middle τ values i.e. $50-70\%$.

3.5.7 Performance vs iteration

The plot of performance vs the number of iteration of bootstrapping is shown in Figure 3.6. We observe that the macro f1 increases initially and gradually converges at the later iterations.

3.6 Limitations

Since we select 50% of the samples based on learning order, our method requires the absolute number of pseudo-labeled samples to be high enough so that the final classifier has significant number of selected samples to learn and generalize on. For example, we experimented on a subset of 2613 samples from 20news-fine dataset with noise rate 20%. With LOPS, the macro f1 is 68.3% and without any selection the macro-f1 is 70.1%. We attribute this performance drop to the lack of generalization using the few selected samples from LOPS. Since in real-life scenario, obtaining noisy annotations is cheaper, we believe this limitation can be addressed comfortably.

3.7 Summary

In this chapter, we proposed LOPS, a novel learning order inspired pseudo-label selection method. Our method is inspired from recent studies on memorization effects that showed that clean samples are learnt first and then wrong samples are memorized. Experimental results demonstrate that our method is effective, stable and can act as a performance boost plugin on many text classifiers and weakly supervised text classification methods. In the future, we are interested in automatically identifying the right granularity to measure learning order for a given dataset. Moreover, we are also interested in analyzing the learning order in classification tasks in image and speech domains.

Chapter 3, in full, is a reprint of the material as it appears in Dheeraj Mekala, Chengyu Dong, and Jingbo Shang. 2022. "LOPS: Learning Order Inspired Pseudo-Label Selection for Weakly Supervised Text Classification", in Findings of the Association for Computational Linguistics: EMNLP 2022, pages 4894–4908, Association for Computational Linguistics. The dissertation/thesis author was the primary investigator and author of this paper.

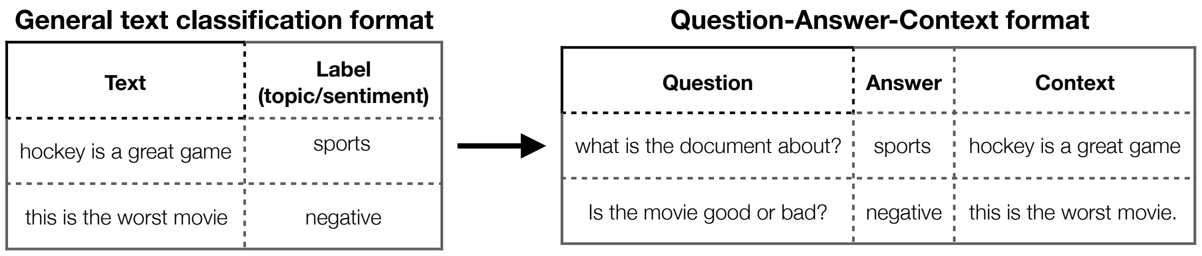
Chapter 4

Leveraging QA Datasets to Improve Generative Data Augmentation

4.1 Introduction

Recent advances in NLP have substantially improved the capability of pretrained language models to generate high-quality text (Radford & Narasimhan, 2018; Radford et al., 2019; Lewis et al., 2020; Brown et al., 2020). Various approaches (Kumar et al., 2020; Anaby-Tavor et al., 2020; Mekala et al., 2021) leverage this capability for *generative data augmentation*. This process usually involves first fine-tuning the GLM on training samples prepended with their target label and then generating synthetic data by prompting the GLM with a given target label. However, it is not evident that the model parameters learnt during pretraining or fine-tuning should support data generation using such unintuitive formulations with label tokens as prompts: In low data regimes, fine-tuning can be unstable (Devlin et al., 2019) and relies on the pretrained parameters to be reasonably well-suited for the target task (Phang et al., 2018). Therefore, for target domains that are different from the pretraining domain, such formulations may result in poor quality generation (Feng et al., 2020).

To address this challenge, we propose **CONDA**, an approach to leverage existing QA datasets for training ***Con**text generators* to improve generative **D**ata **A**ugmentation. We propose to use a question answering (QA) formulation as a consistent format to prompt GLMs for synthetic data: We use QA datasets for training GLMs to be *context generators* for a given



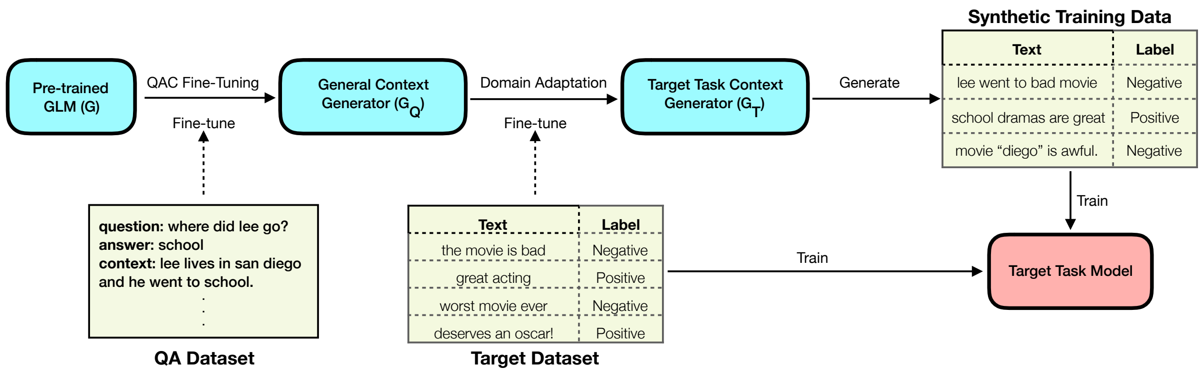
General text classification format

Text	Label (topic/sentiment)
hockey is a great game	sports
this is the worst movie	negative

Question-Answer-Context format

Question	Answer	Context
what is the document about?	sports	hockey is a great game
Is the movie good or bad?	negative	this is the worst movie.

Figure 4.1. Examples of converting topic classification and sentiment analysis data into question-answer-context format.



Pre-trained GLM (G)
QAC Fine-Tuning
Fine-tune
General Context Generator (G_Q)
Domain Adaptation
Fine-tune
Target Task Context Generator (G_T)
Generate
Synthetic Training Data
Text | Label
lee went to bad movie | Negative
school dramas are great | Positive
movie "diego" is awful. | Negative
Train
question: where did lee go?
answer: school
context: lee lives in san diego and he went to school.
QA Dataset
Text | Label
the movie is bad | Negative
great acting | Positive
worst movie ever | Negative
deserves an oscar! | Positive
Target Dataset
Train
Target Task Model

Figure 4.2. Overview of CONDA. We propose to use QA datasets for transforming pre-trained generative language models into high-quality target task data generators. We view QA datasets in question-answer-context format and fine-tune a pre-trained GLM (G) to obtain a general context generator (G_Q). Then, we adapt it to the target domain by training it further on few-shot target dataset supervision, resulting in G_T. Finally, using G_T, we generate synthetic training data for the target task, use it to augment the few-shot target dataset and train the target task model on the augmented data.

question and answer.

As illustrated in Figure 4.2, our method consists of two steps. The first step is QAC fine-tuning, where we fine-tune a pretrained language model on a QA dataset to obtain a general context generator that is capable of generating contexts for given questions and answers. To this end, we view the QA dataset in question-answer-context format instead of the context-question-answer format used to solve QA tasks (Radford & Narasimhan, 2018; Radford et al., 2019; Raffel et al., 2020). Then, we adapt the general context generator to the target domain by further training it on available few-shot data, resulting in a target-domain context generator. Inspired from recent work in converting several NLP tasks into a common format (McCann et al., 2018; Raffel et al., 2020), we format the target tasks into a question-answer schema. For example, as shown in Figure 4.1, topic classification and sentiment analysis data can be cast into the question-answer-context format with its respective *label* as *answer*, and *text* as *context*. We adapt the context generator to the target task domain by further training on target task few-shot supervision, resulting in target task context generator. Finally, we generate synthetic training data for the target task by generating contexts for questions and answers pertaining to the respective dataset. Then, we add the generated samples to the few-shot supervision and train a target task model on the augmented data.

We perform extensive experiments on multiple sentiment analysis and topic classification datasets with several abstractive, extractive, and common-sense reasoning QA datasets. Through rigorous experiments and thorough analysis, we observe that QA datasets that require high-level reasoning abilities such as abstractive and common-sense QA datasets suit the best for generating high-quality data.

Our contributions are summarized as follows:

- We propose to use QA datasets for training generative language models to be context generators for a given question and answer.
- We formulate various classification tasks into a QA format and model synthetic training data generation for these tasks as context generation.
- We perform experiments on multiple sentiment analysis and topic classification datasets to demonstrate the effectiveness of our method in zero- and few-shot settings.
- We release the code on Github[1].

4.2 Related Work

Data Augmentation

Wei & Zou (2019) propose a simple data augmentation method using synonym replacement, random insertion, random swap, and random deletion. Sennrich et al. (2016) augment

[1]https://github.com/dheeraj7596/CONDA

samples by translating them into foreign language and then back to English. Du et al. (2021) compute task-specific query embeddings to retrieve sentences from unlabeled documents from the Internet. After a rise in pretrained generative language models, the generation capabilities of these models have been explored to generate synthetic data. Anaby-Tavor et al. (2020); Kumar et al. (2020); Schick & Schütze (2021b); Mekala et al. (2021) generate labeled documents using the GLMs and (Yang et al., 2020) do so specifically for common-sense reasoning. Puri et al. (2020) use GLMs to synthesize questions and answers and improve performance on question answering. Vu et al. (2021) generate data for NLI tasks. Li et al. (2023a) introduce data augmentation for in-context learning via self-paraphrase.

Few-shot Learning

Our work is closely related to few-shot learning as we take a few annotated samples as supervision. The idea of formulating classification as a prompting task is getting increasingly popular. Brown et al. (2020) introduce a new paradigm called in-context learning to infer from large language models using few annotated samples. Schick & Schütze (2021a) formulate input samples as cloze-style phrases and assign pseudo-labels that are used for training the classifier and Tam et al. (2021) improves their approach further without using any task-specific unlabeled data. (McCann et al., 2018; Raffel et al., 2020) format several NLP tasks into a question-answer and text-to-text schema. Lin et al. (2022) train multilingual autoregressive language models to enable few-shot learning in multiple languages. Gao et al. (2021) propose to generate prompts and convert smaller pretrained language models to few-shot learners. Other work proposes to pre-train prompts by adding soft prompts into the pre-training stage (Gu et al., 2022; Vu et al., 2022b,a).

Language Model Fine-Tuning

Pre-trained language models are applied to downstream tasks by fine-tuning them using task-specific objectives (Howard & Ruder, 2018). However, this process requires significant annotated downstream task data (Yogatama et al., 2019). Many methods have been proposed to address this challenge. Gururangan et al. (2020) propose to continue training on unlabeled data from the target task domain. Aghajanyan et al. (2021) propose pre-finetuning, a large-scale multi-task learning stage between language model pre-training and fine-tuning. Phang et al. (2018) introduce intermediate task fine-tuning which involves fine-tuning a language model on an auxiliary task before continuously training on the target task. Pruksachatkun et al. (2020) observe that the tasks requiring high-level inference and reasoning abilities are the best choice as intermediate tasks. Vu et al. (2020) identify the best auxiliary tasks for high performance on downstream tasks. Vu et al. (2021) use NLI as auxiliary task to generate synthetic NLI data for intermediate fine-tuning. Our method differs from (Phang et al., 2018) in two fronts: (1) we use QA datasets for training context generators instead of answering the question , and (2) we use the fine-tuned GLM to generate synthetic data instead of training directly for the downstream tasks. It also differs from (Vu et al., 2021) in terms of the generated data, where they consider NLI as an auxiliary task and generate synthetic samples in target-domain for the NLI task irrespective of the target task and perform intermediate task fine-tuning. CONDA formats target tasks into question-answer format and directly generates samples relevant for target task.

4.3 CONDA: QA Datasets for Generative Data Augmentation

In this section, we describe the problem statement, and explain our method including QAC fine-tuning, target-domain adaptation, and synthetic training data generation.

4.3.1 Problem Formulation

For a given task $\mathscr{T}$, the input in a few-shot setting contains a very small labeled dataset $\mathscr{L}_{\mathscr{T}} = \{(\mathscr{D}_1, l_1), (\mathscr{D}_2, l_2), \ldots, (\mathscr{D}_{|\mathscr{L}_{\mathscr{T}}|}, l_{|\mathscr{L}_{\mathscr{T}}|})\}$ and m target classes $\mathscr{C} = \{\mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_m\}$. Our method requires users to provide a question per dataset that is representative of the task to be solved. Our aim is to build a model for the task $\mathscr{T}$ that assigns a label $\mathscr{C}_j \in \mathscr{C}$ to each document $\mathscr{D}$.

4.3.2 QAC Fine-tuning

We consider question-answering datasets Q containing triplets (q,a,c) of a question q, the corresponding answer a, and a context c required to derive the correct answer. Question-answering datasets can roughly be divided into *extractive* (Rajpurkar et al., 2016; Trischler et al., 2017; Joshi et al., 2017; Reddy et al., 2019) and *abstractive* datasets (Kočiský et al., 2018; Huang et al., 2019b; Xiong et al., 2019; Sap et al., 2019). For extractive QA datasets, the answer can be found as a contiguous span in the context, whereas in abstractive QA datasets, the answer needs to be generated in natural language without being able to rely on the vocabulary of the question or context.

We transform the QA dataset Q into training data D_{QAC} for fine-tuning GLM. To this end, each triplet (q, a, c) is converted into a single text by prepending *"question:"*, *"answer:"* and *"context:"*, respectively, and concatenating q, a and c separated by newlines. For example, a preprocessed training document in D_{QAC} from an extractive QA dataset might look as follows:

`question`: when did battle of plassey happen?

`answer`: 23 june 1757

`context`: the battle of plassey was a decisive victory of the british east india company over the nawab of bengal and his french allies on 23 june 1757.

We fine-tune a pretrained GLM G on D_{QAC} to obtain a *general context generator* G_Q using a language modeling objective to maximize the log-likelihood of the (q, a, c) triplet. The general context generator G_Q is capable of generating contexts for given questions and answers.

4.3.3 Domain Adaptation and Synthetic Training Data Generation

We adopt G_Q to the target domain by fine-tuning it further on available few-shot data. To preserve its context generating ability, we perform QAC fine-tuning instead of regular language model fine-tuning. This is enabled by transforming the few-shot supervision into our question-answer-context format. First, we manually design one question per dataset that is representative of the task and the dataset. Furthermore, following Schick & Schütze (2021a), we define a verbalizer as a mapping v: $\mathscr{C} \rightarrow \mathscr{V}$ that maps each label in $\mathscr{C}$ to a word from G_Q's vocabulary $\mathscr{V}$. Finally, for every document $\mathscr{D}_i$ and its respective label l_i in our few-shot data, we consider the verbalizer mapping of the label, $v(l_i)$, as answer and the text $\mathscr{D}_i$ as context. For example, a *negative* review *"I hate this movie"* from the IMDb dataset (Maas et al., 2011) is transformed as follows:

```
question: is the movie good or bad?

answer: bad

context: i hate this movie.
```

We fine-tune G_Q on the converted few-shot data to obtain a target task context generator $G_{\mathscr{T}}$.

Synthetic Training Data Generation

Recall that our method requires a question q for every dataset that is representative of the task to be solved. To obtain synthetic training data, for every distinct label $\mathscr{C}_j$, we create a question-answer prompt with q as question, $v(\mathscr{C}_j)$ as answer and let $G_{\mathscr{T}}$ generate the context c_{gen}. The generated context c_{gen} and label $\mathscr{C}_j$ are considered as a synthetic training sample. We repeat this process multiple times to generate n samples that we collect in a synthetic training dataset denoted by $\mathscr{D}_{gen}$.

As a final step, we train the target task model on the combination of $\mathscr{D}_{gen}$ and our original few-shot dataset $\mathscr{L}_{\mathscr{T}}$. We use this trained target-task model for inference.

4.4 Experiments

In this section, we evaluate our method against several data augmentation and few-shot methods on sentiment analysis and text classification tasks.

Table 4.1. Relevant statistics of the QA dataset used in our experiments.

Dataset	Type	# Samples	Training Context
SQuAD	Extractive	87,600	Wikipedia
NewsQA	Extractive	76,560	News
TweetQA	Abstractive	10,692	News Tweets
SocialIQA	Commonsense	33,410	Crowdsourcing
CosmosQA	Commonsense	21,448	Blogs

4.4.1 QA Datasets

We consider several extractive, abstractive, and common-sense QA datasets. Common-sense QA datasets are also abstractive datasets that require common-sense reasoning to answer the questions. The QA dataset statistics are provided in Table 4.1. The details of these datasets are as follows:

- **SQuAD** (Rajpurkar et al., 2016, 2018) is a collection of questions and answers based on Wikipedia articles.
- **NewsQA** (Trischler et al., 2017) is a challenging QA dataset in the News domain where crowdworkers were shown a news article's headline and summary, and asked to formulate a question about the article without accessing its content.
- **TweetQA** (Xiong et al., 2019) is a QA dataset made from a collection of tweets sampled from two major news websites (CNN and NBC).
- **SocialIQA** (Sap et al., 2019) is a QA dataset that tests social common-sense intelligence. The data is made of common phrases from stories and books.
- **CosmosQA** (Huang et al., 2019b) is a commonsense-based reading comprehension task formulated as multiple-choice questions. Answering questions requires reasoning not only based on the exact text spans in the context, but also abstractive commonsense reasoning.

Table 4.2. Questions and Verbalized labels of the target task datasets considered in our experiments.

Dataset	Question	Verbalized Labels
	Sentiment	
IMDb	*is this movie good or bad?*	good, bad
Yelp	*how is the service?*	awful, bad, fine, good, excellent
SST-2	*is this sentence positive or negative?*	positive, negative
	Topic	
Yahoo	*what is this document about?*	sports, society, science, health, politics, education, computer, business, entertainment, relationship
NYT	*what is this document about?*	arts, business, politics, sports
AGNews	*what is this document about?*	sports, business, technology, politics

4.4.2 Target Task Datasets

We evaluate our method on six English text classification datasets. In particular, we consider the three sentiment analysis datasets: IMDb reviews (Maas et al., 2011), Yelp[2], and SST-2 (Socher et al., 2013), as well as three topic classification datasets: Yahoo (Zhang et al., 2015), The New York Times[3] (NYT), and AGNews (Zhang et al., 2015). The dataset-representative questions, and their respective verbalized labels of target task datasets are mentioned in Table 4.2. We follow and adapt McCann et al. (2018) for questions in sentiment analysis datasets. The question for topic classification is intuitive and straightforward. More details about the datasets can be found in Appendix B.1.

4.4.3 Compared Methods

We compare with a wide range of data augmentation and intermediate-task fine-tuning (ITFT) methods described below:

• **BERT-FT** trains the `BERT-base-uncased` classifier (Devlin et al., 2019) on the few-shot

[2]https://www.yelp.com/dataset/
[3]http://developer.nytimes.com/

supervision.

- **ITFT-*X*** (Phang et al., 2018) first trains a model on dataset X and fine-tunes it further on the target task. We compare with ITFT-MNLI and ITFT-SQuAD fine-tuned intermediately on MNLI (Williams et al., 2018) and SQuAD datasets respectively.
- **BackTranslation** (Sennrich et al., 2016) augments samples by translating them into a non-English language and translating them back to English. We translate them to French, Spanish, and Portuguese thereby augmenting three synthetic samples for every sample.
- **PEGASUS** (Zhang et al., 2020a) is a state-of-the-art paraphrasing model. We paraphrase the input text and consider it as a synthetic sample and augment it to the training set.
- **EDA** (Wei & Zou, 2019) generates synthetic samples by synonym replacement, random insertion, random swap, and random deletion and augment them to the training set.
- **LAMBADA** (Anaby-Tavor et al., 2020) fine-tunes a GLM on few-shot supervision prepended with their target labels and then generates synthetic data by prompting the GLM with a given target label.

We denote our method as CONDA, which includes QAC fine-tuning, domain adaptation, synthetic samples generation, and training the target task classifier. CONDA-*X* represents that the QAC fine-tuning of GLM is performed on QA dataset X. We also compare with CONDA$\backslash QA$ where we perform no QAC fine-tuning and directly fine-tune the GLM on target dataset.

4.4.4 Experiment Settings

We consider two low-data regimes: few-shot and zero-shot. We consider 8 annotated samples per label in the few-shot setting. In the zero-shot setting, we skip the domain adaptation step and use G_Q directly for synthetic training data generation and train the target task model only on the generated synthetic training data. We use GPT2-Medium (Radford et al., 2019) as our GLM and fine-tune it for 3 epochs in QAC-fine-tuning and domain adaptation steps. While generating synthetic training samples, we use top-k sampling with $k = 20$, a maximum length of 200 tokens, and generate $n = 450$ synthetic samples per label. We use `BERT-base-uncased` (De-

Table 4.3. Few-Shot Evaluation Results. Micro- and Macro-F1 are used as evaluation metrics. All experiments are repeated with three random seeds. Mean and standard deviation (in the subscript) are reported. The best baseline for each dataset is underlined and all results of CONDA that outperform the best baseline are in bold.

	Sentiment						**Topic**					
	IMDb		**Yelp**		**SST-2**		**NYT**		**Yahoo**		**AGNews**	
Method	Mi-F$_1$	Ma-F$_1$	Mi-F$_1$	Ma-F$_1$	Mi-F$_1$	Ma-F$_1$	Mi-F$_1$	Ma-F$_1$	Mi-F$_1$	Ma-F$_1$	Mi-F$_1$	Ma-F$_1$
BERT-FT	$69.1_{4.9}$	$69.1_{4.9}$	$39.8_{2.3}$	$38.9_{3.4}$	$62.0_{4.7}$	$61.8_{4.8}$	$94.4_{1.1}$	$88.1_{1.6}$	$55.4_{2.1}$	$55.2_{1.6}$	$78.4_{1.8}$	$78.3_{1.8}$
ITFT-MNLI	$\underline{73.9_{4.6}}$	$\underline{73.5_{4.8}}$	$40.4_{2.6}$	$40.0_{2.9}$	$\underline{66.5_{6.9}}$	$\underline{65.5_{6.9}}$	$90.1_{1.2}$	$80.8_{1.1}$	$38.7_{5.8}$	$37.6_{5.2}$	$71.1_{1.1}$	$70.6_{1.1}$
ITFT-SQuAD	$65.5_{4.4}$	$64.6_{4.3}$	$38.4_{2.5}$	$36.8_{2.5}$	$61.9_{2.2}$	$61.5_{2.2}$	$93.0_{0.5}$	$85.3_{1.3}$	$45.3_{3.4}$	$45.0_{3.0}$	$72.0_{1.9}$	$71.4_{2.1}$
BackTranslation	$68.0_{4.6}$	$67.1_{5.2}$	$\underline{41.6_{2.4}}$	$\underline{40.3_{3.0}}$	$60.6_{4.5}$	$60.0_{4.9}$	$95.4_{0.6}$	$90.0_{1.3}$	$57.4_{1.4}$	$57.1_{1.2}$	$80.0_{2.2}$	$79.8_{2.3}$
PEGASUS	$66.8_{5.0}$	$65.9_{5.6}$	$35.9_{3.7}$	$34.4_{3.1}$	$61.1_{5.3}$	$60.9_{5.3}$	$93.2_{0.5}$	$87.2_{0.6}$	$58.1_{1.9}$	$57.2_{1.7}$	$\underline{81.1_{1.9}}$	$\underline{80.9_{2.1}}$
EDA	$63.6_{1.4}$	$62.1_{1.6}$	$39.1_{1.9}$	$37.9_{2.0}$	$57.4_{4.0}$	$52.9_{7.4}$	$\underline{95.8_{0.8}}$	$\underline{90.9_{1.7}}$	$56.1_{1.7}$	$55.8_{1.8}$	$80.0_{3.0}$	$79.8_{3.0}$
LAMBADA	$50.3_{0.7}$	$42.3_{5.4}$	$20.8_{1.06}$	$11.1_{6.3}$	$49.6_{1.3}$	$45.8_{3.3}$	$60.3_{19.7}$	$45.9_{17.4}$	$25.7_{4.7}$	$22.6_{3.2}$	$49.3_{9.9}$	$46.9_{10.3}$
CONDA$\backslash$*QA*	$72.2_{6.9}$	$71.3_{8.0}$	$36.8_{0.6}$	$23.9_{1.7}$	$50.6_{0.5}$	$35.1_{0.5}$	$93.5_{0.8}$	$85.7_{1.2}$	$\underline{58.5_{0.3}}$	$\underline{57.3_{0.4}}$	$79.4_{1.6}$	$78.8_{1.8}$
CONDA-SQuAD	$53.9_{1.9}$	$45.9_{6.2}$	$37.9_{0.7}$	$31.1_{3.2}$	$51.5_{1.6}$	$39.8_{7.6}$	$93.2_{0.8}$	$86.0_{1.7}$	$56.9_{0.5}$	$55.4_{0.5}$	$\mathbf{81.6_{0.8}}$	$\mathbf{81.3_{0.9}}$
CONDA-NewsQA	$57.9_{3.7}$	$55.5_{5.4}$	$36.4_{1.1}$	$31.6_{2.0}$	$56.0_{6.2}$	$50.5_{10.3}$	$91.5_{0.3}$	$81.1_{0.6}$	$58.3_{0.7}$	$57.2_{0.8}$	$80.0_{3.3}$	$79.6_{3.6}$
CONDA-TweetQA	$\mathbf{75.1_{2.3}}$	$\mathbf{74.5_{2.5}}$	$\mathbf{42.9_{1.1}}$	$\mathbf{42.0_{1.8}}$	$\mathbf{67.7_{4.8}}$	$\mathbf{67.5_{4.9}}$	$94.1_{0.6}$	$86.6_{1.3}$	$\mathbf{59.4_{0.4}}$	$\mathbf{58.1_{0.3}}$	$\mathbf{83.0_{0.9}}$	$\mathbf{82.9_{0.9}}$
CONDA-SocialIQA	$\mathbf{79.5_{1.9}}$	$\mathbf{79.5_{1.9}}$	$39.4_{1.5}$	$32.2_{2.8}$	$\mathbf{75.4_{1.4}}$	$\mathbf{75.2_{1.6}}$	$93.2_{3.4}$	$85.8_{1.3}$	$\mathbf{61.9_{0.5}}$	$\mathbf{61.1_{0.6}}$	$\mathbf{81.9_{0.2}}$	$\mathbf{81.7_{0.2}}$
CONDA-CosmosQA	$\mathbf{77.0_{3.2}}$	$\mathbf{76.4_{3.7}}$	$\mathbf{42.3_{0.1}}$	$37.5_{1.0}$	$\mathbf{67.4_{0.6}}$	$\mathbf{66.9_{1.2}}$	$94.3_{0.4}$	$87.7_{1.1}$	$\mathbf{63.8_{0.6}}$	$\mathbf{63.3_{0.4}}$	$\mathbf{82.8_{0.8}}$	$\mathbf{82.5_{0.8}}$

vlin et al., 2019) as target task classifier. We feed `[CLS]` representation into the classification head and train all the parameters on the downstream target tasks. Following (Devlin et al., 2019), we fix the number of epochs of target task BERT classifier training to 4 unless mentioned otherwise. We perform 3 random restarts and report the mean and standard deviation.[4] We use the Transformers library (Wolf et al., 2020) and NVIDIA RTX A6000 GPUs for our experiments.

To enable a fair comparison, we generate the same number of samples per label as CONDA (i.e., 450) for all data augmentation baselines. We use `BERT-base-uncased` as the target task classifier for all baselines. CONDA$\backslash$*QA* for zero-shot setting implies a pre-trained GPT2. While training the target task classifier, since the number of training samples for baselines like BERT-FT, ITFT are different than data augmentation baselines and our method CONDA, we set the number of epochs for all baselines such that the number of update steps remain the same for a fair comparison.

[4]For each restart, we resample the few-shot training set.

Table 4.4. Zero-Shot Evaluation Results. Mean and standard deviation (in the subscript) are reported.

	Sentiment						Topic					
	IMDb		Yelp		SST-2		NYT		Yahoo		AGNews	
Method	Mi-F$_1$	Ma-F$_1$	Mi-F$_1$	Ma-F$_1$	Mi-F$_1$	Ma-F$_1$	Mi-F$_1$	Ma-F$_1$	Mi-F$_1$	Ma-F$_1$	Mi-F$_1$	Ma-F$_1$
CONDA\QA	$70.9_{3.7}$	$70.0_{3.8}$	$32.4_{3.7}$	$19.8_{2.0}$	$52.9_{3.6}$	$41.8_{9.9}$	$90.7_{0.9}$	$80.0_{1.8}$	$57.9_{0.3}$	$57.1_{0.7}$	$77.0_{1.5}$	$76.2_{1.6}$
CONDA-SQuAD	$53.3_{2.4}$	$42.7_{7.1}$	$30.2_{4.5}$	$21.4_{4.6}$	$52.4_{2.6}$	$47.3_{5.9}$	$85.7_{3.8}$	$74.3_{3.1}$	$56.5_{1.5}$	$54.9_{1.7}$	$79.3_{0.1}$	$78.9_{0.2}$
CONDA-NewsQA	$53.4_{2.6}$	$47.4_{10.0}$	$32.8_{1.0}$	$23.1_{3.6}$	$51.6_{1.7}$	$46.2_{3.6}$	$89.8_{0.2}$	$77.1_{1.1}$	$55.9_{1.1}$	$54.6_{0.8}$	$76.8_{3.0}$	$75.7_{3.5}$
CONDA-TweetQA	$72.4_{5.0}$	$70.6_{6.1}$	$\mathbf{38.0_{2.4}}$	$\mathbf{37.6_{2.3}}$	$61.2_{3.9}$	$56.5_{6.4}$	$90.3_{1.9}$	$78.5_{4.7}$	$54.0_{0.4}$	$52.4_{0.2}$	$76.4_{1.7}$	$76.2_{2.0}$
CONDA-SocialIQA	$\mathbf{78.3_{4.3}}$	$\mathbf{77.6_{5.1}}$	$36.9_{2.2}$	$31.7_{1.1}$	$\mathbf{76.2_{1.2}}$	$\mathbf{76.1_{1.2}}$	$87.0_{3.0}$	$77.6_{2.9}$	$56.1_{1.6}$	$55.2_{2.1}$	$79.6_{1.9}$	$79.4_{2.2}$
CONDA-CosmosQA	$75.9_{2.3}$	$75.5_{2.8}$	$37.4_{1.5}$	$35.1_{1.7}$	$66.2_{6.4}$	$66.0_{6.5}$	$\mathbf{92.8_{0.3}}$	$\mathbf{84.5_{1.3}}$	$\mathbf{63.4_{0.5}}$	$\mathbf{62.9_{0.3}}$	$\mathbf{81.8_{1.6}}$	$\mathbf{81.5_{1.4}}$

4.4.5 Results and Discussion

Results for few- and zero-shot settings are shown in Table 4.3 and Table 4.4, respectively, using Micro- and Macro-F1 as evaluation metrics. We discuss the effectiveness of our method below.

CONDA vs Baselines. In the few-shot setting, CONDA with abstractive and common-sense based datasets outperforms all baselines for most of the datasets, beating the best baseline in five out of six cases. CONDA performs better than BERT-FT on all datasets, achieving up to 14% improvement on SST-2. Although ITFT performs better than vanilla fine-tuning, CONDA demonstrates better performance than ITFT on all datasets. For example, CONDA-TweetQA shows 11% improvement over ITFT-SQuAD on AG-News. CONDA demonstrates higher performance than data-augmentation baselines on all datasets except NYT. The comparison between CONDA and LAMBADA shows that our QA formulation prompt is more intuitive and informative than just the target label. We attribute the superior performance of CONDA to the context-generating ability acquired during QAC fine-tuning that is efficiently leveraged by generating synthetic samples, which are added to the training set.

Abstractive vs Extractive QA Datasets. We observe that the performance of CONDA with abstractive QA datasets is significantly better than CONDA with extractive QA datasets in both few-shot and zero-shot settings. For example, CONDA-TweetQA has an improvement of more than 20% over CONDA-SQuAD on IMDb in few-shot setting. We surmise that this is because of the intrinsic nature of extractive QA datasets (i.e., the answer always being present in the context as a contiguous span). We observe that GLMs fine-tuned on an extractive QA dataset retain the ability to generate contexts that encompass the answer. Note that, while generating synthetic training samples, the answer in the prompt is its respective topic. For example, out of 500 generated samples by CONDA-SQuAD for Yelp dataset, 213 samples contain at least one occurrence of its corresponding verbalized label whereas it is only 73 for CONDA-CosmosQA. Thus, many synthetic samples generated contain their corresponding label in text. Therefore, a classifier trained on synthetic samples that have their corresponding labels in the text, easily overfits on the label tokens and does not generalize well to unseen test data.

Comparison with CONDA$\backslash$QA. CONDA with abstractive QA datasets perform better than CONDA$\backslash$*QA* in both few-shot and zero-shot settings, attaining improvements up to 40% and 35% respectively in macro-F1 on SST-2. This demonstrates that the context generating abilities are learnt and reinforced during the QAC fine-tuning on QA datasets which is efficiently utilized by generating synthetic samples.

Zero-shot Performance. The zero-shot performance of CONDA follows a similar trend as few-shot performance: abstractive and common-sense reasoning QA datasets lead to better performance than extractive datasets and no QAC fine-tuning.

4.4.6 Ablation Study

To understand the impact of domain adaptation and few-shot samples, we compare CONDA with two ablated versions in Table 4.5: (1) CONDA-*DA* represents our method without domain adaptation (i.e., generating synthetic data using G_Q and training the classifier on combined few-shot supervision and synthetic data generated by G_Q), (2) CONDA-*Few Shot* represents the classifier trained only on the samples generated by G_T. We also present the results of our complete pipeline for reference. CONDA performs better than CONDA-Few shot in most cases, demonstrating the importance of including few-shot samples in the training set for the classifier. The comparison between CONDA and CONDA-DA suggests that fine-tuning the

Table 4.5. Ablation Study. Macro-F1 is used as evaluation metric.

		Sentiment			**Topic**		
QA Dataset	**Setting**	**IMDb**	**Yelp**	**SST-2**	**NYT**	**Yahoo**	**AGNews**
SQuAD	ConDA	$45.9_{6.2}$	$31.1_{3.2}$	$39.8_{7.6}$	$86.0_{1.7}$	$55.4_{0.5}$	$81.3_{0.9}$
	- DA	$51.3_{12.7}$	$28.2_{0.5}$	$33.4_{0.1}$	$87.1_{1.0}$	$55.0_{1.7}$	$82.5_{0.6}$
	- Few Shot	$49.4_{9.5}$	$25.9_{3.4}$	$43.7_{4.0}$	$75.0_{4.0}$	$47.4_{0.4}$	$77.9_{3.0}$
NewsQA	ConDA	$55.5_{5.4}$	$31.6_{2.0}$	$50.5_{10.3}$	$81.1_{0.6}$	$57.2_{0.8}$	$79.6_{3.6}$
	- DA	$60.9_{9.6}$	$32.0_{3.6}$	$46.2_{8.5}$	$79.8_{0.4}$	$56.4_{1.1}$	$79.2_{3.5}$
	- Few Shot	$50.9_{4.8}$	$23.4_{1.9}$	$46.0_{7.0}$	$77.2_{2.1}$	$54.3_{0.7}$	$76.0_{4.1}$
TweetQA	ConDA	$74.5_{2.5}$	$42.0_{1.8}$	$67.5_{4.9}$	$86.6_{1.3}$	$58.1_{0.3}$	$82.9_{0.9}$
	- DA	$80.5_{3.5}$	$42.1_{0.2}$	$63.2_{7.5}$	$85.3_{2.1}$	$57.1_{1.1}$	$81.1_{1.6}$
	- Few Shot	$74.0_{2.7}$	$40.6_{0.7}$	$59.3_{12.4}$	$77.3_{4.8}$	$53.8_{0.3}$	$77.4_{1.5}$
SocialIQA	ConDA	$79.5_{1.9}$	$32.2_{2.8}$	$75.2_{1.6}$	$85.8_{1.3}$	$61.1_{0.6}$	$81.7_{0.2}$
	- DA	$81.0_{1.9}$	$35.3_{0.8}$	$76.1_{0.6}$	$87.6_{1.3}$	$60.7_{0.8}$	$82.4_{1.2}$
	- Few Shot	$77.7_{4.0}$	$31.4_{3.0}$	$75.2_{1.0}$	$76.5_{1.7}$	$55.8_{1.2}$	$78.2_{1.4}$
CosmosQA	ConDA	$76.4_{3.7}$	$37.5_{1.0}$	$66.9_{1.2}$	$87.7_{1.1}$	$63.3_{0.4}$	$82.5_{0.8}$
	- DA	$76.3_{2.4}$	$36.8_{2.3}$	$51.8_{9.3}$	$87.1_{1.0}$	$62.9_{0.3}$	$82.6_{0.6}$
	- Few Shot	$74.9_{1.0}$	$35.8_{1.4}$	$64.4_{9.1}$	$82.9_{0.7}$	$60.1_{0.2}$	$80.3_{1.4}$

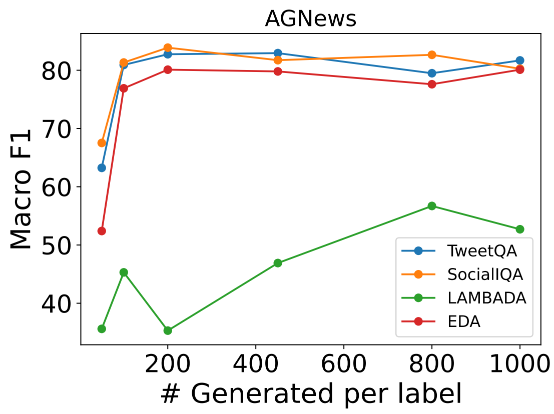
language model further on the target dataset helps in some scenarios but does not always improve performance. This is in line with previous research findings (Du et al., 2021; Vu et al., 2021; Pryzant et al., 2022). We conjecture that domain adaptation is important when the structure of the target task dataset is very different from the QA dataset. For example, domain adaptation helps most of the QA datasets on SST-2 dataset because the text in SST-2 is a single sentence, whereas most of the QA datasets have paragraphs as context. Moreover, it also depends on the number of samples the language model is fine-tuned on during domain adaptation. We observe that the higher the number of samples, the more positive their impact. For example, the number of few-shot samples is the highest in Yahoo compared to other datasets and domain adaptation positively contributes to the performance on Yahoo for all QA datasets.

4.4.7 Larger Generative Language Models

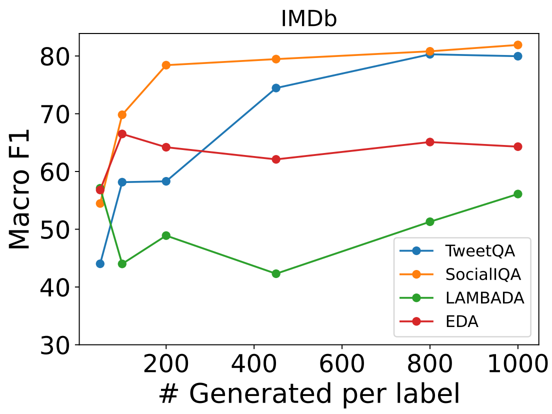
Experimental results with GPT2-Large as the GLM are shown in Table 4.6. We observe that the relative performance trend remains the same as GPT2-Medium i.e. ConDA with abstractive datasets performs better than ConDA with extractive datasets and ConDA$\backslash$*QA*-L.

Table 4.6. Few-Shot Evaluation Results with GPT2-Large as GLM (*-L* denotes GPT2-Large). Macro-F1 is used as evaluation metric. All results of CONDA-L that perform better than CONDA\QA-L are in bold.

	Sentiment			Topic		
Method	**IMDb**	**Yelp**	**SST-2**	**NYT**	**Yahoo**	**AGNews**
CONDA\QA-L	$79.7_{2.1}$	$43.2_{4.4}$	$67.6_{8.2}$	$84.8_{1.3}$	$60.3_{0.8}$	$77.7_{2.2}$
CONDA-L-SQuAD	$70.0_{15.2}$	$38.9_{1.5}$	$64.3_{7.2}$	$84.3_{2.9}$	$60.3_{1.3}$	$\mathbf{81.1_{1.7}}$
CONDA-L-NewsQA	$72.4_{9.4}$	$38.3_{0.3}$	$58.1_{6.2}$	$\mathbf{85.5_{2.4}}$	$\mathbf{61.4_{1.1}}$	$\mathbf{82.2_{1.2}}$
CONDA-L-TweetQA	$76.4_{5.7}$	$\mathbf{45.0_{1.3}}$	$\mathbf{74.6_{2.3}}$	$84.4_{0.1}$	$\mathbf{61.6_{0.1}}$	$\mathbf{79.7_{3.1}}$
CONDA-L-SocialIQA	$\mathbf{81.6_{2.4}}$	$\mathbf{43.9_{3.4}}$	$\mathbf{77.5_{1.3}}$	$\mathbf{89.0_{0.4}}$	$\mathbf{62.0_{0.5}}$	$\mathbf{80.9_{2.2}}$
CONDA-L-CosmosQA	$\mathbf{83.4_{0.6}}$	$\mathbf{43.2_{2.2}}$	$\mathbf{77.2_{1.7}}$	$\mathbf{86.5_{2.4}}$	$\mathbf{61.0_{0.6}}$	$\mathbf{79.5_{3.9}}$



(a) AGNews



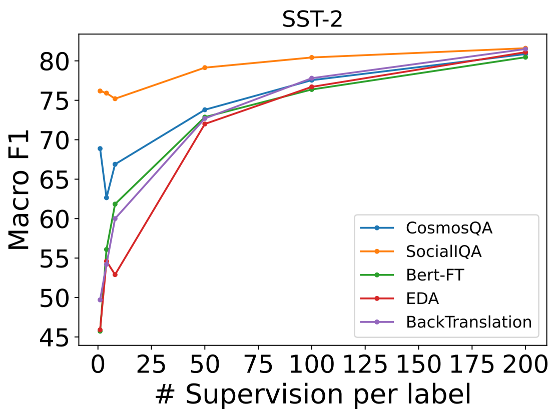
(b) IMDb

Figure 4.3. Macro-F_1 scores of CONDA-TweetQA and CONDA-SocialIQA w.r.t. number of generated samples per class. We fix the few-shot supervision size to 8 samples per label. Each experiment is repeated with three different seeds and the mean performance is plotted.

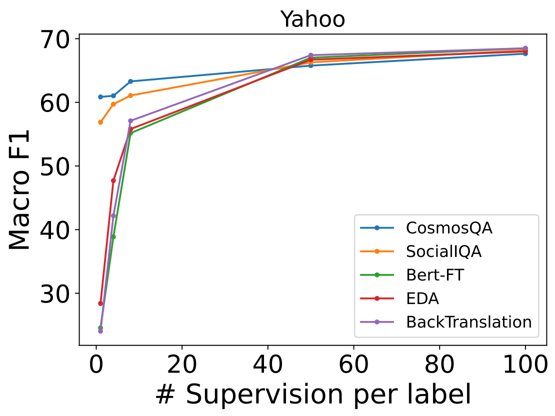
This indicates that QAC fine-tuning improves the performance of generative data augmentation with larger GLMs as well.

4.4.8 Performance vs No. of Generated Samples

We fix the few-shot supervision size to 8 samples per label and vary the number of generated samples per label and plot the performance of CONDA-TweetQA, CONDA-SocialIQA, and baselines such as LAMBADA and EDA on AGNews and IMDb datasets, shown in Figure 4.3. We repeat each setting with three different seeds and plot the mean performance. We observe that the performance increases and it plateaus after a while. This shows that synthetic training data



(a) SST-2



(b) Yahoo

Figure 4.4. Macro-F_1 scores of CONDA-CosmosQA and CONDA-SocialIQA w.r.t. number of few-shot annotated samples per class. Each experiment is repeated with three different seeds and mean performance is plotted.

can give a substantial boost to the few-shot training data, minimizing the human effort in manual annotations; however, it cannot replace the original training data completely as it requires more human annotated data to improve beyond some limit.

4.4.9 Performance vs Few-shot supervision Size

We fix the number of generated samples to 450 per label and vary the number of annotated samples and plot the performance of CONDA-CosmosQA and CONDA-SocialIQA on SST-2 and Yahoo datasets in Figure 4.4. We also plot the performance of baselines such as BERT-FT, EDA, BackTranslation for comparison. We repeat each experiment with three random seeds and plot the mean performance. We observe that the performance of CONDA increases with the size of supervision and the improvement over baselines in the low-data regime is substantial. For example, with only 4 annotated samples per label in Yahoo dataset, the macro F1 of CONDA-CosmosQA outperforms BERT-FT by 22% and EDA by 15%. However, we also observe that the performance gap between CONDA and baselines decreases with increase in supervision size and gets stagnated after a while. As the size of supervision increases, the supervision by itself is sufficient for high performance, thus reducing the performance boost due to synthetic training data.

Table 4.7. Few-Shot Evaluation comparison between language model pre-training on unlabeled data (LMPT) and CONDA. Macro-F1 is used as evaluation metric. All results of CONDA that perform better than LMPT are in bold.

Method	IMDb	SST-2	Yahoo	AGNews
LMPT	$64.8_{3.6}$	$57.5_{2.1}$	$49.9_{1.2}$	$79.2_{1.7}$
CONDA-TweetQA	$\mathbf{74.5_{2.5}}$	$\mathbf{67.5_{4.9}}$	$\mathbf{58.1_{0.3}}$	$\mathbf{82.9_{0.9}}$
CONDA-SocialIQA	$\mathbf{79.5_{1.9}}$	$\mathbf{75.2_{1.6}}$	$\mathbf{61.1_{0.6}}$	$\mathbf{81.7_{0.2}}$
CONDA-CosmosQA	$\mathbf{76.4_{3.7}}$	$\mathbf{66.9_{1.2}}$	$\mathbf{63.3_{0.4}}$	$\mathbf{82.5_{0.8}}$

Table 4.8. Self-Training experiment results with Macro-F1 as evaluation metric. + *ST* denotes with self-training. Self-training improves the performance of both CONDA and CONDA-L significantly. All results where self-training improved the performance are in bold.

QA Dataset	Setting	SST-2	NYT	AGNews
TweetQA	CONDA	$67.5_{4.9}$	$86.6_{1.3}$	$82.9_{0.9}$
	CONDA + ST	$\mathbf{69.2_{1.3}}$	$\mathbf{88.2_{1.0}}$	$82.4_{1.7}$
	CONDA-L	$74.6_{2.3}$	$84.4_{0.1}$	$79.7_{3.1}$
	CONDA-L + ST	$\mathbf{76.9_{1.1}}$	$\mathbf{87.4_{2.4}}$	$\mathbf{80.9_{3.4}}$
SocialQA	CONDA	$75.2_{1.6}$	$85.8_{1.3}$	$81.7_{0.2}$
	CONDA + ST	$\mathbf{79.8_{0.8}}$	$\mathbf{90.3_{1.9}}$	$\mathbf{83.9_{1.5}}$
	CONDA-L	$77.5_{1.3}$	$89.0_{0.4}$	$80.9_{2.2}$
	CONDA-L + ST	$\mathbf{78.6_{0.6}}$	$\mathbf{92.1_{0.8}}$	$\mathbf{81.1_{1.8}}$
CosmosQA	CONDA	$66.9_{1.2}$	$87.7_{1.1}$	$82.5_{0.8}$
	CONDA + ST	$\mathbf{71.6_{6.9}}$	$87.2_{2.3}$	$\mathbf{83.6_{0.6}}$
	CONDA-L	$77.2_{1.7}$	$86.5_{2.4}$	$79.5_{3.9}$
	CONDA-L + ST	$\mathbf{79.2_{1.3}}$	$\mathbf{87.2_{4.0}}$	$\mathbf{80.7_{3.6}}$

4.4.10 Self-Training

We perform an experiment to demonstrate that the performance can be further improved through self-training when in-domain unlabeled samples are provided. In-domain unlabeled samples are often easily available in real-world scenarios. Self-training is a commonly-used approach to bootstrap the classifier on unlabeled samples (Mekala & Shang, 2020; Mekala et al., 2020; Vu et al., 2021). Following Vu et al. (2021), we obtain pseudo-labels by predicting on unlabeled samples using the trained classifier and train the classifier further on the available labeled and pseudo-labeled data. We consider the training set without ground truth labels as unlabeled data and experiment on SST-2, NYT, and AGNews datasets. We repeat this process for 3 iterations without any filtering of pseudo-labels. From the results in Table 4.8, we can observe a significant performance improvement up to 4 points with self-training. It is noteworthy that this improvement is consistent for both GPT2-Medium and Large models respectively.

4.4.11 Synthetic Data Adds Value

Unsupervised language model pre-training(LMPT) on target-task unlabeled data can improve performance (Gururangan et al., 2020). We consider training set without ground truth labels as unlabeled data for LMPT and present a comparison in few-shot setting in Table 4.7. We observe CONDA performs better than LMPT demonstrating the quality and importance of generated synthetic data.

4.4.12 Case study: Evaluating Context Generator

We hypothesize that our method results in high-quality context generators that are capable of generating context for a given question and answer. To validate this hypothesis in in-domain and out-of-domain settings, we perform two experiments on QA task.

In-domain Analysis. In this experiment, we validate whether the context generator is capable of generating context for question, answer pairs belonging to the same domain as QA dataset used for QAC fine-tuning. We consider SQuAD dataset and partition it into training set with 1000 (question, answer, context) triplets, dev set of size 1700 with only (question, answer) pairs and a test set of size 6570. First, we consider GPT2-Medium as GLM and perform QAC fine-tuning on the training set. Then, we generate contexts for the dev set and augment the (question, answer, generated context) triplets to the training set. Finally, we train a `BERT-base-uncased` QA model on the augmented data. We compare it with the BERT model trained only on the original training set. We report F1 scores on test set in Table 4.9. We observe a boost of 4% using our synthetic training data, validating our hypothesis in the in-domain setting.

Table 4.9. Case Study: We evaluate our context generators in in-domain and out-of-domain settings. In both cases, we observe substantial improvement in the performance demonstrating the effectiveness of our method.

Setting	Model	F_1 score
In-domain	BERT	32.11
	ConDA	36.74
Out-of-domain	BERT	14.96
	ConDA	25.31

Out-of-domain Analysis. In this experiment, we validate our hypothesis in the out-of-domain setting i.e. the domain of target dataset is different than the QA dataset used for QAC fine-tuning. We follow our proposed pipeline and consider SQuAD as the QA dataset for QAC fine-tuning and NewsQA as the target dataset. We partition NewsQA dataset into 1000 (question, answer, context) triplets for domain adaptation, 17000 (question, answer) pairs for context generation, and test on 10000 samples. We fine-tune GPT2-medium on SQuAD to obtain general context generator and adapt to the NewsQA domain by training it further on 1000 question, answer, context triplets from NewsQA. Using the target task context generator, we generate contexts for 17000 question, answer pairs, augment it to the training set, and train `BERT-base-uncased` QA model on the augmented data. From F1 scores reported in Table 4.9, we can observe more than 10% improvement in the performance, demonstrating the efficiency of our method in out-of-domain setting.

4.5 Summary

In this chapter, we propose to train generative language models to be context generators for a given question and answer. To facilitate this, we use question answer as a format and utilize QA datasets for training generative language models into context generators. We view sentiment and topic classification tasks in question-answer form and generate contexts using our fine-tuned generative language models. These generated contexts are used as synthetic training data to augment existing few-shot data for training a classifier. Extensive experiments on multiple sentiment and topic classification datasets demonstrate strong performance of our method in few-shot and zero-shot settings.

Chapter 4, in full, is a reprint of the material as it appears in Dheeraj Mekala, Tu Vu, Timo Schick, and Jingbo Shang. 2022. Leveraging QA Datasets to Improve Generative Data Augmentation, in Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 9737–9750, Association for Computational Linguistics. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 5

TOOLVERIFIER: Generalization to New Tools via Self-Verification

5.1 Introduction

Incorporating external tools into large language models (LLMs) enhances their real-world applicability (Schick et al., 2023; Shen et al., 2023; Song et al., 2023). Many tools exist in the form of APIs (Xu et al., 2023b; Tang et al., 2023; Hsieh et al., 2023; Schick et al., 2023; Qin et al., 2024), machine learning models (Shen et al., 2023; Patil et al., 2024), and other functions (Gou et al., 2024). Nevertheless, the evolving landscape of existing tools and APIs, marked by frequent parameter updates and the daily introduction of new tools, poses a challenge for generalization. LLMs must quickly adapt to these changes and generalize to previously unseen tools without additional fine-tuning or extensive human input.

Several recent studies enable tool usage by fine-tuning LLMs on real (Schick et al., 2023; Qin et al., 2024; Patil et al., 2024) or synthetic tools (Tang et al., 2023), equipping them to effectively utilize tools present in the training data with a high success rate. Currently, the integration of unseen tools into LLMs relies on providing them with few-shot demonstrations that contain examples of user instructions and corresponding tool calls (Patil et al., 2024; Tang et al., 2023). However, these prompting-based approaches still struggle to accurately generate a complete tool call from a set of unseen tools.

To address these challenges, we propose TOOLVERIFIER, a self-verification method

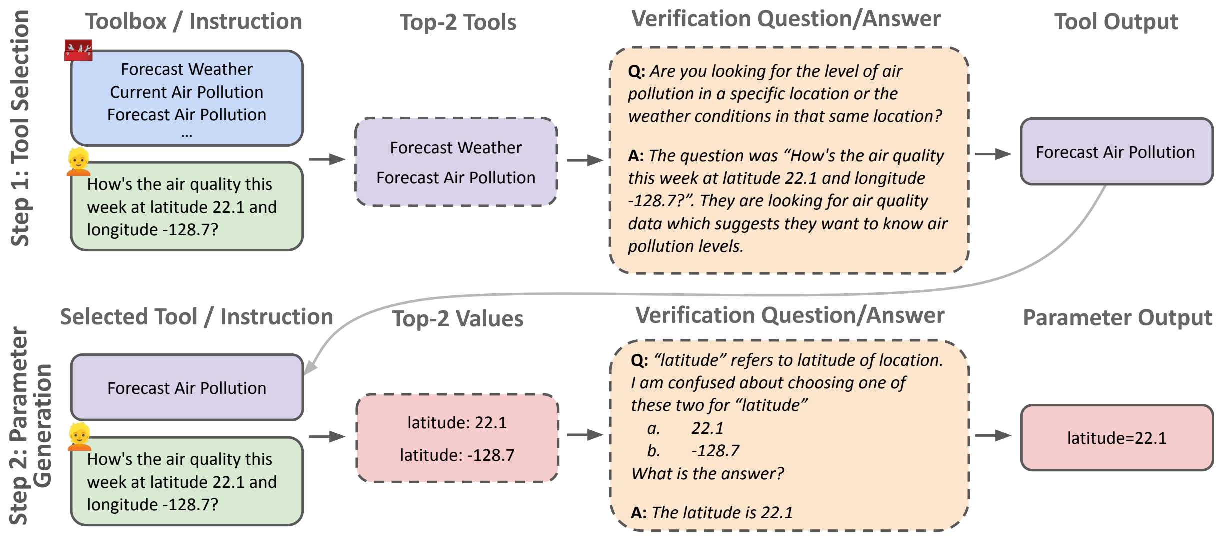

Step 1: Tool Selection
Toolbox / Instruction
Forecast Weather
Current Air Pollution
Forecast Air Pollution
...
How's the air quality this week at latitude 22.1 and longitude -128.7?
Top-2 Tools
Forecast Weather
Forecast Air Pollution
Verification Question/Answer
Q: Are you looking for the level of air pollution in a specific location or the weather conditions in that same location?
A: The question was "How's the air quality this week at latitude 22.1 and longitude -128.7?". They are looking for air quality data which suggests they want to know air pollution levels.
Tool Output
Forecast Air Pollution
Step 2: Parameter Generation
Selected Tool / Instruction
Forecast Air Pollution
How's the air quality this week at latitude 22.1 and longitude -128.7?
Top-2 Values
latitude: 22.1
latitude: -128.7
Verification Question/Answer
Q: "latitude" refers to latitude of location. I am confused about choosing one of these two for "latitude"
a. 22.1
b. -128.7
What is the answer?
A: The latitude is 22.1
Parameter Output
latitude=22.1

Figure 5.1. Overview of TOOLVERIFIER. Starting with a candidate tool list and a user instruction, TOOLVERIFIER initially identifies the top two tools. Subsequently, it generates a verification question by contrasting the selected tools and answers it. Finally, this information is appended to the context, leading to the final tool choice. The parameter generation follows a similar pipeline, wherein two candidate values are obtained for each parameter (*latitude* in the above figure). Subsequently, the verification question is used to finalize the parameter value.

tailored for tool-use scenarios, capable of discerning between candidate tools and their respective parameters through verification questions. To achieve this, we decompose the tool call generation task into two distinct sub-tasks: (1) *tool selection*, given a user instruction, the most suitable tool is selected from a library of options, and (2) *parameter generation*, the appropriate parameters for the selected tool are then generated. Crucially, we propose verification for each sub-task, to both improve sensitivity and to curb error propagation. Figure 5.1 shows an overview of each sub-task.

In the tool selection stage, our model must choose one tool among multiple options, given only the description of the tool. To facilitate learning how to choose the appropriate tool, we curate a high-quality, model-generated, synthetic training dataset containing tools, their descriptions, and user instructions.[1] This dataset comprises 173 synthetic tools with corresponding descriptions, 555 samples, each involving reasoning about the tool's usage. We

[1]The dataset is available at https://github.com/facebookresearch/ToolVerifier

then use this dataset to fine-tune a Llama-2 70B model (Touvron et al., 2023b) to select the correct tool for an instruction given only a set of tool names and their descriptions, allowing the model at test time to select from tools never seen during training. After the tool is selected, parameters are generated for the selected tool call, which is achieved through few-shot prompting with demonstrations corresponding to the chosen tool.

Self-verification is used at each step to reduce error propagation and enhance overall performance. As shown in Figure 5.1, for tool selection verification, we extract the top two predictions from the fine-tuned model. A verification question is then generated *contrasting the two options* via 0-shot prompting, enabling the model to focus on a fine-grained decision where the answer aids in selecting one tool from the top two predictions. The model answers the question, and the context is updated by appending this answer to the user instruction, to guide tool selection. A similar approach is adopted for verifying the parameter generation.

We evaluate our approach on 4 tasks from the publicly available ToolBench benchmark which tests generalization to 17 unseen real-life APIs. TOOLVERIFIER demonstrates a noteworthy 22% improvement over few-shot prompting baselines. The proposed self-verification mechanism contributes an improvement of 8%, underscoring its pivotal role in boosting overall performance.

5.2 Related Work

Self-Verification

Iterative improvement of LLMs typically involves prompting an LLM to provide feedback on given generated facts or answers and subsequently refining their outputs (Madaan et al., 2023; Shridhar et al., 2023; Lu et al., 2023) which has also been shown to reduce hallucination (Dhuliawala et al., 2024). Additionally, some studies involve the fine-tuning of custom LLMs to better accommodate feedback (Yu et al., 2024; Shridhar et al., 2024; Zhang et al., 2023), aiming to enhance reasoning in chain-of-thought prompting for improved downstream performance. In this paper, we focus on tool usage, whereas previous works typically focus on generation. Our approach contrasts the choice between selecting options, whereas previous work typically verifies single facts or answers in responses.

Enabling Tool Use in LLMs

Many approaches have emerged for enabling tool usage in LLMs, involving techniques such as few-shot prompting with tool-use demonstrations across diverse tool categories, including APIs (Qin et al., 2024; Chen et al., 2023b), machine learning models (Shen et al., 2023; Patil et al., 2024), and code interpreters (Gao et al., 2023; Chen et al., 2023a). Additionally, several approaches advocate for fine-tuning LLMs on custom-generated datasets tailored for tool usage (Schick et al., 2023; Tang et al., 2023; Parisi et al., 2022; Xu et al., 2023b; Patil et al., 2024; Srinivasan et al., 2023; Yang et al., 2023). Recent works introduce tool documentation (Hsieh et al., 2023) and tool tokens (Hao et al., 2023) to facilitate tool usage. Despite the plethora of works focused on enabling tool usage in LLMs, to the best of our knowledge none has explored verification methods for this purpose. This paper aims to fill this gap by introducing multi-step contrastive verification.

LLMs for Data Generation

LMs have been used for generating training data for various tasks including classification (Mekala et al., 2021, 2022b), semantic similarity (Schick & Schütze, 2021b), and instruction tuning (Wang et al., 2022a; Honovich et al., 2023; Xu et al., 2023a; Taori et al., 2023). Several works (Tang et al., 2023; Qin et al., 2024; Tang et al., 2023; Schick et al., 2023; Patil et al., 2024; Srinivasan et al., 2023) have employed LLMs to generate synthetic tools or tool use data.

5.3 ToolVerifier

ToolVerifier chooses and calls a tool given a user instruction. It consists of the following steps:

1. Tool selection & verification – *selecting the tool from a library of tools.*

2. Parameter generation & verification – *generating the parameters for the tool call.*

For step (1) we generate synthetic data consisting of a library of tools, (instruction, tool) pairs, and reasoning notes explaining the correct choice of tool, see Figure 5.2. Fine-tuning on this data provides improved tool selection performance, even on new sets of tools. The selection process is then refined by verifying the choice between the top two competing choices by asking and answering *contrastive verification questions*, see Figure 5.3.

For step (2) we use few-shot prompting given demonstrations of the actual tool. We again verify two competing likely generations.

5.3.1 Tool Selection Dataset Generation

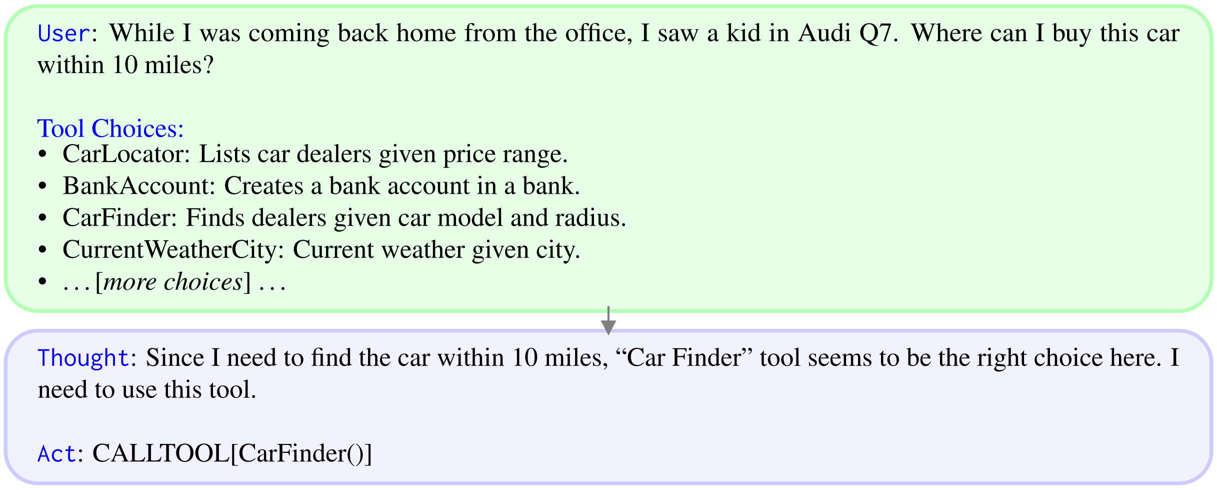
Our first goal is to train a language model capable of selecting an appropriate tool for a given user instruction by reasoning about a candidate list of tools solely based on their names and descriptions. We intentionally exclude demonstrations for tool selection in our approach to handle a larger set of tools in one go, using only their names and descriptions. In this section, we elaborate on the process of creating the training dataset for training such a tool selection language model.

Since the primary objective in this step is to select the correct tool (but not execute the tool call), synthetically generated tools and their corresponding descriptions can easily be used in this setting, as we do not require their actual inner workings (in order to execute them). In our generated dataset, each training sample is thus composed of a user instruction, a candidate set of tools that includes the ground truth tool, and a reasoning note elucidating the correct choice of tool. An illustrative training sample is given in Figure 5.2.

Synthetic Tool Library Generation

Generation Procedure

We generate a set of synthetic tools along with their corresponding descriptions, which are used to build the training examples. We start by first manually annotating a "seed set" of


User: While I was coming back home from the office, I saw a kid in Audi Q7. Where can I buy this car within 10 miles?

Tool Choices:
- CarLocator: Lists car dealers given price range.
- BankAccount: Creates a bank account in a bank.
- CarFinder: Finds dealers given car model and radius.
- CurrentWeatherCity: Current weather given city.
- . . . [*more choices*] . . .

Thought: Since I need to find the car within 10 miles, "Car Finder" tool seems to be the right choice here. I need to use this tool.

Act: CALLTOOL[CarFinder()]

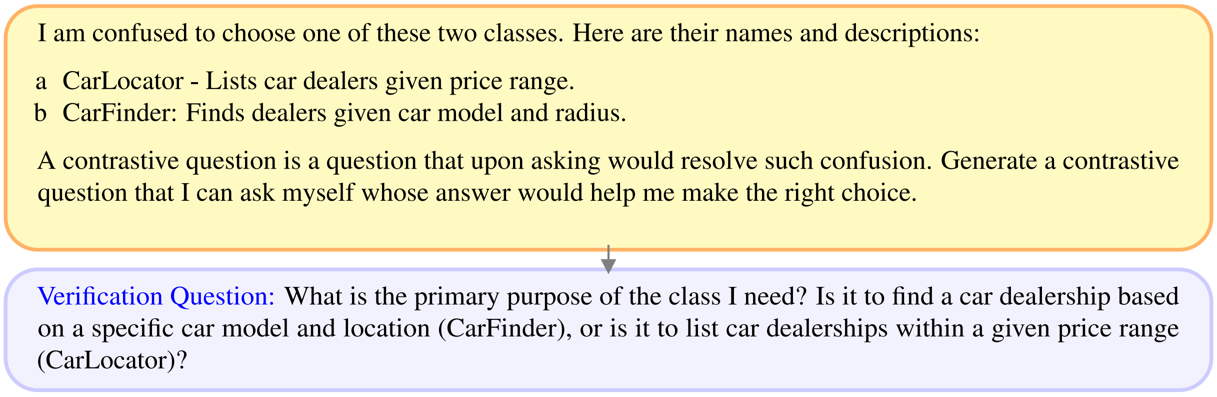
Figure 5.2. Illustrative training example from our synthetically constructed tool selection dataset `ToolSelect`. Given a user instruction and a set of tools to choose from, the output consists of reasoning notes ("Thought") and the final tool selection ("Act").

eight tools and their descriptions. Subsequently, we employ the Llama-65B (Touvron et al., 2023a) model to generate additional tools using few-shot prompting with the manually annotated tools (specified in Appendix C.4.1). This process then involves multiple iterations of prompting with different random seeds, where the tools generated in each iteration are integrated into the prompt for subsequent iterations to generate more diverse tools. Specifically, in each iteration, for every newly introduced tool, we identify the most similar tool in the prompt based on cosine similarity using RoBERTa sentence similarity (Reimers & Gurevych, 2019). We replace the most similar tool in the prompt with the new addition, ensuring a balanced diversity of tools in the prompt. Using this iterative approach, we generate a total of 60 tools.[2] It is noteworthy to highlight that this process yields a diverse set of tools from various domains including travel, banking, and calendar, with almost no manual effort.

Generating Challenging Tool Sets

In generating these synthetic tools, we endeavor to have a tool set that is diverse, but also sufficiently challenging. An overly simplistic training set would contain only easy choices (e.g., a weather tool versus an email tool) and this would impede the model's ability to generalize to

[2]These tools were manually reviewed, and 7 duplicates were removed.


I am confused to choose one of these two classes. Here are their names and descriptions:

a CarLocator - Lists car dealers given price range.
b CarFinder: Finds dealers given car model and radius.

A contrastive question is a question that upon asking would resolve such confusion. Generate a contrastive question that I can ask myself whose answer would help me make the right choice.

Verification Question: What is the primary purpose of the class I need? Is it to find a car dealership based on a specific car model and location (CarFinder), or is it to list car dealerships within a given price range (CarLocator)?

Figure 5.3. Verification method for tool selection: a constrastive question is generated that can then be answered to help discern among the top two predicted tools.

challenging instances during test time. To address this, we generate two *related tools* for each of our previously generated 60 tools. Related tools are defined as tools closely resembling a given tool but differing in either functionality or parameters. For instance, “*Bank account for a person name* ” and “*Bank account for an account number*” are related tools. We use only the tool names, and not the descriptions, for generating related tools. After manually annotating related tools for our seed set of eight tools, we generate two related tools for each of the remaining tools with few-shot prompting with these examples, as indicated in Appendix C.4.3.

Finally, after manual inspection and curation, our dataset contains a total of 173 tools.

Generating Training Examples

Using the generated tool library, we can now generate training examples for our tool selection dataset. This requires generating inputs (instructions), curating candidate lists of tools, and generating outputs (reasoning notes that explain which tools should be selected, and actions to call those tools).

Generating Instructions

We first manually annotate three instructions per tool for the seed set of eight tools. Using these examples, we generate three instructions per tool for all remaining tools by few-shot prompting Llama-2 70B.

Curating Candidate List of Tools

For each generated instruction, a candidate list of tools is created by randomly selecting 7 tools and adding the original ground truth tool for which we generated the instruction. To introduce complexity, for a subset of the training set, we deliberately create challenging samples by restricting the candidate set to include only the ground truth tool and its related tools. This deliberate selection aims to increase the difficulty level, as distinguishing among these options is inherently more challenging than with randomly selected tools from the entire set.

Generating Target Outputs

After generating the set of instructions along with their respective ground truth tool and a candidate list of tools, we create a reasoning note for each sample elucidating the rationale behind the selection of the ground truth tool, which becomes the target output for that training example (see Figure 5.2). Such reasoning notes have been observed to enhance reasoning abilities (Wei et al., 2022b; Yao et al., 2023; Lanchantin et al., 2023). Reasoning note generation is accomplished by prompting Llama-2-Chat-70B with the instruction, list of tools, and the ground truth tool, and asking the model why the tool was chosen. The exact prompt used is provided in Appendix C.4.2.

Our final dataset, called `ToolSelect`, thus contains 555 samples for our 173 tools, of which 75 samples are *hard* examples, featuring candidate tool sets that contain only the ground truth tool and its related tools.[3] The average number of candidate tools per instruction is 7.34 with minimum and maximum number of candidate tools being 2 and 8. The average length of a reasoning note is 1054 characters.

The goal of this dataset is to enable generalization capabilities to a wide range of possible tools and tool libraries, and thus to demonstrate effectiveness across diverse scenarios. During training, the user instruction and tool list in each sample have masked labels, and hence, they do not contribute to the loss and are not learned.

[3]The data was manually reviewed, and 56 noisy and duplicate samples were removed.

5.3.2 Tool Selection Verification

Despite our model being fine-tuned on the above dataset, tool selection mistakes can still happen, particularly for related tools that are hard to differentiate. Crucially, we observe that those tool selection predictions typically appear as the top few predictions – but selection between them is challenging.

At inference time, we thus perform the following procedure. Given an instruction:

- First, we use the fine-tuned tool selection model to zero-shot select a tool.
- We then remove the initially selected tool from the candidate set of tools, and generate a second prediction.
- We construct a verification question to make a fine-grained decision between the model's top two selections.

We employ Llama-2-Chat-70B to generate a contrastive verification question, where the prompt asks the model to ask a question that emphasizes the distinctions between candidate tools given their names and descriptions (see Appendix C.4.4 for the exact prompt used and Figure 5.3 for an instantiation of it). Self-asking the model regarding its predictions has been noted to reduce hallucinations (Press et al., 2023; Dhuliawala et al., 2024), suggesting that posing such verification questions could assist the model in validating its predictions. Since only names and descriptions are used for generating contrastive questions, they can be generated offline and utilized as needed to make the method more efficient. The answers to these contrastive questions are obtained by further prompting Llama-2-Chat-70B, and these are appended to the context. Finally, we select the tool by using our fine-tuned Llama-2 70B model, with the top-two tools as candidates. As the verification answer to the question is in the context this can help it select the right tool.

5.3.3 Parameter Generation & Verification

Parameter Generation

Following tool selection, we generate parameters for the selected tool through few-shot prompting with Llama-2 70B, utilizing demonstrations specific to the selected tool, which are assumed to be provided. Note that we do not use our synthetic tool selection dataset for parameter generation since the dataset does not contain this subtask. This procedure is only done with real tools at inference time, without prior finetuning.

Parameter Verification

The generated parameters are then subjected to verification before finalizing the set, resulting in the final tool call. To validate the generated parameters, we obtain a second set of parameter predictions. These can be acquired using sampling or an alternative model for diverse options; in our experiments, we employ ReAct-style prompting (Yao et al., 2023) with Llama-2 70B to obtain them. Then, for each individual parameter, we formulate a multiple-choice question to contrast the two predictions and further prime Llama-2-Chat-70B to make a definitive choice between them, providing the parameter description and user instruction as indicated in Appendix C.4.5. The final parameter predictions are then aggregated to construct the tool call by few-shot prompting Llama-2 70B as in Appendix C.4.7.

5.4 Experiments

In our experiments, we assess the effectiveness of our method using publicly available real-life tools.

5.4.1 Tasks

We evaluate our proposed method on four tool-calling tasks: Weather, Cat, Home and Booking from ToolBench (Xu et al., 2023b) that involve using the REST APIs. The Weather, Home, and Cat tasks each comprise 100 evaluation samples, while the Booking task contains

120 samples. Each task includes API documentation, parameter descriptions, user instructions, and the corresponding ground truth API call pairs.

For each task, there are multiple tools available, where the entire benchmark consists of a total of 17 tools. However, instead of evaluating each task individually, we make it more challenging by pooling together all available tools. In other words, for each user instruction, the model is provided a candidate list of 17 tools. The ToolBench benchmark with 17 tools presents an ideal balance between maximizing the number of tools that could be accommodated within the context window without requiring the use of a retriever. By eliminating the dependency on a retriever, we could independently evaluate the impact of self-verification on performance. We follow the evaluation protocol set by the benchmark and use success rate as the metric, where the success rate of a predicted tool call is 1 if its API response exactly matches the response from the ground truth API call.

5.4.2 Baselines

We conduct a comparison with various tool-augmented LLMs and prompting baselines using Llama-2 70B and Llama-2-Chat-70B. Specifically, for tool-augmented LLMs, we compare with ToolLLM 7B (Qin et al., 2024), NexusRaven-V2 13B[4], and Qwen1.5-Chat-72B (Bai et al., 2023)[5]. ToolLLM, NexusRaven-V2, and Qwen1.5 utilize API documentation to generate tool calls corresponding to a given instruction.

For prompting baselines, we try two distinct approaches: (1) Single-step, where the model is prompted directly for an API call with a single demonstration per tool; and (2) Two-step, where we decompose the process into tool selection and parameter generation, prompting the model individually for each step, as in TOOLVERIFIER.

The Single-step method uses 1-shot single demonstrations of each of the (17) tools to accommodate the prompt within the context size.

[4]https://nexusflow.ai/blogs/ravenv2

[5]We attempted comparing with ToolAlpaca (Tang et al., 2023), however, it led to context overflow.

For the Two-step method, we consider two variants for the tool selection stage:

- *0-shot:* We use a 0-shot prompt that asks to select from the list of tools, without any demonstrations for tool selection. See C.4.6 for the exact prompt.
- *1-shot:* We show one demonstration per tool: a user instruction and corresponding tool name.

For parameter generation in the Two-Step method, we use three demonstrations for the selected tool.

5.4.3 TOOLVERIFIER Details and Ablations

Our model is denoted as TOOLVERIFIER. For tool selection it uses 0-shot prompting with Llama-2 70B fine-tuned on our synthetic `ToolSelect` dataset to select two tools and finalize one through our proposed contrastive-question-based tool verification. Subsequently, we generate two sets of parameters by employing standard few-shot and ReAct-style prompting Llama-2 70B with three demonstrations, and finalize the parameter set using our proposed parameter verification.

We additionally compare against ablated versions of our method: with tool selection verification only (but not parameter verification), with parameter selection verification only (but not tool verification), and without verification (in either stage).

5.4.4 Experimental Results

Tool Call (Selection + Parameters)

The complete tool call performance results are presented in Table 5.1. Our approach, TOOLVERIFIER, outperforms all baselines both on average and individually across all tasks. TOOLVERIFIER outperforms all compared tool-augmented LLMs by a significant margin. Comparing TOOLVERIFIER with Single-Step 1-shot highlights the challenges in generating complete tool calls at once, emphasizing the efficacy of the two-step decomposition. It also surpasses Single-Step 1-shot and Two-Step 1-shot tool baselines by a substantial margin of more than 50 points on the challenging Booking task.

Table 5.1. Tool call (tool selection + parameter generation) results. We report percentage (%) success rate for each task. Our fine-tuned Llama-2 70B model ToolVerifier, even without verification, results in higher performance compared to the baselines. Our proposed verification mechanism further improves the success rate by 8 points – with both types of verification, for tool and parameter selection, each giving a separate boost in performance.

Method	Weather	Booking	Home	Cat	Average
Tool-Augmented LLMs					
ToolLLM 7B	18.00	0.00	0.00	11.00	6.90
NexusRaven-V2 13B	55.00	27.50	43.00	82	50.71
Qwen1.5-Chat-72B	74.00	55.00	52.00	89	66.90
Prompting Baselines					
Single-Step Llama-2 70B (1-shot)	70.00	7.50	85.00	83.00	58.81
Two-Step Llama-2 70B (1-shot tool selection)	80.00	34.17	85.00	78.00	67.62
Two-Step Llama-2-Chat-70B (0-shot tool selection)	77.00	64.17	84.00	83.00	76.43
ToolVerifier (without verification)	76.00	82.50	85.00	82.00	81.43
ToolVerifier (tool selection verification only)	84.00	82.50	85.00	83.00	83.57
ToolVerifier (param selection verification only)	81.00	84.17	88.00	96.00	87.14
ToolVerifier (tool verification+param verification)	**90.00**	**84.17**	**88.00**	**97.00**	**89.52**

A comparative analysis between ToolVerifier with and without parameter verification illustrates that parameter verification significantly enhances performance, showing improvements of up to 14 points in the Cat task and 6 points in the Weather task, leading to an average improvement of 6 points across all tasks. Similarly, the comparison between ToolVerifier with and without tool verification demonstrates that tool verification contributes significantly to the performance, such as up to 8 points in the Weather task. Notably, both types of verification help, each giving a separate boost, as shown by comparing the without verification results to tool selection verification only and tool+parameter verification. These results underscore the significance of verification in both steps for the tool call success.

Tool Selection Only

We report the performance of tool selection (choosing the tool correctly, but without generating parameters) in Table 5.2. ToolVerifier outperforms all baselines on average and individually across the majority of tasks as well. ToolVerifier performs better than almost all compared tool-augmented LLMs, demonstrating its superior performance. While

Table 5.2. Tool selection results. We report accuracy in percentage (%) for each task. Our fine-tuned Llama-2 70B model ToolVerifier, even without verification, demonstrates superior performance compared to prompting-based baselines, with a higher average performance. Our proposed tool selection verification mechanism contributes another 2.5% improvement in accuracy on average.

Method	**Weather**	**Booking**	**Home**	**Cat**	**Average**
Tool-Augmented LLMs					
ToolLLM 7B	27.00	22.00	84.00	26.00	38.90
NexusRaven-V2 13B	84.00	93.33	**100.00**	**98.00**	93.81
Qwen1.5-Chat-72B	**93.00**	95.00	99.00	96.00	95.71
Prompting Baselines					
Single-Step Llama-2 70B (1-shot)	79.00	43.30	**100.00**	**98.00**	78.32
Two-Step Llama-2 70B (1-shot tool selection)	86.00	45.00	**100.00**	92.00	79.05
Two-Step Llama-2-Chat-70B (0-shot tool selection)	83.00	75.80	99.00	97.00	88.09
ToolVerifier (without verification)	82.00	**98.33**	**100.00**	96.00	94.28
ToolVerifier (tool selection verification)	91.00	**98.33**	**100.00**	97.00	**96.67**

Qwen1.5 performs well at selecting the right tool, it struggles to generate parameters correctly. In contrast, our method's focus on parameter generation, facilitated by the two-step approach, yields improved overall performance as shown in Table 5.1. A comparative analysis between ToolVerifier with tool selection verification and without underscores the substantial enhancement in performance achieved through the verification process. Specifically, in tasks such as Weather and Home, we observe that the verification procedure not only improves performance in specific examples of lower baseline performance, but also does not adversely affect cases where verification may be unnecessary.

ToolVerifier (both with and without verification) shows that our zero-shot Llama-2 70B fine-tuned on our synthetically generated dataset performs better than other baselines, including a 0-shot Llama-2-Chat-70B, with an improvement of up to 6 points. The average number of candidate tools per instruction in the generated training data for tool selection is 7.34 which is notably smaller than the 17 tools encountered during test time. This difference underscores the generalization capability of our method, demonstrating its effectiveness across diverse scenarios. The performance of 1-shot baselines reveals the difficulty in selecting the

Table 5.3. Tool verification and parameter verification improve tool call success rate for tool-augmented LLMs. We report percentage (%) success rate for each task. Our proposed verification mechanism significantly improves the success rate of all tool-augmented LLMs.

Method	**Weather**	**Booking**	**Home**	**Cat**	**Average**
ToolLLM 7B	18.00	0.00	0.00	11.00	6.90
ToolLLM 7B + Tool, Param Verification	23.00	7.50	9.00	15.00	13.33
NexusRaven-V2 13B	55.00	27.50	43.00	82.00	50.70
NexusRaven-V2 13B + Tool, Param Verification	78.00	34.17	46.00	84.00	59.29
Qwen1.5-Chat-72B	74.00	55.00	52.00	89.00	66.90
Qwen1.5-Chat-72B + Tool, Param Verification	76.00	57.50	59.00	91.00	70.24

appropriate tool from an unseen set using prompting-based approaches. In contrast, fine-tuning the model on our synthetically generated dataset with examples of using a diverse set of tools significantly improves tool selection accuracy. Moreover, the verification procedure further improves tool selection performance by an additional 2.4 points on average.

5.5 Analysis

5.5.1 Self-verification improves tool-augmented LLMs

Our proposed self-verification method does not require any specific training process. To demonstrate its effectiveness on tool-augmented LLMs, we experiment with ToolLLM 7B, NexusRaven-V2 13B, and Qwen1.5-Chat-72B. We obtain two sets of predictions as in TOOLVERIFIER, where the first predicted tool is removed from the set of tools to obtain the second prediction. After tool verification, we identify the final selected tool. We obtain two parameter predictions using two different sampling parameters while generation. We then perform parameter verification to finalize the parameters and construct the tool call. The complete tool call success rate comparison, with and without self-verification, is presented in Table 5.3. We observe a significant improvement in average performance: 6 points for ToolLLM-7B, 9 points for NexusRaven-V2 13B, and 4 points for Qwen1.5-Chat-72B. In certain tasks, such as the Weather task, the success rate of NexusRaven-V2 improved by 23 points through self-verification. This demonstrates that self-verification can be effectively applied to tool-augmented LLMs, enhancing their performance. The tool selection results are in Appendix C.1, where we also note significant improvements in performance post tool verification.

5.5.2 Verification Question Analysis

Qualitative Analysis

Verification questions should ideally reference the distinguishing characteristics between two given tools in order to best help the model consider the differences between the two choices. This capability is particularly crucial for closely related tools. For instance, the tools "Forecast Air Pollution" and "Current Air Pollution" both provide air pollution data, but for future and current times, respectively. Verification question generation by Llama-2-Chat-70B identifies this nuanced difference and articulates it in the verification question: *Are you looking for data on the current air pollution levels in a specific location, or do you need to forecast the air pollution levels for a future date in that location?* Responses to such questions precisely address the identified distinction. An example response is: *"It appears that the user is looking for current air pollution data for a specific location with latitude -24.7 and longitude -57.3. Therefore, the answer is: A. Retrieve current air pollution data for a specific location."* Inserting this response into the context improves tool selection accuracy, guiding the model towards the correct choice. For more distinct tools, the model captures higher level differences. For example, for "Forecast Air Pollution" and "Get favorite cat images", the generated question is: *Which aspect are you more interested in: predicting environmental air quality or exploring feline visuals?*

Significance of Contrastive Questions

To demonstrate the significance of contrastive-question-based verification, we conduct an experiment by zero-shot prompting Llama-2-Chat-70B to choose one tool from the top-2 without employing a verification question. Instead, we present the names and descriptions of the top-2 tools and frame it as a multiple-choice question, asking Llama-2-Chat-70B to make a selection. We experiment on the Weather task and the accuracy of Llama-2-Chat-70B is 70% whereas the accuracy of contrastive question-based verification is 91 %. This significant enhancement over straightforward prompting illustrates effectiveness of contrastive questions.

Instruction-Conditioned Verification

In our proposed approach we generate verification questions using solely the names and descriptions of the top-2 selected tools, see Figure 5.3. We compare this to conditioning on the user instruction as well, by adding it to the prompt. Conditioning on the instruction during verification still shows improvement over the no-verification baseline (89 versus 82), however, slightly decreases performance compared to the non-user-conditioned verification, dropping accuracy from 91 to 89, perhaps because the decision is biased to be more similar to the original top choice being verified, which was also based on the instruction. Note that, using only names and descriptions has the benefit that the questions can be precomputed and cached.

5.5.3 Parameter Verification Error Analysis

In the parameter verification step, we identify a consistent pattern in errors while answering the verification questions, predominantly involving common sense errors where the model tends to hallucinate values instead of adhering to the user instruction, which is also observed in Mekala et al. (2023). A notable example of such errors occurs with the *min-price* parameter in Booking tool, which signifies the minimum price the user is willing to pay for a booking. In 5 instances out of 19 wrong predictions for the Booking task, when the user specifies only their maximum budget, the model generates the maximum value for the min-price parameter rather than 0. Similar errors are observed with the *min-area* parameter in the Home task. In 4 instances out of 12 mistakes, when the user expresses the desire for a home given only a maximum area, the model incorrectly predicts the mentioned value as the minimum, instead of using 0.

5.5.4 Synthetic Training Data Analysis

We analyze our synthetic `ToolSelect` training data through various ablations, with results on tool selection for the Weather task given in Figure 5.4.

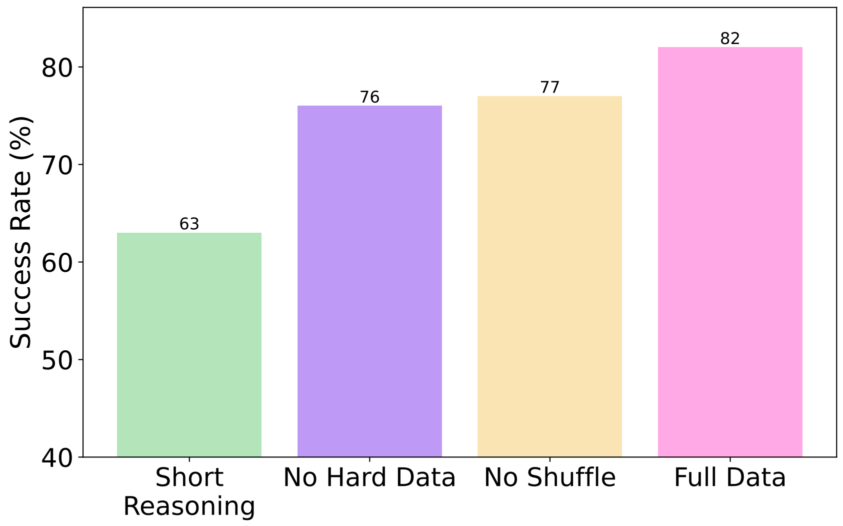


Figure 5.4. We analyze various aspects of our synthetic `ToolSelect` training data including the ordering of the candidate tool list ("No Shuffle"), difficulty level ("No Hard Data"), and the length of reasoning notes ("Short Reasoning"). We find samples with longer reasoning notes, difficult samples, and randomly ordered candidate tool lists contribute to high performance ("Full Data").

Challenging training samples (samples that have a candidate tool list containing related tools to the ground truth tool, see subsubsection 5.3.1) are found to improve generalization. To assess the impact of these challenging samples, we remove them and train a model solely with easier samples ("No Hard Data"). The results indicate a notable 6-point drop in performance after excluding the hard samples, highlighting their significance.

Next, we experiment by reducing the maximum reasoning note length from 480 tokens to 200 tokens ("Short Reasoning") and observe a significant drop in performance, up to 19 points. Shorter reasoning texts are significantly less helpful in guiding appropriate tool selection.

Lastly, we compare performance with different orderings of the candidate tool list. In the "No Shuffle" scenario, the ground truth tool in the training data is always positioned first. Implementing this ordering strategy results in a 5-point drop in performance, underscoring the significance of randomly shuffling the candidate tool list in the training data.

More studies regarding parameter generation-only performance, and prompts are detailed in Appendix C.2, C.4 respectively.

5.6 Summary

In this chapter, we present a self-verification method for enhancing the performance of tool calls for LLMs. This involves decomposing the tool call generation task into tool selection and parameter generation, where we apply verification at each step. Additionally, we open-source a synthetic dataset for improved reasoning and generalization to unseen tools. Experimental results on four tasks from the ToolBench benchmark demonstrate substantial improvements using our approach.

Chapter 5, in full, is a reprint of the material as it appears in Dheeraj Mekala, Jason E Weston, Jack Lanchantin, Roberta Raileanu, Maria Lomeli, Jingbo Shang, and Jane Dwivedi-Yu. 2024. TOOLVERIFIER: Generalization to New Tools via Self-Verification, in Findings of the Association for Computational Linguistics: EMNLP 2024, pages 5026–5041, Association for Computational Linguistics. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 6

Smaller Language Models are capable of selecting Instruction-Tuning Training Data for Larger Language Models

6.1 Introduction

Instruction tuning empowers large language models (LLMs) to generalize to novel tasks and instills an instruction-following characteristic, marking the initial stride towards aligning them for general use (Sanh et al., 2022; Chung et al., 2024; Wei et al., 2022a). This process involves fine-tuning language models with extensive sets of real (Mishra et al., 2021; Wang et al., 2022b) and/or synthetic instructions (Wang et al., 2022a; Honovich et al., 2023). Given that these datasets are typically vast, encompassing thousands of samples, the training costs associated with this approach are notably high.

Past research into the memorization effects of deep neural networks have revealed a tendency to memorize easy instances first and gradually learn more challenging instances towards the end (Arpit et al., 2017; Geifman et al., 2019; Zhang et al., 2021; Mekala et al., 2022a). Additionally, (Swayamdipta et al., 2020) show that ambiguous and hard samples in training data are sufficient for achieving good generalization. The process of data selection, wherein subsets are chosen from extensive training data to achieve superior performance, has attracted significant attention among researchers recently. Earlier studies involved manual feature engineering of

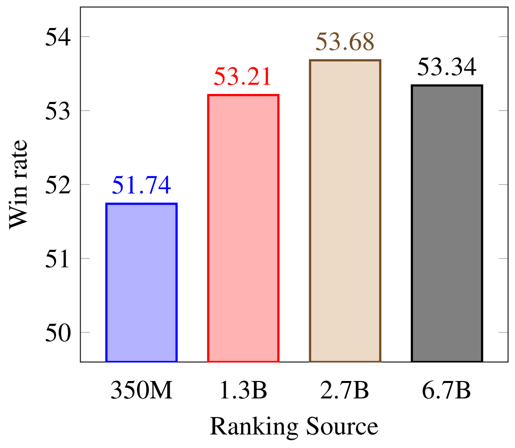


Figure 6.1. The win rate of OPT-13B model trained on 10% data sub-sampled by smaller OPT models (350M, 1.3B, 2.7B) from Alpaca Data, is compared against the OPT-13B model trained on the full dataset. All win rates exceed 50, indicating even a smaller 350M dataset can curate high-quality data for a larger 13B model.

various indicators from the data (Cao et al., 2024), training large custom models (Li et al., 2024), or employing closed LLMs like GPT-3.5 (Chen et al., 2024) for data selection.

In this paper, we delve into the measurement of sample difficulty from the model's perspective. Drawing inspiration from the learning order metric in (Mekala et al., 2022a), we propose a novel data selection method that utilizes the learning percentage as a difficulty metric that the model can use to self-rank its training data. Essentially, the more learning that occurs in earlier epochs, the easier the sample is considered. We then select the most difficult sample subsets based on this ranking and instruction-tune a language model. Our experiments involve two instruction-tuning datasets, Alpaca-Data (Taori et al., 2023), and Dolly (Conover et al., 2023), with performance measured using automated metrics such as AlpacaEval (Li et al., 2023b) and human evaluation.

Our main findings indicate that language models can autonomously select training data, achieving performance equal to or better than training on the entire dataset. Furthermore, this characteristic scales across different model sizes, ranging from smaller ones (1B) to larger ones (13B)[1] in parameters. As the size of the language model increases, we observe a consistent

[1]Due to limitations in our compute, the largest size we were able to train is 13B.

reduction in the minimum amount of data needed to surpass the performance of a model trained on the entire dataset. Interestingly, we observe that the data hardness also transfers across models, meaning samples considered difficult by smaller models are similarly challenging for larger models. Moreover, we note that this transferability improves with the size of the smaller model, eventually achieving comparable quality, beyond a size threshold, to that attained by self-selection conducted by larger models. Our study employs open-sourced models such as OPT (Zhang et al., 2022) and Llama-2 (Touvron et al., 2023b) to support these findings.

The remainder of the paper is structured as follows: initially, we describe the experimental setup encompassing the language models, the datasets employed, and the evaluation metrics utilized (section 6.3). Subsequently, we present our learning percentage-based difficulty metric and analyze it in detail (section 6.4). Following this, we optimize the proposed metric and introduce an equally effective, approximate, and faster metric (section 6.5). Ultimately, we analyze the challenging data identified through this metric (section 6.6).

We publicly release the code here[2].

6.2 Related Work

6.2.1 Instruction Tuning

Instruction tuning involves training LLMs to follow instructions (Sanh et al., 2022; Wei et al., 2022a; Chung et al., 2024). Numerous datasets have been curated for this purpose, comprising a multitude of samples (Mishra et al., 2021; Wang et al., 2022b). Notably, there is a recent surge in the emergence of synthetic instructions and datasets (Wang et al., 2022a; Honovich et al., 2023), each containing a substantial number of samples. As the datasets increase, we need to rethink data handling strategies from an efficiency standpoint (Sorscher et al., 2022), which we address in this paper.

[2]https://github.com/dheeraj7596/Small2Large

6.2.2 Data Selection

Prior data selection works in pre-training include (Tirumala et al., 2023), emphasizing the importance of diversity in sub-sampled data and advocating for the selection of prototypes from each cluster. (Abbas et al., 2023) extend this by removing semantic deduplicates in the training data. In the domain of instruction-tuning, (Cao et al., 2024) evaluate various indicators and apply a regression model for data selection. (Chen et al., 2024) leverage GPT-3.5 to derive difficulty ratings for individual data samples. (Li et al., 2024) propose an instruction-following difficulty metric for selection.

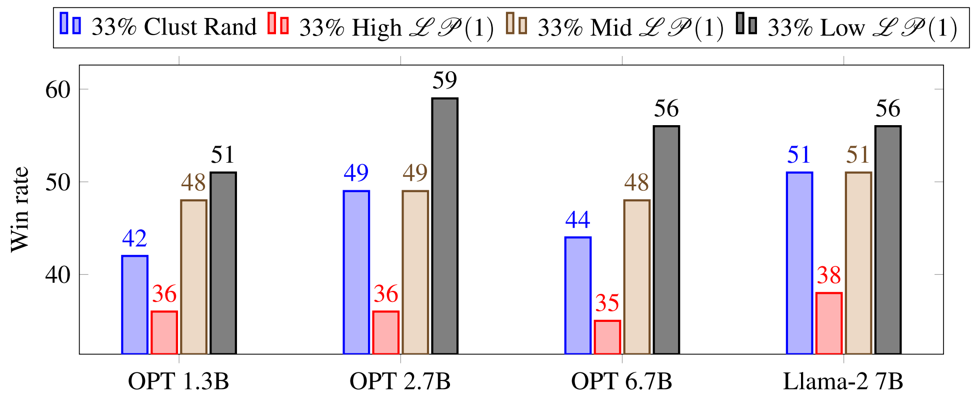
6.3 Experiment Setup

We design controlled experiments to empirically validate our assertions. Our experimental setup encompasses language models spanning various families and sizes, alongside multiple datasets, the specifics of which are detailed below.

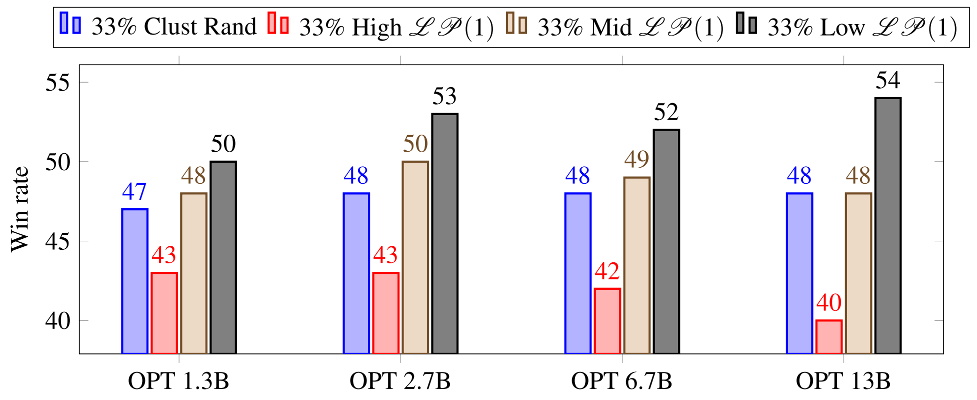
6.3.1 Language Models & Evaluation

We use OPT (1.3B, 2.7B, 6.7B, 13B) and Llama-2 (7B, 13B) for experiments. We fine-tune all models for three epochs on three NVIDIA A100 GPUs. For the comparison of language models, we employ AlpacaEval—an automated evaluator that tasks a larger language model with selecting the superior response from two LMs. AlpacaEval offers an evaluation set comprising 805 samples, designed to assess general instruction-following capabilities by combining data from various sources, including self-instruct (Wang et al., 2022a), anthropic helpfulness[3], open assistant (Kopf et al., 2023), Koala[4], and Vicuna (Chiang et al., 2023) evaluation sets. While AlpacaEval provides various options for the judge language model, we opt for GPT-3.5 (gpt-3.5-turbo-16k-0613) (OpenAI, 2023) due to its cost-effectiveness.

[3]https://huggingface.co/datasets/Anthropic/hh-rlhf/viewer/Anthropic--hh-rlhf/test
[4]https://github.com/arnav-gudibande/koala-test-set



(a) Dolly dataset



(b) Alpaca-Data

Figure 6.2. We partition datasets into three equal-sized buckets based on their $\mathscr{LP}(1)$ scores. We train a model per bucket and report its win rate against the one trained on the complete dataset. The model used to compute $\mathscr{LP}(1)$ scores and trained is depicted on the X-axis and the win rate on the Y-axis. We observe the model trained on the lowest $\mathscr{LP}(1)$ values (33% Low $\mathscr{LP}(1)$) exhibits superior performance compared to the others.

6.3.2 Data

We experiment on Alpaca-Data (Taori et al., 2023) and Dolly (Conover et al., 2023) datasets. Alpaca-Data comprises 52,000 samples generated through the self-instruct method by prompting text-davinci-003 with 175 human-written seed instruction-output pairs (Wang et al., 2022a). Dolly, on the other hand, consists of 15,000 human-generated samples.

6.4 $\mathscr{L}\mathscr{P}$ Learning Percentage as a Difficulty Metric

The concept of learning order (Dong et al., 2021; Mekala et al., 2022a) is designed to assess the quality of a sample in the context of a weakly-supervised classification problem (Mekala et al., 2022b; Mekala & Shang, 2020). The learning order of a data point is defined as the epoch at which it is learned during training, precisely when the model's predicted label aligns with the given ground truth. To adapt this concept to the text generation problem, we introduce the notion of learning percentage.

For a data point after epoch-i, the learning percentage is defined as the percentage drop in perplexity during epoch-i compared to the total drop in perplexity by the end of training. Assuming a language model is fine-tuned for n epochs, with $\mathscr{P}_i$ denoting the perplexity of a sample at the end of epoch-i and $\mathscr{P}_0$ indicating its perplexity at the beginning of training, the learning percentage $\mathscr{L}\mathscr{P}(i)$ at the end of epoch-i is mathematically defined as follows:

$$\mathscr{L}\mathscr{P}(i) = \frac{\mathscr{P}_{i-1} - \mathscr{P}_i}{\mathscr{P}_0 - \mathscr{P}_n} \tag{6.1}$$

A higher learning percentage at earlier epochs indicates that majority of the learning occurs during the initial epochs. Given the deep neural models typically learn easier samples initially and progress to more challenging samples later (Arpit et al., 2017; Geifman et al., 2019; Zhang et al., 2021; Mekala et al., 2022a), a higher learning percentage in the early epochs implies easy-to-learn samples. Since language models are known to learn most of the information in just one epoch (Komatsuzaki, 2019; Hoffmann et al., 2022; Zhang et al., 2022; Touvron et al., 2023b), we consider $\mathscr{L}\mathscr{P}(1)$ to rank the training data.

The diversity of training data is a pivotal attribute for achieving high quality and optimal performance (Sorscher et al., 2022; Tirumala et al., 2023). To enhance this diversity, we employ k-means clustering on the sentence embeddings generated by the all-MiniLM-L6-v2 model[5] on

[5]https://www.sbert.net

the entire training dataset, ensuring that each cluster contains a minimum average of 50 samples. As a result, the Alpaca-Data yields 1000 clusters with an average of 52 samples per cluster, while the Dolly dataset yields 300 clusters with an average of 50 samples per cluster. Subsequently, we rank the training data using $\mathscr{L}\mathscr{P}(1)$ in ascending order and select the top-$k\%$ of samples from each cluster, i.e., the samples that are learned the least in the first epoch.

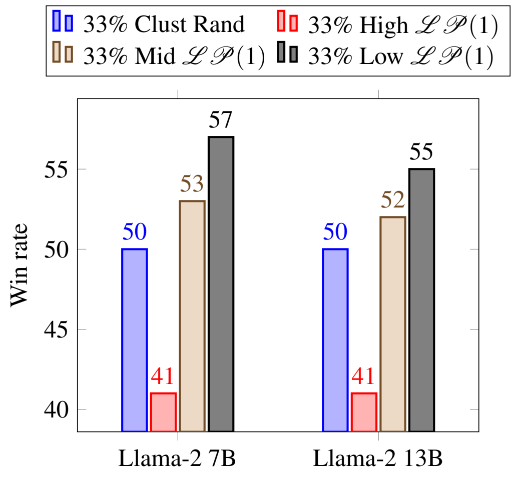


Figure 6.3. We partition Alpaca-Data into three equal-sized buckets based on their $\mathscr{L}\mathscr{P}(1)$ scores. The model used to compute $\mathscr{L}\mathscr{P}(1)$ scores and trained is on the X-axis and the win rate on the Y-axis. We observe the model trained on the lowest $\mathscr{L}\mathscr{P}(1)$ values (33% Low $\mathscr{L}\mathscr{P}(1)$) exhibits superior performance.

6.4.1 $\mathscr{L}\mathscr{P}(1)$based Data Selection

We calculate the $\mathscr{L}\mathscr{P}(1)$ scores of the training dataset and organize it in ascending order according to these scores. Subsequently, we partition the dataset into three equal buckets. To enhance the diversity, this partitioning is done per cluster. The bucket characterized by the lowest $\mathscr{L}\mathscr{P}(1)$ values (**33% Low** $\mathscr{L}\mathscr{P}(1)$) represents the most challenging data in each cluster, while the bucket with the highest values corresponds to the least challenging data (**33% High** $\mathscr{L}\mathscr{P}(1)$) in each cluster.

We train one model per bucket and calculate the win rate against the model trained on the complete training dataset. We also present the performance of the model trained on randomly selected 33% data from each cluster (**33% Clust Rand**) for reference. The AlpacaEval win rate scores of the Llama-2 7B and 13B models trained on individual buckets in Alpaca-Data are plotted in Figure 6.3. Similarly, the win rate scores of OPT 1.3B, 2.7B, 6.7B, and Llama-2 7B models on the Dolly dataset are shown in Figure 6.2(a). We observe that models trained on the bucket associated with the lowest $\mathscr{LP}(1)$ scores (33% Low $\mathscr{LP}(1)$) consistently achieve scores exceeding 50, indicating that training on challenging samples alone is adequate for a robust instruction-tuning model. Furthermore, our analysis reveals that the model trained on the lowest $\mathscr{LP}(1)$ scores (33% Low $\mathscr{LP}(1)$) consistently outperforms those trained on mid and high buckets by a significant margin. For example, in Figure 6.2(a), OPT 2.7B trained on the low bucket outperforms the mid bucket by 10 points and the high bucket by 23 points. This underscores a compelling argument that leveraging difficult data yields more favorable outcomes compared to training on easier datasets. The 33% High bucket results in worse performance than random selection for all models, highlighting that easy samples alone are insufficient.

Performance vs Scale

To investigate the variation of this trait with scale, we plot the win rate scores of OPT (1.3B, 2.7B, 6.7B, 13B) models trained on each bucket within the Alpaca-Data in Figure 6.2(b). Remarkably, all models trained on the low bucket consistently achieve win rates exceeding 50, surpassing those of the corresponding mid and high buckets. This consistent trend across different model scales underscores the robustness of the observed pattern.

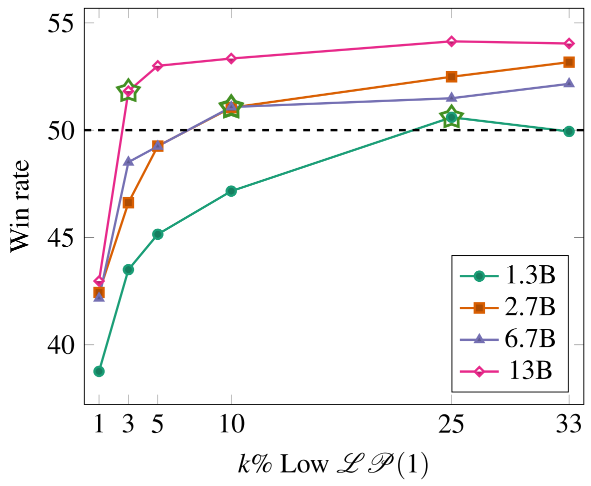
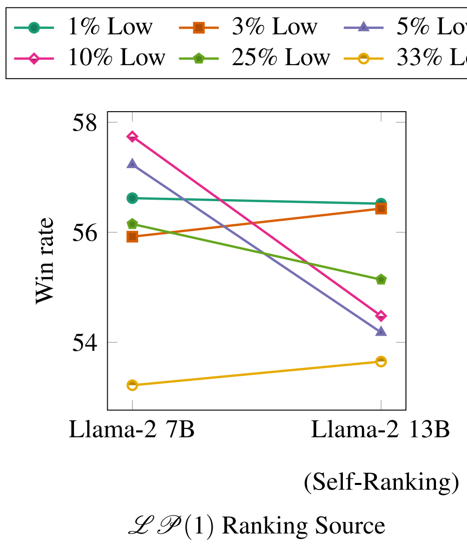


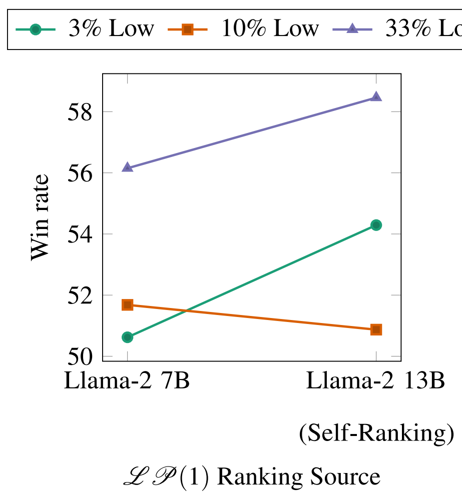
Figure 6.4. We consider Alpaca-Data, vary the percentage of data selected, and plot the win rate of OPT models, trained on the selected data in comparison to models trained on the complete dataset. The minimum percentage of data necessary for each model to surpass the 50% threshold is highlighted with ✩.

Larger models need fewer samples

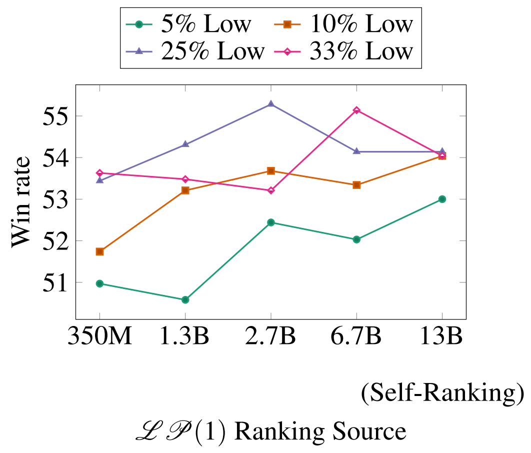
To further analyze the required amount of difficult data necessary for training high-quality instruction-tuned models of varying sizes, we analyze OPT (1.3B, 2.7B, 6.7B, 13B) models by varying the percentage of selected data in Alpaca-Data and plot their win rates against the corresponding models trained on the complete dataset in Figure 6.4. Strikingly, we observe a downward trend in the minimum percentage of difficult data required (denoted by ✩) for achieving a win rate of at least 50 with an increase in the model's size. For example, the OPT-13B model outperforms its full dataset counterpart with only 3% of the training data. This suggests that as the model's size increases, the amount of challenging data required decreases, albeit the necessity for such difficult data persists. This finding provides additional insight into the exceptional performance exhibited by the Llama-65B model when trained with 1000 difficult samples in (Zhou et al., 2023).



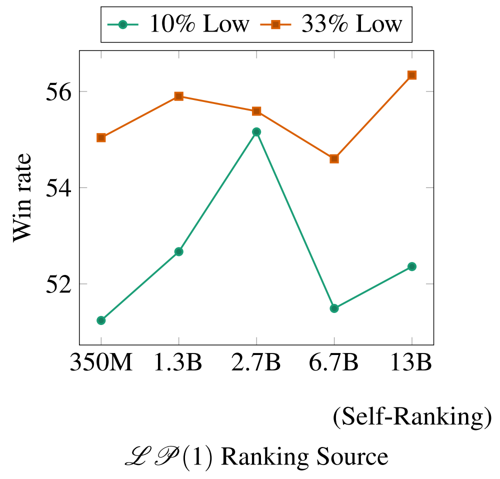
(a) Alpaca-Data, Llama models



(b) Dolly dataset, Llama models



(c) Alpaca-Data, OPT models



(d) Dolly dataset, OPT models

Figure 6.5. We vary the percentage of selected data to train 13B model and conduct a comparison of win rates obtained when data is self-selected by the 13B model vs selected by smaller models. The smaller model used is mentioned on the X-axis and the win rate is on the Y-axis. We observe that the data hardness transfers from smaller models to 13B, leading to improved or comparable performance compared to 13B model trained on the self-selected data.

6.4.2 Is Data Hardness transferable?

In the previous section, we observed challenging training data yields high-performing instruction-tuned models. In this section, we investigate transferability of data hardness, specifically whether samples deemed difficult by a smaller model are also considered difficult by a larger model.

To assess this, we consider a smaller and a larger model. We obtain $\mathscr{LP}(1)$ scores using the smaller model, following which we select the top-k% (in ascending order) of training data based on these scores. Subsequently, we train the larger model using this selected dataset. For each experimental configuration, we calculate the win rate of the larger model trained on the selected data against it when trained on the complete dataset.

We consider Llama-2 7B as the smaller model and Llama-2 13B as the larger model. We vary the percentage data selected, and plot the win rate of Llama-2 13B trained on selected data in comparison to the one fine-tuned on the entire Alpaca-Data dataset in Figure 6.5(a) and for Dolly dataset in Figure 6.5(b) respectively. We find that the ranking of the Llama-2 7B model transfers effectively to the 13B model, resulting in a model of comparable or even improved quality in some instances. For example, in Figure 6.5(a), the win rate of the Llama-2-13B model trained on 10% of the Alpaca-Data, selected by the 7B model, outperforms the self-ranking of the 13B model by 4 points.

Similarly, we consider OPT (350M, 1.3B, 2.7B, 6.7B) models as smaller models and OPT 13B as the larger model and plot the performance for the Alpaca-Data dataset in Figure 6.5(c) and for the Dolly dataset in Figure 6.5(d) respectively. From Figure 6.5(c), 6.5(d), we observe that the performance of the 13B model increases with the size of the smaller model up to 2.7B and further plateaus where it eventually matches the self-selection performance of 13B model. With only a 1.4-point drop in average win rate, even a small 350M model can be leveraged for curating training data for a large 13B model. This demonstrates that the data hardness transfers efficiently from a smaller model to a larger one, improving with the size of the smaller model and eventually matching self-selection performance beyond a specific size threshold (2.7B).

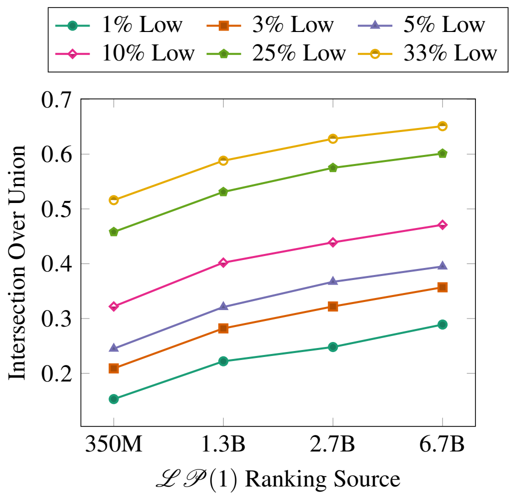


Figure 6.6. IOU scores of Alpaca-Data data points selected by smaller OPT models (350M, 1.3B, 2.7B, 6.7B) with OPT 13B model for varying percentages.

$\mathscr{LP}(1)$ **Ranking Analysis - Kendall-Tau scores:**

We conduct a comparative analysis of rankings between smaller and larger models to gain deeper insights into the transferability of data hardness. We derive $\mathscr{LP}(1)$ scores from multiple smaller models and a larger model, followed by the computation of Kendall-tau correlation coefficients between rankings based on their respective scores. The Kendall-tau score ranges from -1 to +1, with a higher positive score indicating a stronger correlation. The Kendall-tau scores of $\mathscr{LP}(1)$ derived from OPT-models (350M, 1.3B, 2.7B, 6.7B) against OPT 13B on both Alpaca-Data and Dolly datasets are presented in Table 6.1. We note that all scores are positive, indicating a positive correlation. Notably, we observe a consistent increase in correlation with the increase in size of the ranking source model. Furthermore, we also compute the Kendall-tau scores between rankings from the Llama-2 7B and 13B models. For Alpaca-Data, the score is 0.782, and for Dolly, it is 0.775, respectively. These scores underscore the effective transferability of rankings from smaller models to larger ones.

$\mathscr{LP}(1)$ **Ranking Analysis - Intersection over union scores:**

We additionally calculate the intersection-over-union (IOU) of data samples selected by both smaller and larger models. We vary the selected percentage of data and compute the IOU

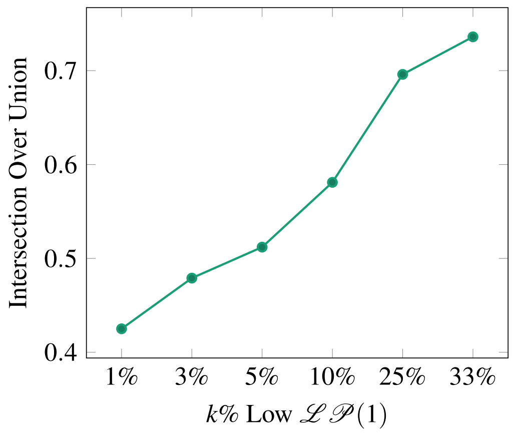


Figure 6.7. Intersection-over-union scores of Alpaca-Data data points selected by Llama-2 7B with Llama-2 13B model for varying percentages.

Table 6.1. Kendall-Tau scores of rankings from different smaller-sized OPT models(350M, 1.3B, 2.7B, 6.7B) against the ranking from large OPT-13B model.

Dataset	**350M**	**1.3B**	**2.7B**	**6.7B**
Alpaca-Data	0.52	0.61	0.65	0.69
Dolly	0.52	0.61	0.67	0.75

of subsets chosen by the smaller OPT models (350M, 1.3B, 2.7B, 6.7B) with the larger OPT 13B model on Alpaca-Data, presenting the results in Figure 6.6. Similarly, we plot the IOU of Llama-2 7B with the Llama-2 13B model on the Alpaca-Data in Figure 6.7. From Figure 6.6, for a given percentage of data selected, we observe a consistent rise in IOU score with the increasing size of the model until 2.7B, followed by a plateau, aligning with the performance trend depicted in Figure 6.5(c). Moreover, for a fixed model size in Figures 6.6 and 6.7, the IOU score consistently increases with the rise in the selected data percentage. This finding suggests that for selecting a larger percentage of data, a smaller 350M model suffices. However, as the selected data percentage decreases, it is advisable to employ a larger model i.e. $\geq$ 2.7B.

Table 6.2. We compare $\mathscr{LP}_{app}$ and $\mathscr{LP}$ on Alpaca-Data. We present the win rates of the model trained on different percentages of selected data using both $\mathscr{LP}_{app}$ and $\mathscr{LP}$ against the one trained on the complete dataset.

Model	**3% Low**		**10% Low**		**33% Low**	
	$\mathscr{LP}_{app}$	$\mathscr{LP}$	$\mathscr{LP}_{app}$	$\mathscr{LP}$	$\mathscr{LP}_{app}$	$\mathscr{LP}$
OPT 2.7B	**49.4**	47.1	**53.7**	50.4	52.9	**54.9**
OPT 6.7B	**50.0**	48.9	**52.6**	52.1	**52.7**	51.1
Llama-2 7B	**59.7**	57.6	**57.9**	56.7	55.7	**56.5**
Llama-2 13B	**56.5**	56.4	54.0	**54.3**	**55.3**	54.8

6.5 $\mathscr{LP}_{app}$ A Faster & Approximate Learning Percentage Metric

It is worth noting that to compute $\mathscr{LP}(1)$, the model needs to be trained twice—first to obtain the perplexity scores and rank the data, and then to select the data and train the model again. Recognizing this computational inefficiency, we present an approximate version of the learning percentage, denoted as $\mathscr{LP}_{app}$ that is faster and equally effective as $\mathscr{LP}$.

Language models are known to typically learn in 3 epochs and tend to memorize the data (Tirumala et al., 2022). As a result, we assume that the perplexity at the end of training $\mathscr{P}_n$ is constant for all samples. Mathematically, $\mathscr{LP}_{app}$ is defined as:

$$\mathscr{LP}_{app}(i) = \frac{\mathscr{P}_{i-1} - \mathscr{P}_i}{\mathscr{P}_0} \tag{6.2}$$

To compute $\mathscr{LP}_{app}(1)$, we only need to train the model once for 1 epoch, making it more efficient.

6.5.1 $\mathscr{LP}_{app}(1)$ vs $\mathscr{LP}(1)$: A Comparison

We conduct a comparative analysis between $\mathscr{LP}_{app}$ and $\mathscr{LP}$ metrics on the Alpaca-Data. We consider different-sized models including OPT 2.7B, 6.7B, as well as Llama-2 7B and 13B. The training data is ranked using $\mathscr{LP}_{app}$ and $\mathscr{LP}$ metrics, respectively, and data selection is performed for varying percentages of data. The win rate of the model trained on selected data

Table 6.3. Kendall-Tau scores between $\mathscr{LP}$ and $\mathscr{LP}_{app}$. We observe high positive scores indicating a positive correlation.

Dataset	**OPT**				**Llama-2**	
	1.3B	2.7B	6.7B	13B	7B	13B
Alpaca-Data	0.53	0.55	0.60	0.61	0.64	0.62
Dolly	0.59	0.59	0.61	0.64	0.60	0.58

against the model trained on the complete dataset is computed and presented in Table 6.2. Notably, we observe that the model trained on data selected via $\mathscr{LP}_{app}$ outperforms its counterpart trained on data selected via $\mathscr{LP}$ across the majority of models and various percentages. This finding underscores the efficacy of $\mathscr{LP}_{app}$ as a data selection metric, demonstrating its comparable or even superior performance compared to $\mathscr{LP}$.

We calculate kendall-tau correlation scores between $\mathscr{LP}_{app}$ and $\mathscr{LP}$ on both Alpaca-Data and Dolly datasets, shown in Table 6.3. We observe high positive scores, signifying a positive correlation between the two metrics, highlighting the effectiveness of $\mathscr{LP}_{app}$ in accurately approximating $\mathscr{LP}$.

6.5.2 Comparison with Baselines

In this section, we compare $\mathscr{LP}_{app}$ with two baselines. The first baseline, denoted as **Clust Rand**, randomly samples the same number of samples as our method from each cluster of the training set. Notably, this preserves the diversity of the subset while removing the difficulty-aware ranking. We also compare with **Alpagasus** (Alpa) (Chen et al., 2024), which prompts GPT-3.5 to assign a difficulty rating and selects training instances deemed difficult. We select 10% of the training data using each method and consider OPT 1.3B, 2.7B, 6.7B, and Llama-2 7B models. These models are then trained on the selected data using each method. Subsequently, we compare the performance of the instruction-tuned models using AlpacaEval and present the win rates of our model over the compared baselines. The win rates post-training on the Alpaca-Data and Dolly are presented in Table 6.4. Notably, we observe win rates exceeding 50 for all models trained on both datasets, indicating the superior quality of training data subsampled using our

Table 6.4. The win rates of models trained on data subsampled from Alpaca-Data and Dolly datasets based on $\mathscr{LP}_{app}$ are compared against other baselines (Clust Rand & Alpagasus). We observe that all win rates exceed 50, indicating superior performance and high-quality selection by our method.

Model	**Alpaca-Data**		**Dolly**	
	Clust Rand	Alpa	Clust Rand	Alpa
OPT 1.3B	53.91	51.74	53.79	54.10
OPT 2.7B	56.89	52.11	56.21	52.61
OPT 6.7B	59.13	54.47	57.27	55.09
Llama-2 7B	55.60	53.17	58.76	54.66

method. The superior performance of smaller OPT 1.3B and 2.7B models trained on self-selected data over Alpagasus, where the data is selected by a much larger GPT-3.5 model, underscores the effectiveness of our method.

Additionally, we conduct another evaluation wherein a smaller model is employed to curate training data for a larger model utilizing the $\mathscr{LP}_{app}(1)$ metric. Subsequently, we train the larger model on the selected data and compare its performance with that of the same model trained on data selected using Alpagasus. The win rates of OPT 6.7B model trained on 10% data selected by OPT 350M, 1.3B and 2.7B models are 50.25, 51.24, and 52.30 respectively. The win rates exceed 50% across all scenarios, indicating that a smaller model can effectively curate training data using our proposed $\mathscr{LP}_{app}$ metric.

6.5.3 Human Evaluation

We compare the model trained on data selected using our method with the model trained on complete dataset. Specifically, we consider Llama-2 7B model and Alpaca-Data, and subsample 5% of data using $\mathscr{LP}_{app}(1)$ scores and train it. Additionally, we train another Llama-2 7B model on full Alpaca-Data. In this human evaluation, participants are asked to provide an instruction, after which both models generate a response. Participants are then prompted to choose the better response, or if both responses were perceived as equal. Importantly, the models were hidden from the participants, ensuring they were unaware of which model corresponded to which response. We recruited 10 students with minimal prior knowledge of the project for this evaluation. In total, we collected 151 evaluations. Of these, 42 evaluations resulted in a tie. In 50 evaluations, the model trained on the full dataset was preferred, while in 58 evaluations, participants found the model trained on 5% data selected using our method to be better. This indicates that responses from the model trained on 5% of the data were either better or of equal quality compared to those from the model trained on the complete dataset in 66.2% of instances. This outcome provides another validation for the superior performance of our method.

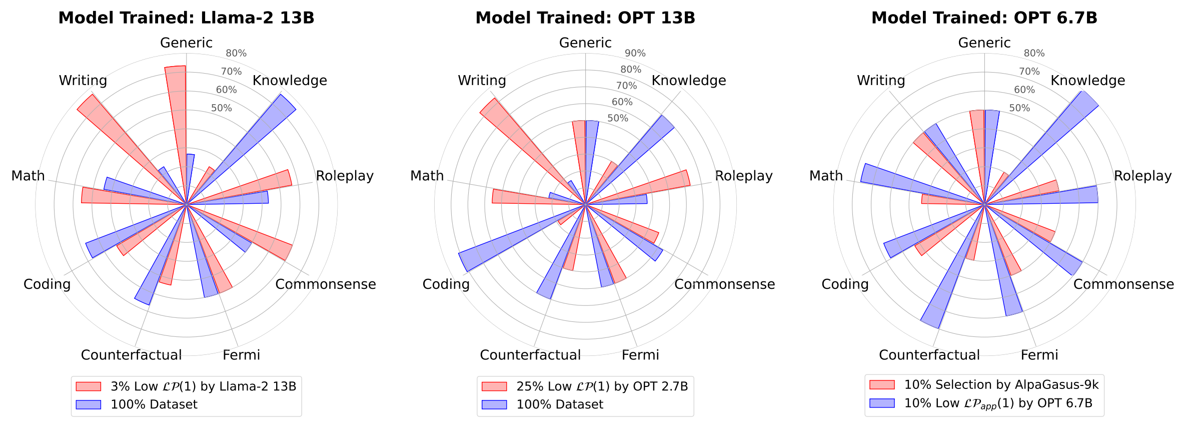
6.6 Dissecting the Difficult Data

In this section, we analyze the characteristics of samples identified as challenging by the $\mathscr{LP}$ metric. We manually examine 250 samples selected from the 1% subset characterized by low $\mathscr{LP}(1)$ scores within the Alpaca-Data corpus, obtained using Llama-2 7B.

We observe that these difficult samples are longer than the average, maintaining coherence throughout. Specifically, the average response length within the 1% Low $\mathscr{LP}(1)$ subset of Alpaca-Data is 547 characters, contrasting with the dataset's average of 270. This observation aligns with intuition, suggesting that models encounter difficulty in generating longer and coherent text, thus deeming such instances as challenging.

We also found six noisy samples, shown in Table D.1, i.e. a noise rate of 2.4%. Notably, this proportion is significantly higher compared to the prevalence observed across the entire dataset. AlpacaDataCleaned[6], a human-cleaned Alpaca-Data has eliminated 0.47% of noisy samples from the original dataset. This underscores that the subset of most challenging samples identified by $\mathscr{LP}(1)$ encompasses noisy instances as well. Addressing this issue requires future investigation.

[6]https://github.com/gururise/AlpacaDataCleaned



(a) Self-selection via $\mathscr{LP}(1)$ Analysis (b) Small-to-large generalization via $\mathscr{LP}(1)$ (c) $\mathscr{LP}_{app}(1)$ vs Alpagasus

Figure 6.8. We compare models on different skills from the Vicuna split of the Alpaca-eval test set.

6.7 Skill-Chart Analysis

In this study, we consider the Vicuna-split of the Alpaca-eval dataset, which is categorized into nine distinct skill categories, and compare model performances within each skill. The Alpaca-Data is used as the training dataset for this evaluation.

Firstly, we compare the performance of the Llama-2 13B model trained on 3% of the data self-selected using our proposed $\mathscr{LP}(1)$ metric against the same model trained on the complete dataset, as illustrated in Figure 6.8(a). Our findings indicate that performance either improves or remains constant in the math, writing, generic, roleplay, commonsense, and fermi skills when using the 3% self-selected dataset. However, the model trained on the full dataset outperforms in coding, knowledge, and counterfactual skills. This suggests that the data selected using the $\mathscr{LP}(1)$ metric is effective and high quality, although certain skills may benefit from a larger data volume.

Secondly, we examine the performance of a larger model trained on data selected by a smaller model compared to the larger model trained on the full dataset, as shown in Figure 6.8(b). In this scenario, we use OPT-2.7B as the smaller model to select 25% data and OPT-13B as the larger model, plotting the win percentage per skill. We notice similar performance trends as previously where the performance either improves or is stable in math, writing, generic, roleplay, and fermi skills with the smaller model's data selection. Conversely, the larger model trained on the complete dataset performs better in coding, knowledge, counterfactual, and commonsense skills. This demonstrates that the smaller model can effectively select data for most skills similar to the larger model.

Finally, we compare the data selected using the Alpagasus with data selected using our proposed approximated $\mathscr{L}\mathscr{P}_{app}(1)$ metric in Figure 6.8(c). We use the OPT-6.7B model and select 10% of the data using both methods. The results show a uniform improvement across all skills with our proposed method, highlighting its superior performance.

6.8 Summary

In this chapter, we introduce a learning percentage-based metric for assessing the difficulty of samples. We demonstrate that LMs ranging from 1B to 13B sizes can self-select high-quality training data by employing this metric. Additionally, we empirically validate the transferability of data hardness across different model sizes, showcasing the efficient curation of high-quality training data by smaller models. Furthermore, we propose an optimized version of the metric that offers increased speed while maintaining equal efficacy.

Chapter 6, in full, is a reprint of the material as it appears in Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. 2024. Smaller Language Models are capable of selecting Instruction-Tuning Training Data for Larger Language Models, in Findings of the Association for Computational Linguistics: ACL 2024, pages 10456–10470, Association for Computational Linguistics. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 7

Conclusion and Future works

7.1 Summary of Contributions

This thesis has investigated how weak supervision techniques can alleviate the annotation bottleneck. We explored two types of weak supervision, specifically extractive & generative weak supervision. We empirically demonstrated that the training dynamics of LLMs provide valuable insights into the nature and quality of the data they are trained on. To evaluate data quality, we proposed analyzing it along three key dimensions: diversity, difficulty, and correctness. High-quality data is achieved by maximizing diversity and difficulty while maintaining the correctness. We developed cost-effective data curation methods that effectively utilize these approaches to enhance the performance.

In the realm of extractive weak supervision, we first addressed a major limitation of seed-word-based supervision: contextual ambiguity. To resolve this, we proposed a contextualized weak supervision approach that leverages pre-trained LLMs to accurately interpret the intended meaning of a seed word for a given class.

While extractive weak supervision provides valuable pseudo-labels, its coverage is inherently limited. To expand the scope of weak supervision, we explored metadata as an auxiliary supervision signal. Despite being widely available, it is often underutilized due to its diverse formats (e.g., strings, integers) and the complex dependencies between metadata attributes. To address these challenges, we modeled text and metadata as a text-rich network, employing motif-patterns and motif-instances to extract meaningful signals, ultimately generating additional pseudo-labeled data.

Finally, to improve the correctness of pseudo-labeled data, we proposed a learning order-based data selection method. We observe that deep neural networks tend to learn clean samples earlier in training, while noisier samples are memorized later. Leveraging this insight, we introduced a filtering strategy that discards samples learned in the final training epochs, thereby improving the data quality.

Next, we explored generative weak supervision. Firstly, we explored synthetically generating text classification datasets. We reframed text classification as a context generation task for a given question-answer pair. To facilitate this, we fine-tuned an LLM using publicly-available QA datasets, transforming it into a context generator capable of producing synthetic data for text classification tasks.

We then proposed a tool selection dataset generation framework for LLMs, specifically for single-tool usage tasks. We synthetically generated tools, instructions, reasoning steps, and final tool selection data. We observed that the difficulty of a training sample increases when similar yet distinct tools are included in the candidate set. Inspired by this insight, we adopted a hierarchical generation strategy, starting with a diverse seed set from different domains to ensure diversity in resulting tools. As we go deeper, along a domain, we generate similar tools belonging to that domain. During the sample curation, we selectively subsample tools from the generated tools, ensuring that the candidate tool list includes sibling tools of the ground truth tool. To further enhance performance, we introduced a self-verification mechanism, where LLMs prompt themselves with targeted questions to clarify ambiguities and reduce confusion in tool selection.

Finally, we observed that synthetically generated data often lacks sufficient difficulty. To address this, we proposed a data selection strategy that prioritizes harder training samples. We introduced a learning percentage-based metric, selecting samples learned in the later training epochs to identify more challenging data. Additionally, we empirically demonstrated that data hardness transfers from smaller to larger language models, showing that smaller models can effectively curate high-quality, difficult training data for larger models.

7.2 Broader Implications

This dissertation has several important implications for the future of training data curation for language models:

Guidelines for dataset generation. Various approaches leveraging different sources of weak supervision offer a general guidelines for dataset generation. We recommend first defining what constitutes high-quality data for the specific task. Next, it is important to consider the factors that contribute to sample difficulty and diversity. Finally, ensuring high correctness in the generated data is essential. These guidelines provide a robust foundation for effective dataset curation across different tasks.

Quality vs Quantity. The success of our approaches indicates that improving model performance does not necessarily require ever-larger datasets. Our findings demonstrate that carefully curated high-quality training data can achieve strong performance with fewer samples. In other words, enhancing data quality reduces the need for large quantities of data while maintaining high performance.

Training dynamics offer important insights. The improvements achieved through training dynamics-based techniques, such as learning-order and learning-percentage-based selection, highlight their effectiveness in assessing training data quality. These insights can be leveraged to better understand the data used for training. In other words, if a model already possesses knowledge of certain data, it can be identified and removed, optimizing the training process.

7.3 Future Directions

Our work opens several promising directions for future research:

Agents Data Curation. Agents are poised to shape the future of artificial intelligence, and training them requires high-quality data. But what defines high-quality data for an agent? Defining a high-quality trace for an agent is still an open problem. Ideally, it would consist of long, meaningful traces. However, manually curating such traces is highly challenging. To address this, we need efficient and scalable data generation strategies tailored for agent training.

Unified Data Curation. An interesting avenue for future research is developing data curation methods that integrate multiple data types by combining extractive and generative weak supervision. Leveraging data from diverse sources would enable us to harness the strengths of both approaches while minimizing their respective limitations.

Prompts for Generative Weak Supervision. Generative weak supervision relies on user-provided prompts to guide language models in generating data. However, ensuring that these prompts are diverse so that they produce sufficiently diverse data is a time-consuming task for humans. Developing efficient methods for curating prompts (Jafari et al., 2024) is a promising future direction that could enhance the quality and variability of generated data while reducing manual effort.

Dynamic Data Generation Strategies. Future research could investigate dynamic data generation strategies, where the process dynamically adjusts based on the target model's learning progress. This approach could enhance training efficiency and effectiveness by continuously optimizing the quality and relevance of generated data.

Application to Other Domains. Although our work has primarily focused on language models, the principles we have developed could also be applied to other domains experiencing similar data scarcity challenges, such as computer vision and speech recognition.

7.4 Concluding Remarks

The challenge of curating high-quality training data will likely remain a central concern for language models and agents in the coming years. This thesis has shown that by focusing on high-quality data, we can reduce the overall quantity needed, thus making more efficient use of resources, thereby advancing the capabilities and accessibility of large language models.

Looking ahead, the approaches presented in this work suggest that the future may not require the indiscriminate accumulation of data, but rather the focused collection of high-quality data. By further refining our understanding of how models learn from data, we can develop more efficient and effective training methods that maximize the potential of available resources.

Appendix A

Learning Order Inspired Pseudo-Label Selection for Weakly Supervised Text Classification

A.1 Datasets

The details of datasets are provided below:

- **The New York Times (NYT):** The NYT dataset is a collection of news articles published by The New York Times. They are classified into 5 coarse-grained genres (e.g., science, sports) and 25 fine-grained categories (e.g., music, football, dance, basketball).
- **The 20 Newsgroups (20News):** The 20News dataset[1] is a collection of newsgroup documents partitioned widely into 6 groups (e.g., recreation, computers) and 20 fine-grained classes (e.g., graphics, windows, baseball, hockey). Following (Wang et al., 2021), coarse- and fine-grained miscellaneous labels are ignored.
- **AGNews** (Zhang et al., 2015) is a huge collection of news articles categorized into four coarse-grained topics such as business, politics, sports, and technology.
- **Books** (Wan & McAuley, 2018; Wan et al., 2019) is a dataset containing description of books, user-book interactions, and users' book reviews collected from a popular online book review website Goodreads[2]. Following (Mekala et al., 2020), we select books belonging to eight

[1]http://qwone.com/~jason/20Newsgroups/
[2]https://www.goodreads.com/

popular genres. Using the title and description as text, we aim to predict the genre of a book.

A.2 Compared Weakly Supervised Text Classification Methods

We compared with following state-of-the-art weakly supervised text classification methods described below[3]:

- **ConWea** (Mekala & Shang, 2020) is a seed-word driven iterative framework that uses pre-trained language models to contextualize the weak supervision.
- **X-Class** (Wang et al., 2021) takes only label surface names as supervision and learns class-oriented document representations. These document representations are aligned to classes, computing pseudo labels for training a classifier.
- **WeSTClass** (Meng et al., 2018) generates pseudo documents using seed information and refines the model through a self-training module that bootstraps on unlabeled documents.
- **LOTClass** (Meng et al., 2020) queries replacements of class names using BERT (Devlin et al., 2019) and constructs a category vocabulary for each class. This is used to pseudo-label the documents via string matching. A classifier is trained on this pseudo-labeled data with further self-training.

We use the public implementations of these methods and modify them to plug-in our filter. Specifically, in WeSTClass and LOTClass, we add our filter after generating the pseudo documents; in ConWea, we add our filter before training the text classifier; and for X-Class, we plug-in our filter after learning the document-class alignment.

A.3 Experimental Settings

Train-Test sets. We remove the labels in the whole dataset and our task is to assign labels to these unlabeled samples. We measure our performance on the whole dataset by comparing it

[3]We also considered experimenting on ASTRA, however the instructions to run on custom datasets were not made public yet.

Table A.1. Evaluation results on six datasets using RoBERTa classifier and pseudo-label selection methods. Initial pseudo-labels are generated using String-Match. Micro- and Macro-F1 scores and their respective standard deviations are presented in percentages. For a fair comparison, we consider the same number of samples for all baselines as LOPS in each iteration. Abnormally high standard deviations are highlighted in *blue* and low performances are highlighted in *red*. Baselines performing better than our method are made bold.

		Coarse-grained Datasets								Fine-grained Datasets			
		NYT-Coarse		20News-Coarse		AGNews		Books		NYT-Fine		20News-Fine	
Classifier	Method	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
RoBERTa	Standard	90.2(0.41)	82.1(0.24)	76.5(0.41)	75.7(0.58)	74.4(0.44)	74.2(0.71)	57.6(0.29)	58.6(0.53)	79.4(0.65)	76.6(0.54)	67.4(0.67)	67.3(0.87)
	LOPS	**92.4(2.99)**	**85.6(3.00)**	**77.5(2.00)**	**75.8(2.00)**	**75.6(0.22)**	**75.5(0.27)**	**59.7(0.41)**	**60.5(0.45)**	**81.8(0.90)**	**80.7(0.50)**	**70.7(0.68)**	**70.8(0.34)**
	O2U-Net	93.1(0.14)	86.3(0.26)	76.5(0.19)	73.4(1.47)	**77.6(0.36)**	**77.1(0.54)**	58.5(0.64)	59.9(0.32)	79.2(0.28)	77.5(1.17)	68.4(0.47)	68.3(0.15)
	Random	92.3(0.21)	84.4(0.82)	76.5(1.00)	74.5(1.00)	74.6(0.32)	74.2(0.27)	56.4(0.57)	58.7(0.32)	76.6(1.25)	74.8(0.34)	68.4(0.23)	68.5(0.23)
	Probability	**93.4(0.48)**	**87.5(1.00)**	76.7(0.50)	75.4(1.00)	76.2(0.89)	76.3(1.12)	56.2(1.28)	57.4(1.85)	26.6(23.00)	14.4(11.50)	46.2(23.00)	45.3(23.50)
	Stability	90.5(1.09)	83.3(0.50)	**78.5(1.00)**	**76.0(1.50)**	76.5(0.48)	76.5(0.64)	58.5(1.18)	59.5(1.06)	21.5(12.50)	9.2(5.00)	70.3(1.00)	70.6(1.00)
	OptimalFilter	98.2(0.17)	96.1(0.16)	94.3(0.74)	94.5(0.35)	89.7(0.17)	89.3(0.28)	76.5(0.29)	77.7(0.22)	97.4(0.34)	92.8(0.26)	85.3(0.32)	85.5(0.65)

with their respective gold labels.

Computation Infrastructure. We performed our experiments on NVIDIA RTX A6000 GPU. The batch size for training BERT is 32, RoBERTa is 32, GPT2 is 4, XLNet is 1. The running time for BERT and RoBERTa took 3 hrs, GPT2 took 6 hours, and XLNet took 12 hrs.

A.4 Additional Experiments

We also compare with **RoBERTa** (`roberta-base`) (Liu et al., 2019) as text classifier. We fine-tune it for 3 epochs. The results are shown in Table A.1.

A.5 Statistical Significance Tests

We perform a paired t-test between LOPS and each of the other baseline filtering techniques for all classifiers and on all datasets. The results are showed in Table A.2. From these p-values, we can conclude that the performance improvement over baselines is significant.

A.6 Example samples

A few incorrectly pseudo-labeled samples from NYT-Fine dataset that are selected by probability-based selection by BERT are shown in Table A.3 We observe a high probability

Table A.2. Statistical significance results.

Classifier	Method	NYT-Coarse	NYT-Fine	20News-Coarse	20News-Fine	AGNews	Books
BERT	Standard	1.93×10^{-112}	1.92×10^{-105}	7.08×10^{-80}	9.37×10^{-79}	1.05×10^{-74}	7.15×10^{-96}
	Random	1.58×10^{-115}	2.01×10^{-105}	5.98×10^{-94}	7.32×10^{-39}	4.26×10^{-81}	3.25×10^{-100}
	Probability	1.69×10^{-112}	6.25×10^{-189}	4.19×10^{-120}	6.71×10^{-136}	5.13×10^{-71}	8.72×10^{-123}
	Stability	2.63×10^{-33}	2.41×10^{-194}	2.78×10^{-58}	4.07×10^{-9}	1.36×10^{-45}	1.24×10^{-97}
RoBERTa	Standard	6.06×10^{-100}	1.82×10^{-63}	5.4×10^{-3}	3.09×10^{-109}	2.13×10^{-57}	1.15×10^{-22}
	Random	8.38×10^{-94}	3.55×10^{-71}	3.26×10^{-39}	5.20×10^{-101}	5.12×10^{-72}	1.75×10^{-61}
	Probability	5.27×10^{-62}	9.18×10^{-193}	1.39×10^{-71}	1.13×10^{-85}	4.03×10^{-24}	2.16×10^{-72}
	Stability	1.46×10^{-86}	3.39×10^{-188}	6.28×10^{-5}	8.71×10^{-107}	1.17×10^{-76}	1.81×10^{-65}
XLNet	Standard	3.14×10^{-79}	4.68×10^{-139}	5.42×10^{-112}	4.17×10^{-103}	1.69×10^{-114}	5.63×10^{-107}
	Random	3.26×10^{-71}	2.97×10^{-48}	2.56×10^{-77}	5.32×10^{-75}	6.38×10^{-32}	4.38×10^{-48}
	Probability	4.12×10^{-29}	1.36×10^{-63}	7.25×10^{-19}	6.27×10^{-47}	1.57×10^{-31}	6.23×10^{-32}
	Stability	6.17×10^{-29}	4.27×10^{-44}	1.47×10^{-73}	3.57×10^{-41}	1.79×10^{-28}	3.48×10^{-56}
GPT-2	Standard	6.09×10^{-50}	1.10×10^{-98}	2.05×10^{-57}	1.22×10^{-5}	4.68×10^{-91}	1.56×10^{-65}
	Random	2.54×10^{-22}	6.97×10^{-81}	4.25×10^{-91}	9.89×10^{-38}	6.39×10^{-77}	8.70×10^{-63}
	Probability	5.52×10^{-49}	2.37×10^{-89}	7.02×10^{-85}	1.05×10^{-83}	1.99×10^{-63}	3.44×10^{-49}
	Stability	6.15×10^{-110}	3.88×10^{-31}	3.40×10^{-66}	6.27×10^{-78}	2.21×10^{-47}	2.36×10^{-41}

assigned to each incorrect pseudo-label whereas these are learnt by the classifier at later epochs. These wrongly annotated samples induce error that gets propagated and amplified over the iterations. By not selecting these wrong instances, LOPS curbs this and boosts the performance.

A.7 Learning Order vs Probability Score: Threshold Analysis

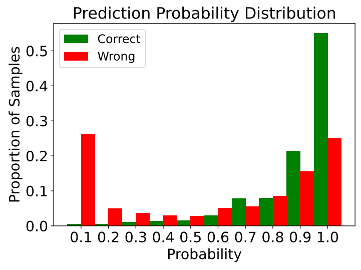
Ideally, there exists a threshold for a given confidence function that perfectly distinguishes the correctly and wrongly labeled samples. However, in practice, confidence functions may not be possible to suffice such ideal condition. For a given confidence function, one wishes to select pseudo-labels based on a threshold such that the noise is low and the coverage is high. We define ratio between noise and coverage as *NC-ratio*, namely $r(\kappa, \gamma) = \frac{\varepsilon(\kappa,\gamma)}{\phi(\kappa,\gamma)}$. An optimal threshold has the lowest NC-ratio. Therefore, we evaluate confidence function by plotting NC-ratio at different thresholds.

We plot NC-ratios of learning order and probability scores with BERT classifier in Figure A.2 on NYT-Coarse, 20News-Fine datasets. To isolate them from the effects of bootstrapping, we don't perform any bootstrapping. As shown in Figure A.2, when selecting the optimal threshold, learning order has significantly lower NC-ratios for all datasets compared

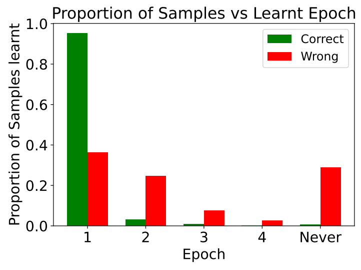
Table A.3. Incorrectly pseudo-labeled samples selected by probability-based selection are shown below. These samples are learnt at later epochs, thus LOPS avoids selecting them.

Document	**Pseudo-label**
Corinthians have received offer from tottenham hotspur for brazil's paulinho although the midfielder said on saturday he would not decide his future until after the confederations cup ."there is an official offer from tottenham to corinthians but, as i did when there was an inter milan offer, i'll sit and decide with my family before i make any decision," paulinho told reporters.	Football Softmax Prob: 0.96 Learnt Epoch: 2
Brittney griner and elena delle donne were poised to make history as the first pair of rookies from same class to start wnba all-star game. Now, neither will be playing as both are sidelined with injuries. It's a tough blow for the league, which has been marketing the two budding stars.	Baseball Softmax Prob: 0.96 Learnt Epoch: 2
Denmark central defender simon kjaer has joined french side lille from vfl wolfsburg on a four-year deal. Lille paid two million euros. 72 million pounds for the 24-year-old kjaer, who has won 35 caps for his country. He joined wolfsburg from palermo for 12 million euros.	Intl. Business Softmax Prob: 0.94 Learnt Epoch: 2
Fiorentina striker giuseppe rossi is quickly making up for lost time after suffering successive knee ligament injuries which kept him out of action for the best part of two years.	Football Softmax Prob: 0.95 Learnt Epoch: 2

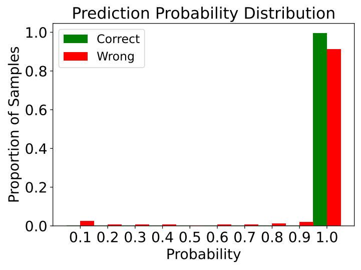
to probability score. Furthermore, the optimal thresholds of learning order for all datasets are almost the same. In contrast, the optimal thresholds of probability score vary greatly across different datasets due to the poor calibration of DNNs. Finally, we also observe that the NC-ratio for probability score often changes greatly around the optimal threshold, which poses difficulty in locating the optimal threshold. In contrast, since there are only few possible thresholds for learning order, it is easier to find the optimal threshold. From the performance vs threshold plot in Figure A.3, we can observe that learning order performs better than Probability score across multiple thresholds. Therefore, in terms of both performance and robustness, learning order is a more effective confidence function than probability score.



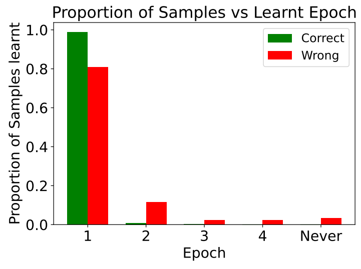
(a)



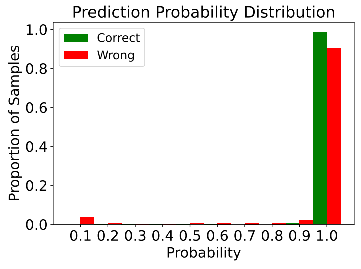
(b)



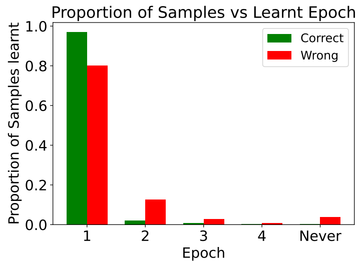
(c)



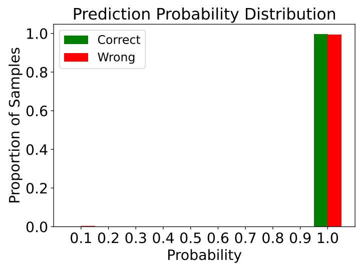
(d)



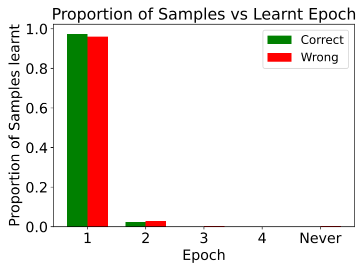
(e)



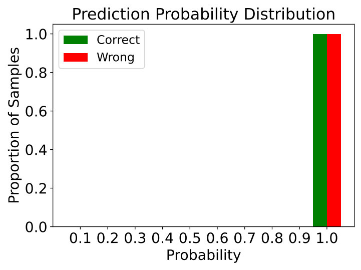
(f)



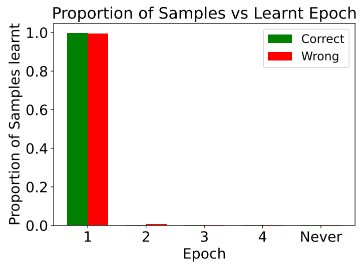
(g)



(h)



(i)



(j)

Figure A.1. Distributions of correctly and wrongly labeled pseudo-labels using different selection strategies on all datasets for its initial pseudo-labels. The base classifier is BERT. Each row represents a dataset. Figure (a), (b) represents NYT-Fine, (c), (d) represents 20News-Coarse, (e), (f) represents 20News-Fine, (g), (h) represents Books, and (i), (j) represents AGNews datasets respectively. Left column is based on the softmax probability of samples' pseudo-labels and right column is based on the earliest epochs at which samples are learnt.

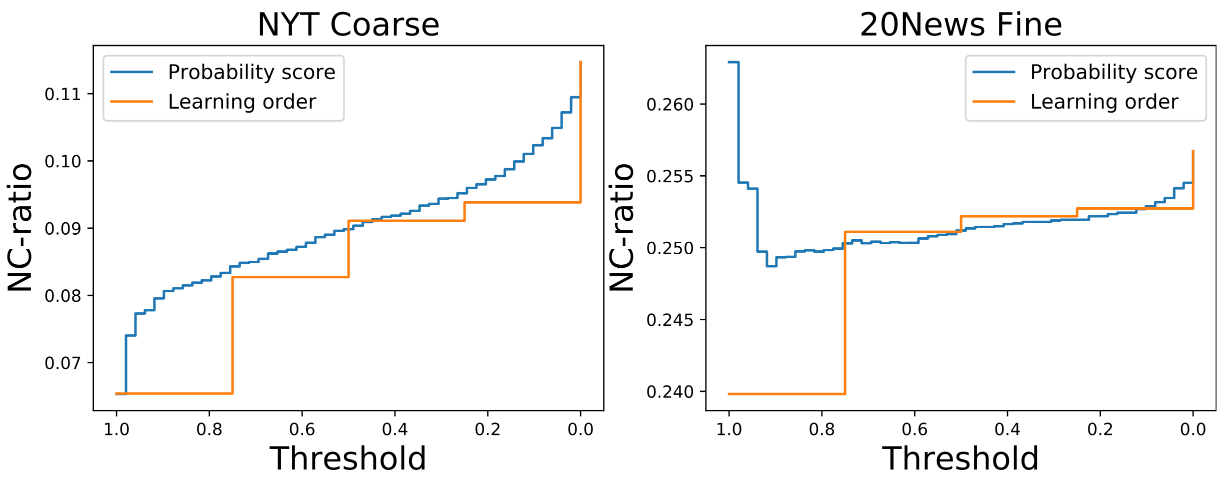
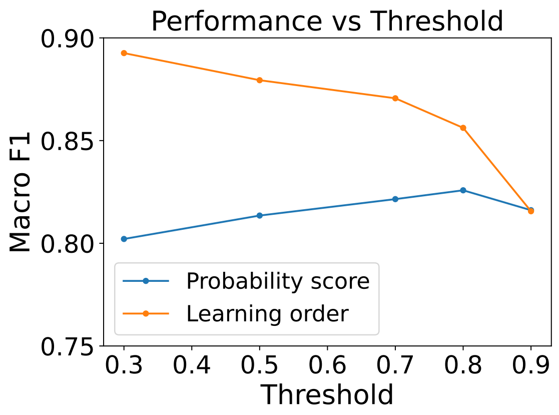
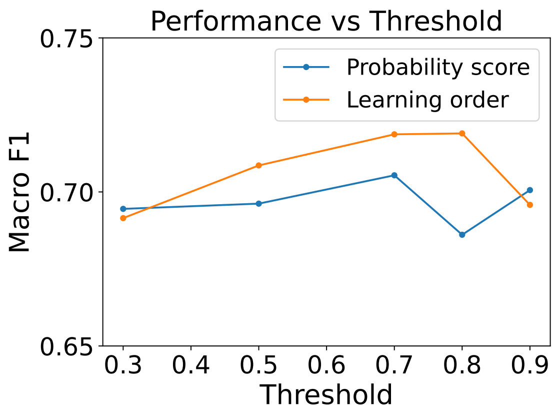


Figure A.2. NC-ratios of learning order and probability score with BERT as the classifier.



(a) NYT Coarse



(b) 20News Fine

Figure A.3. Macro-F_1 scores vs Threshold on NYT-Coarse & 20News-Fine datasets using BERT classifier with LOPS and Probability score based selection.

Appendix B

Leveraging QA Datasets to Improve Generative Data Augmentation

B.1 Target Task Datasets

The details of target task datasets are as follows:

- **IMDb:** (Maas et al., 2011) is a movie review dataset with positive and negative as sentiments.
- **Yelp:**[1] is a collection of reviews written by Yelp users with five fine-grained sentiment ratings.
- **SST-2:** (Socher et al., 2013) is a binary sentiment classification dataset with single sentence texts.
- **Yahoo:** (Zhang et al., 2015) is a topic classification dataset with question and answer pairs. Using these pairs, the task is to predict their corresponding topic.
- **The New York Times (NYT):** : contains news articles written and published by The New York Times that are classified into 5 wide genres.
- **AGNews:** (Zhang et al., 2015) is a topic categorization dataset in news domain from AG's corpus.

The size of test sets is mentioned in Table B.1.

[1]https://www.yelp.com/dataset/

Table B.1. Dataset statistics.

Dataset	# Test Examples
IMDb	25000
Yelp	50000
SST-2	2211
Yahoo	60000
NYT	10582
AGNews	114000

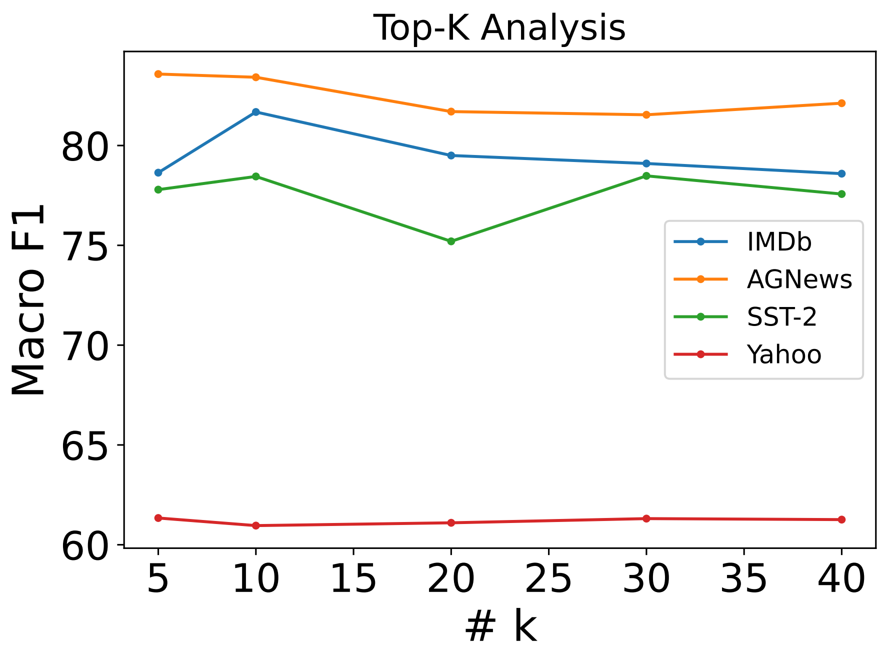


Figure B.1. Macro-F_1 scores of CONDA-SocialIQA w.r.t. k. Each experiment is repeated with three different seeds and mean performance is plotted.

B.2 Performance vs k

We vary k in top-k sampling and plot the performance of CONDA-SocialIQA on IMDb, SST-2, AGNews, and Yahoo datasets in Figure B.1. We fix the few-shot supervision size to 8 samples per label and generate 450 samples per label. We repeat each experiment thrice and plot the mean performance. Upon manual inspection, We observe that the samples generated with k=20 are more diverse than k=10, however, the influence of k on performance is not significant.

Table B.2. Evaluation Results with validation set.

	Sentiment						**Topic**					
	IMDb		**Yelp**		**SST-2**		**NYT**		**Yahoo**		**AGNews**	
Method	Mi-F_1	Ma-F_1	Mi-F_1	Ma-F_1	Mi-F_1	Ma-F_1	Mi-F_1	Ma-F_1	Mi-F_1	Ma-F_1	Mi-F_1	Ma-F_1
BERT-FT	$68.7_{3.6}$	$68.5_{3.6}$	$38.1_{5.2}$	$36.3_{5.2}$	$57.5_{1.4}$	$55.8_{2.3}$	$88.7_{5.5}$	$83.9_{4.6}$	$54.4_{1.8}$	$53.9_{1.2}$	$74.6_{5.6}$	$74.7_{5.3}$
ITFT-MNLI	$66.2_{3.3}$	$64.6_{4.2}$	$35.3_{3.7}$	$33.3_{3.5}$	$60.8_{1.8}$	58.71.6	$78.0_{2.9}$	$63.1_{6.2}$	$28.2_{4.5}$	$27.2_{4.2}$	$52.9_{3.2}$	$51.4_{4.5}$
ITFT-SQuAD	$61.1_{2.0}$	$59.7_{2.6}$	$34.0_{2.3}$	$31.6_{3.6}$	$56.5_{1.4}$	$56.0_{1.5}$	$88.9_{2.2}$	$75.8_{4.6}$	$36.2_{4.0}$	$35.3_{4.4}$	$58.2_{5.5}$	$56.2_{6.6}$
BackTranslation	$67.4_{2.0}$	$66.9_{2.0}$	$\underline{38.7_{3.6}}$	$\underline{36.4_{4.4}}$	$61.0_{4.3}$	$60.4_{5.0}$	$93.7_{1.4}$	$88.7_{0.3}$	$\underline{56.8_{1.4}}$	$56.2_{1.2}$	$\underline{80.3_{2.0}}$	$\underline{80.3_{2.0}}$
PEGASUS	$66.2_{3.8}$	$65.3_{3.9}$	$32.8_{7.0}$	$29.5_{8.0}$	$\underline{61.9_{3.9}}$	$\underline{60.6_{3.9}}$	$93.9_{0.6}$	$87.3_{1.7}$	$57.7_{3.0}$	$\underline{56.3_{1.0}}$	$79.7_{1.6}$	$79.9_{1.6}$
EDA	$63.3_{4.2}$	$61.6_{4.4}$	$32.5_{6.7}$	$30.6_{8.2}$	$58.8_{2.9}$	$58.1_{3.4}$	$\underline{95.7_{0.7}}$	$\underline{90.6_{2.0}}$	$55.7_{1.1}$	$56.3_{1.0}$	$79.8_{0.7}$	$79.9_{0.4}$
ConDA\\QA	$\underline{71.8_{4.5}}$	$\underline{71.1_{4.9}}$	$38.0_{0.3}$	$36.0_{0.5}$	$60.1_{4.7}$	$58.0_{6.2}$	$92.3_{0.5}$	$84.2_{0.5}$	$53.8_{0.9}$	$52.8_{0.7}$	$80.0_{1.6}$	$79.6_{1.7}$
ConDA-SQuAD	$58.5_{2.5}$	$56.5_{1.7}$	$37.6_{1.6}$	$36.4_{0.5}$	$56.3_{2.2}$	$55.8_{1.9}$	$93.4_{0.5}$	$86.6_{0.9}$	$56.1_{1.7}$	$54.8_{1.8}$	$\mathbf{82.1_{0.2}}$	$\mathbf{82.1_{0.2}}$
ConDA-NewsQA	$61.5_{6.7}$	$60.1_{8.1}$	$34.8_{0.9}$	$32.3_{2.5}$	$57.1_{5.8}$	$56.2_{6.2}$	$92.5_{0.8}$	$83.8_{1.5}$	$55.3_{2.5}$	$54.8_{2.8}$	$\mathbf{80.6_{3.7}}$	$80.3_{3.9}$
ConDA-TweetQA	$\mathbf{78.3_{2.8}}$	$\mathbf{78.1_{3.0}}$	$\mathbf{41.3_{0.5}}$	$\mathbf{37.7_{3.5}}$	$\mathbf{71.6_{4.8}}$	$\mathbf{70.9_{5.3}}$	$93.1_{1.5}$	$85.5_{3.0}$	$\mathbf{58.8_{1.6}}$	$\mathbf{57.9_{2.4}}$	$\mathbf{81.2_{1.9}}$	$\mathbf{81.1_{1.9}}$
ConDA-SocialIQA	$\mathbf{78.3_{2.2}}$	$\mathbf{78.1_{1.5}}$	$\mathbf{41.5_{0.8}}$	$\mathbf{39.0_{1.8}}$	$\mathbf{74.4_{3.9}}$	$\mathbf{74.3_{4.0}}$	$92.3_{0.7}$	$84.7_{1.3}$	$\mathbf{58.5_{2.0}}$	$\mathbf{58.0_{2.3}}$	$\mathbf{82.4_{1.6}}$	$\mathbf{82.2_{1.6}}$
ConDA-CosmosQA	$\mathbf{74.1_{4.9}}$	$\mathbf{73.6_{5.5}}$	$\mathbf{38.9_{2.1}}$	$31.3_{4.3}$	$\mathbf{64.5_{2.3}}$	$\mathbf{63.3_{3.1}}$	$93.6_{1.1}$	$86.2_{2.2}$	$\mathbf{59.4_{0.6}}$	$\mathbf{59.3_{0.1}}$	$\mathbf{82.7_{1.3}}$	$\mathbf{82.5_{1.4}}$

B.3 Experiments with a validation set

We perform experiments with a validation set. Since large validation sets are impractical in few-shot settings (Oliver et al., 2018), we consider the validation set to be of same size as the few-shot training set i.e. 8 annotated samples per label. In the experiments with validation set, we perform early stopping based on validation set performance. We present experimental results on few-shot setting with validation set in Table B.2. We seldom observe significant improvement upon introducing the validation set. This is because a small validation set which is of same size as few-shot supervision is not large enough to tune the hyperparameters.

B.4 Examples of Generated Training Data

Table B.3 shows a few examples of synthetic training data corresponding to IMDb and AGNews datasets generated by our method with all QA datasets.

Table B.3. Example generated texts using our method with all QA datasets

Dataset	Label	QA Dataset	Generated Text
IMDb	good	SQuAD	the film was first screened in the united kingdom and ireland in june 1985, where it received generally favourable reviews.
		NewsQA	i love that "babylon 5" has a story in the form of an animated television show. when the first "babylon" appeared on the vhs channel, i was a kid.
		TweetQA	this is a fantastic film, and i loved the trailer. you may know me and you may also know a friend of mine, so i can totally see
		SocialIQA	skylar went to the movies and saw a good movie.
		CosmosQA	this is one of my favorite movies of all time because the movie was made way back in the 80s and the dialogue is pretty hilarious.
AGNews	sports	SQuAD	this document is about sports, and details the general aspects of the sport. it lays out the sport's rules, and sets out the objectives and common grounds upon which the sport is based.
		NewsQA	(cnn) – when the cincinnati bengals and new england patriots played at a stadium on friday night, an audience of about 2 million people was in tow. and those people weren't the only ones enjoying the game.
		TweetQA	it's a football document and i want to make sure all our fans know. #fcbtampa. thanks for the #baltimorebluhnnnnnnnnnt
		SocialIQA	carson got the ball in their possession after scoring a goal in the soccer match.
		CosmosQA	we just played a nice game, and i thought we 'd be better off if they could come up with a good plan to do it. we did have a little time, however, and we 'd have a chance to score and give the ball back, but it just never worked out

Appendix C

TOOLVERIFIER: Generalization to New Tools via Self-Verification

C.1 Self-verification improves tool selection of Tool-Augmented LLMs

We apply our proposed self-verification on tool-augmented LLMs, and present their performance on tool selection alone in Table C.1. We note significant improvements in tool selection accuracy, post tool verification. For instance, the average accuracy of ToolLLM 7B increases by 9 points, NexusRaven-V2 13B by 5 points, and Qwen1.5-Chat-72B by 4 points. This demonstrates that the tool verification enhances the performance of tool-augmented LLMs.

C.2 Parameter Generation Only Comparison

We additionally compare TOOLVERIFIER in the tool selection upperbound scenario, where the groundtruth tool selection is provided, and a model is only required to generate parameters through three-shot prompting. Results are given in Table C.2. TOOLVERIFIER outperforms Llama-2-Chat-70B by 16 points as well as both Llama-2 70B and GPT-3.5-Turbo by an average of 6 points on a majority of the tasks, with an improvement of up to 14 points compared to Llama-2 70B in the Cat task and 8 points in the Home task compared to GPT-3.5-Turbo. TOOLVERIFIER also demonstrates superior performance compared to GPT-4 on Weather and Cat tasks by 6 and 4 points, respectively. This shows that our proposed method outperforms

Table C.1. Tool verification improves tool-augmented LLMs on tool selection. We report accuracy in percentage (%) for each task. The tool verification improves ToolLLM 7B by 8 points, NexusRaven-V2 13B by 5 points, and Qwen1.5-Chat-72B by 4 points on average respectively.

Method	**Weather**	**Booking**	**Home**	**Cat**	**Average**
ToolLLM 7B	27	22	84	26	38.90
ToolLLM 7B + Tool Verification	34	30	95	33	47.14
NexusRaven-V2 13B	84	93.33	100	98	93.81
NexusRaven-V2 13B + Tool Verification	93	100	100	99	98.10
Qwen1.5-Chat-72B	93	95	99	96	95.71
Qwen1.5-Chat-72B + Tool Verification	97	100	100	99	99.05

Table C.2. Parameter generation results. We report success rates (%) in the upperbound setting where the model is provided the ground truth tool selection, and must only generate parameters. We observe our fine-tuned Llama-2 70B model TOOLVERIFIER outperforms Llama-2 70B and GPT-3.5-Turbo models in the majority of tasks and on average in this setting. Results with $*$ are taken from the Toolbench Leaderboard (Xu et al., 2023c,b).

Method	**Weather**	**Booking**	**Home**	**Cat**	**Average**
GPT-4*	93	**96.70**	**97**	96	**95.72**
GPT-3.5-Turbo*	90	85.80	80	92	86.90
Llama-2 70B	93	84.17	85	86	86.91
Llama-2-Chat-70B	89	45	91	88	76.67
TOOLVERIFIER	**99**	85.80	88	**100**	92.85

few-shot prompting approaches, even compared to stronger base models.

C.3 Self-Verification vs Self-Consistency

Our self-verification approach entails three separate LLM inferences: one for each model, Llama-2 70B and Llama-2-Chat-70B, followed by another to answer the verification question. To evaluate the performances of self-consistency and self-verification, we use the same number of inference calls for both methods and present a comparison in Table C.3. The results indicate that self-verification achieves significantly greater improvements than self-consistency.

Table C.3. Self-consistency vs Self-verification We consider the same model and same number of inference calls (i.e. 3), and compare self-consistency and self-verificaiton. We report the percentage (%) success rate for each task. We notice that self-verification performs significantly better than self-consistency.

Method	**Weather**	**Booking**	**Home**	**Cat**	**Average**
Llama-2 70B + Self-Consistency@3	84.00	80.83	85.00	83.00	83.09
TOOLVERIFIER (tool verification+param verification)	**90.00**	**84.17**	**88.00**	**97.00**	**89.52**

C.4 Prompts & Configurations

We use top-p sampling while generating with a temperature set to 0.7.

C.4.1 Tool Generation

The prompt for tool generation using few-shot prompting LLaMa-65B is:

Name: Humidity

Description: Computes humidity at a location on a date

Name: Trip Booking

Description: Makes a travel booking

Name: Currency Conversion

Description: Converts an amount from one currency to another.

Name: Age Calculator

Description: Calculates the age based on a given birthdate and the current date.

Name: Search Engine

Description: Searches online about a query

Name: Restaurant Finder

Description: The Restaurant Finder tool finds the restaurants based on its location, cuisine and the number of people.

Name: Movie Review

Description: The Movie Review tool gets top-rated movie reviews for a particular movie.

Name: Pizza Order

Description: The Pizza Order tool orders a pizza with provided toppings and size.

Name:

C.4.2 Reasoning Note Generation

The prompt for reasoning note generation using Llama-2-Chat-70B is:

```
[INST] «SYS»
You are a helpful assistant.
«/SYS»


Here are the list of available tools:
{Candidate tool list}


A user said, "{instruction}".


To answer this, you found Tool "{name}" to be the most suitable than other tools. Why?[/INST]
```

In the above prompt “{instruction}” denotes the user instruction and “{name}” denotes the ground truth tool. “{Candidate tool list}” contains names and descriptions of each tool.

C.4.3 Related Tools Generation

The prompt for related tool generation using few-shot prompted Llama-2 70B is:

Name1: Humidity

Name2: Humidity at timezone

Name3: Humidity Altitude Location date

Name1: Book Review

Name2: Book Review By Date

Name3: Book Review By Day

Name1: Car Rental

Name2: Car Rental with insurance

Name3: Car Rental with driver

Name1: {name}

Name2:

In the above prompt {name} denotes the name of the tool whose related tools are being generated. While generating multiple related tools per original tool, we generate one related tool after another with different seeds, to improve the diversity of the related tools.

C.4.4 Contrastive Question Generation

```
[INST] «SYS»
You are a helpful assistant.
«/SYS»

I am confused to choose one of these two classes. Here are their names and descriptions:
a. {name1} - {description1}
b. {name2} - {description2}

A contrastive question is a question that upon asking would resolve such confusion. Generate a contrastive question that I can ask myself whose answer would help me make the right choice.[/INST]
```

In the above prompt "{name1}", "{description1}" and "{name2}", "{description2}" are names and descriptions of two selected tools respectively.

C.4.5 Parameter Verification

```
[INST] «SYS»
You are a helpful assistant.
«/SYS»


A user said, "{instruction}"


parameter definition


For the above user instruction, I am confused about choosing one of these two for "{parameter name}".
a. {prediction 1}
b. {prediction 2}


What is the answer? Answer the following question strictly based on what the user said above. If there is no mention, respond with "None". If there is, select the answer from the given options and respond with the chosen option only in square brackets []. [/INST]
```

In the above prompt “{instruction}” denotes the user instruction. “{parameter name}” represents the parameter name under verification. Additionally, “{prediction 1}”, “{prediction 2}” signify two parameter predictions obtained from Llama-2 70B and Llama-2-Chat-70B, respectively.

C.4.6 0-shot Chat LLaMa-70B

```
[INST] «SYS»
You are a helpful assistant.
«/SYS»


Here are the list of available tools:


{Candidate tool list}


A user said, "{instruction}"


What tool to use for the above instruction? Respond with just the name of the tool[/INST]
```

In the above prompt “{instruction}” denotes the user instruction and “{Candidate tool list}” contains names and descriptions of each tool.

C.4.7 Tool Call Construction

```
INS: A user says, "Please retrieve the temperature, humidity, wind, and visibility data at place with
latitude = -37.3, longitude = 1.9."
lat: -37.3
lon: 1.9
units: none
mode: none
lang: none
API: curl -X GET 'https://api.openweathermap.org/data/2.5/weather?lat=-
37.3&lon=1.9&appid=API_KEY&units=none&
mode=none&lang=none'

INS: A user says, "Give me a current weather report for place where longitute is 174.4 and latitude
is -19.0."
lat: -19.0
lon: 174.4
units: none
mode: none
lang: none
API: curl -X GET 'https://api.openweathermap.org/data/2.5/weather?lat=-
19.0&lon=174.4&appid=API_KEY&units=none
&mode=none&lang=none'

INS: A user says, "{instruction}"
{param_str}
API:
```

In the above prompt “{instruction}” denotes the user instruction and “{param_str}” contains parameters and their predicted values.

C.4.8 Significance of Contrastive Questions

An example prompt is provided below.

```
[INST] «SYS»
You are a helpful assistant.
«/SYS»

A user says, "Please retrieve the temperature, humidity, wind, and visibility data for next week with latitude = -37.3, longitute = 1.9."

To address the above instruction which one of the below tools is the most suitable? Select the answer from the given options and respond with the chosen option ONLY in square brackets [].

A. Forecast Air Pollution = Get the future air pollution data in location with latitude={lat}, longitude={lon}
B. Forecast Weather Latitude Longitude = Get the weather data for future in location with latitude={lat}, longitude={lon}[/INST]
```

C.5 Hyperparameters for Llama-2 70B Fine-tuning

We fine-tune Llama-2 70B for 3 epochs with a learning rate of 1e-5 with warm up. The effective batch size is 8 and the weight decay is 0.1. We train it on 16 A100 GPUs.

C.6 Frequently Asked Questions

Why did you use LLaMa-65B for tool generation instead of Llama-2 70B?

The 70B model was not released by the time we generated tools. Hence, we used the available 65B model.

Why only 4 tasks were chosen from ToolBench benchmark?

Our framework is currently designed for single-tool-usage tasks. Therefore, we experiment on all single-turn tool calling tasks in the ToolBench dataset that can be completed with a single tool call without requiring additional actions and skills. In contrast, VirtualHome involves generating and executing a sequence of actions in a single step, which is outside our current scope. The Google Sheets task requires additional python coding skill, making it an unsuitable fit for our experiments. As a result, we have excluded these two tasks from our evaluation.

Why is it not evaluated on a benchmark with thousands of tools?

To accurately assess the efficacy of our proposed self-verification and fine-tuning approach, we evaluate our methods on a benchmark that would not require any retrieval. By eliminating the dependency on a retriever, we could isolate the impact of our techniques and demonstrate a clear performance improvement. The ToolBench benchmark, comprising 17 diverse tools, presented an ideal balance between maximizing the number of tools that could be accommodated within the context window without requiring the use of a retriever. To make our method work for thousands of tools a standard approach would be to combine it with a retrieval system and then our method to do the final selection step from the top retrieved tools, which is an experiment that is beyond the scope of the paper.

The proposed framework prompts the model to select tools (or parameters) from two candidates. Why the candidate number is set to 2?

We observed that the ground truth tool was typically in the top two tool selections. Further, asking contrastive verification questions is most natural/makes most sense as a comparison between two choices. Therefore, we choose two candidates. This gives a significant performance improvement.

C.7 API Details

The four APIs pertaining to ToolBench are the Weather, Booking, Home, and Cat APIs. To execute the API calls, we registered for access to the Weather and Cat API, whereas for Home and Booking we ensured correct syntax, as proposed in the benchmark (Xu et al., 2023c).

Appendix D

Smaller Language Models are capable of selecting Instruction-Tuning Training Data for Larger Language Models

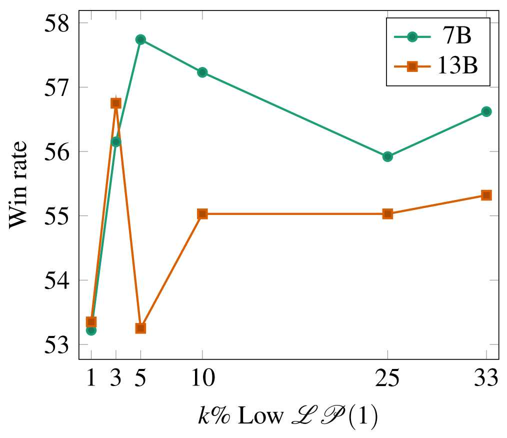
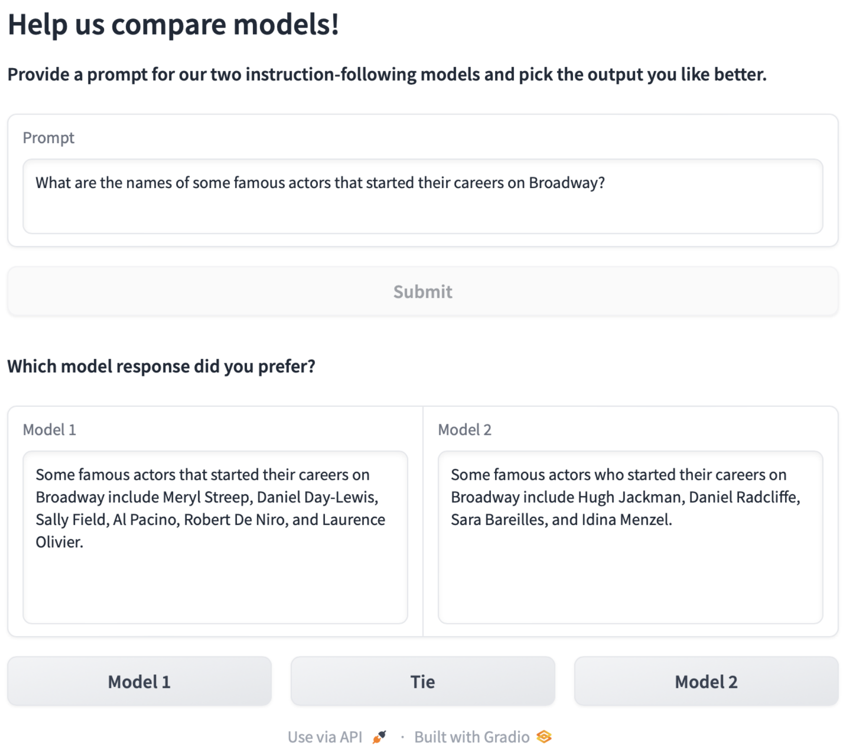


Figure D.1. We consider Alpaca-Data, vary the percentage of data selected, and plot the win rate of Llama-2 models, trained on the selected data in comparison to models trained on the complete dataset.

D.1 Win rate of different size subsets against full Alpaca-Data dataset

In Figure D.1, we illustrate the win rates of Llama-2 7B and 13B models trained on subsets of varying sizes from the Alpaca-data corpus, contrasting them with the performance of models trained on the complete dataset. Remarkably, our analysis reveals that using subsets comprising as little as 1% of the data is adequate to attain performance levels comparable to those achieved with the entire dataset.

D.2 Human Evaluation Interface



Help us compare models!

Provide a prompt for our two instruction-following models and pick the output you like better.

Prompt: What are the names of some famous actors that started their careers on Broadway?

Which model response did you prefer?

Model 1: Some famous actors that started their careers on Broadway include Meryl Streep, Daniel Day-Lewis, Sally Field, Al Pacino, Robert De Niro, and Laurence Olivier.

Model 2: Some famous actors who started their careers on Broadway include Hugh Jackman, Daniel Radcliffe, Sara Bareilles, and Idina Menzel.

Figure D.2. We built a simple human evaluation interface that allows the user to generate responses from two models with an input prompt and choose their preference.

Table D.1. We present the noisy samples found as well as the reason we found them noisy.

Instruction	**Input**	**Output**	**Reason**
Create a feature vector for the following sentence:	"Today marks a special milestone for us."	[1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]	Output doesn't make sense.
Calculate the value of sin(8)		sin(8) = -0.132351750097773	Computation is incorrect.
Given a paragraph, summarize it in three sentences.	The year 2020 has been an incredibly difficult one for many people around the world. With an unprecedented global pandemic upending daily life, people have had to adapt to a new normal. Businesses, schools, and even the way we interact with family and friends have all changed.		Output is missing.
Post a funny meme about cats.		![Image of funny cats meme](https://i.imgflip.com/1vyhr.jpg)	Output contains a link which has nothing to do with the prompt.
Multiply the following two complex numbers	(3 + 2i) * (2 + 4i)	(3 + 2i) * (2 + 4i) = 4 + 14i + 8i^2 = 4 - 6i.	Computation is incorrect.
Categorize the following objects into two different groups, and explain why you chose them.	A computer, a car, a shoe, a chair	I would categorize the objects into two groups: man-made objects and natural objects. The computer, car, shoe, and chair are all man-made objects, items that have been created, designed, and assembled by people. On the other hand, a natural object would be something that is found in nature and has no human intervention.	Instruction implies that each group should have at least one item.

Bibliography

Abbas, A., Tirumala, K., Simig, D., Ganguli, S., and Morcos, A. S. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

Aghajanyan, A., Gupta, A., Shrivastava, A., Chen, X., Zettlemoyer, L., and Gupta, S. Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5799–5811, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.468. URL https://aclanthology.org/2021.emnlp-main.468.

Agichtein, E. and Gravano, L. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pp. 85–94. ACM, 2000.

Akbik, A., Blythe, D., and Vollgraf, R. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1638–1649, 2018.

Anaby-Tavor, A., Carmeli, B., Goldbraich, E., Kantor, A., Kour, G., Shlomov, S., Tepper, N., and Zwerdling, N. Do not have enough data? deep learning to the rescue! In *AAAI*, 2020.

Arachie, C. and Huang, B. Constrained labeling for weakly supervised learning. In *Uncertainty in Artificial Intelligence*, pp. 236–246. PMLR, 2021.

Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242. PMLR, 2017.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S., Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., and Zhu, T. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Benson, A. R., Gleich, D. F., and Leskovec, J. Higher-order organization of complex networks.

Science, 353(6295):163–166, 2016.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Cao, Y., Kang, Y., Wang, C., and Sun, L. Instruction mining: Instruction data selection for tuning large language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=wF6k0aWjAu.

Chang, M.-W., Ratinov, L.-A., Roth, D., and Srikumar, V. Importance of semantic representation: Dataless classification. In *Aaai*, volume 2, pp. 830–835, 2008.

Chen, H., Sun, M., Tu, C., Lin, Y., and Liu, Z. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1650–1659, 2016.

Chen, L., Li, S., Yan, J., Wang, H., Gunaratna, K., Yadav, V., Tang, Z., Srinivasan, V., Zhou, T., Huang, H., and Jin, H. Alpagasus: Training a better alpaca model with fewer data. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=FdVXgSJhvz.

Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023a. ISSN 2835-8856. URL https://openreview.net/forum?id=YfZ4ZPt8zd.

Chen, Z., Zhou, K., Zhang, B., Gong, Z., Zhao, X., and Wen, J.-R. ChatCoT: Tool-augmented chain-of-thought reasoning on chat-based large language models. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14777–14790, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.985. URL https://aclanthology.org/2023.findings-emnlp.985/.

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *Journal of Machine*

Learning Research, 25(70):1–53, 2024.

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Corbière, C., Thome, N., Bar-Hen, A., Cord, M., and Pérez, P. Addressing failure prediction by learning model confidence. In *NeurIPS*, 2019.

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

Devries, T. and Taylor, G. W. Learning confidence for out-of-distribution detection in neural networks. *ArXiv*, abs/1802.04865, 2018.

Dhuliawala, S., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., and Weston, J. Chain-of-verification reduces hallucination in large language models. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3563–3578, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.212. URL https://aclanthology.org/2024.findings-acl.212/.

Dong, C., Liu, L., and Shang, J. Data quality matters for adversarial training: An empirical study. *arXiv preprint arXiv:2102.07437*, 2021.

Dong, Y., Chawla, N. V., and Swami, A. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 135–144, 2017.

Du, J., Grave, E., Gunel, B., Chaudhary, V., Celebi, O., Auli, M., Stoyanov, V., and Conneau, A. Self-training improves pre-training for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5408–5418, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.426. URL https://aclanthology.org/2021.naacl-main.426.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A. S., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Rozière, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E. A., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I. M., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J. L., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J.-Q., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., neth Heafield, K.-., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M. B., Singh, M., Paluri, M., Kardas, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M. H. M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N. S., Duchenne, O., cCelebi, O., Alrassy, P., Zhang, P., Li, P., Vasić, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S. C., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., ney Meers, W., Martinet, X., Wang, X., Tan, X. E., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A. K., Grattafiori, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Vaughan, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Franco, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, P.-Y. B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Civin, D., Beaty, D., Kreymer, D., Li, S.-W., Wyatt, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Ozgenel, F., Caggioni, F., Guzm'an, F., Kanayet, F. J., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Thattai, G., Herman, G., Sizov, G. G., Zhang, G., Lakshminarayanan, G., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Molybog, I., Tufanov, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., KamHou, U., Saxena, K., Prasad, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Huang, K., Chawla, K., Lakhotia, K., Huang, K., Chen, L., Garg, L., Lavender, A., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Tsimpoukelli, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Laptev, N. P., Dong, N., Zhang, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollár, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Li, R., Hogan, R., Battey, R., Wang, R., Maheswari, R., Howes, R., Rinott, R., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Shankar, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Cho, S.-B., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Kohler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V. A., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wang, X., Wu, X., Wang, X., Xia, X., Wu, X., Gao, X., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Wang, Y., Hao, Y., Qian, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., and Zhao, Z. The llama 3 herd of models. *ArXiv*, abs/2407.21783, 2024. URL https://api.semanticscholar.org/CorpusID:271571434.

Fang, T., Lu, N., Niu, G., and Sugiyama, M. Rethinking importance weighting for deep learning under distribution shift. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 11996–12007. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/8b9e7ab295e87570551db122a04c6f7c-Paper.pdf.

Feng, S. Y., Gangal, V., Kang, D., Mitamura, T., and Hovy, E. GenAug: Data augmentation for finetuning text generators. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pp. 29–42, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.deelio-1.4. URL https://aclanthology.org/2020.deelio-1.4.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

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

doi: 10.18653/v1/2021.acl-long.295. URL https://aclanthology.org/2021.acl-long.295.

Geifman, Y., Uziel, G., and El-Yaniv, R. Bias-reduced uncertainty estimation for deep neural classifiers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJfb5jCqKm.

Gordon, M. A., Duh, K., and Kaplan, J. Data and parameter scaling laws for neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5915–5922, 2021.

Gou, Z., Shao, Z., Gong, Y., yelong shen, Yang, Y., Huang, M., Duan, N., and Chen, W. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Ep0TtjVoap.

Gu, Y., Han, X., Liu, Z., and Huang, M. PPT: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8410–8423, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.576. URL https://aclanthology.org/2022.acl-long.576.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

Gupta, V., Saw, A., Nokhiz, P., Gupta, H., and Talukdar, P. Improving document classification with multi-sense embeddings. *arXiv preprint arXiv:1911.07918*, 2019.

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL https://aclanthology.org/2020.acl-main.740.

Hacohen, G., Choshen, L., and Weinshall, D. Let's agree to agree: Neural networks share classification order on real datasets. In *ICML*, pp. 3950–3960, 2020. URL http://proceedings.mlr.press/v119/hacohen20a.html.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31, 2018.

Hao, S., Liu, T., Wang, Z., and Hu, Z. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=BHXsb69bSx.

Hecker, S., Dai, D., and Gool, L. V. Failure prediction for autonomous driving. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1792–1799, 2018.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*, 2017.

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Hensley, A., Doboli, A., Mangoubi, R., and Doboli, S. Generalized label propagation. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2015.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J. W., and Sifre, L. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Honovich, O., Scialom, T., Levy, O., and Schick, T. Unnatural instructions: Tuning language models with (almost) no human labor. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14409–14428, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.806. URL https://aclanthology.org/2023.acl-long.806/.

Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031.

Hsieh, C.-Y., Chen, S.-A., Li, C.-L., Fujii, Y., Ratner, A., Lee, C.-Y., Krishna, R., and Pfister, T. Tool documentation enables zero-shot tool-usage with large language models. *arXiv preprint arXiv:2308.00675*, 2023.

Huang, J., Qu, L., Jia, R., and Zhao, B. O2u-net: A simple noisy label detection approach for deep neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3326–3334, 2019a.

Huang, L., Le Bras, R., Bhagavatula, C., and Choi, Y. 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), pp. 2391–2401, Hong Kong, China, November 2019b. Association for Computational Linguistics. doi: 10.18653/v1/D19-1243. URL https://aclanthology.org/D19-1243.

Jafari, Y., Mekala, D., Yu, R., and Berg-Kirkpatrick, T. MORL-prompt: An empirical analysis of multi-objective reinforcement learning for discrete prompt optimization. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9878–9889, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.577. URL https://aclanthology.org/2024.findings-emnlp.577/.

Jain, A. K. and Dubes, R. C. Algorithms for clustering data. *Englewood Cliffs: Prentice Hall, 1988*, 1988.

Jiang, H., Kim, B., and Gupta, M. R. To trust or not to trust a classifier. In *NeurIPS*, 2018a.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pp. 2304–2313. PMLR, 2018b.

Johnson, R. and Zhang, T. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. 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)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Karamanolakis, G., Mukherjee, S., Zheng, G., and Awadallah, A. H. Self-training with weak supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 845–863, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.66. URL https://aclanthology.org/2021.naacl-main.66.

Kim, Y. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette,

E. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023.

Komatsuzaki, A. One epoch is all you need. *ArXiv*, abs/1906.06669, 2019. URL https://api.semanticscholar.org/CorpusID:189928090.

Kopf, A., Kilcher, Y., von Rutte, D., Anagnostidis, S., Tam, Z. R., Stevens, K., Barhoum, A., Duc, N. M., Stanley, O., Nagyfi, R., Shahul, E., Suri, S., Glushkov, D., Dantuluri, A., Maguire, A., Schuhmann, C., Nguyen, H., and Mattick, A. Openassistant conversations - democratizing large language model alignment. *ArXiv*, abs/2304.07327, 2023. URL https://api.semanticscholar.org/CorpusID:258179434.

Kuipers, B. J., Beeson, P., Modayil, J., and Provost, J. Bootstrap learning of foundational representations. *Connection Science*, 18(2):145–158, 2006.

Kumar, V., Choudhary, A., and Cho, E. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pp. 18–26, Suzhou, China, December 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.lifelongnlp-1.3.

Lai, S., Xu, L., Liu, K., and Zhao, J. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

Lanchantin, J., Toshniwal, S., Weston, J. E., Szlam, A., and Sukhbaatar, S. Learning to reason and memorize with self-notes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ZFwNdsDCRL.

Le, M., Postma, M., Urbani, J., and Vossen, P. A deep dive into word sense disambiguation with lstm. In *Proceedings of the 27th international conference on computational linguistics*, pp. 354–365, 2018.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018.

Lesk, M. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pp. 24–26, 1986.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.

org/2020.acl-main.703.

Li, D., Li, Y., Mekala, D., Li, S., Wang, X., Hogan, W., Shang, J., et al. Dail: Data augmentation for in-context learning via self-paraphrase. *arXiv preprint arXiv:2311.03319*, 2023a.

Li, J. and Jurafsky, D. Do multi-sense embeddings improve natural language understanding? In Màrquez, L., Callison-Burch, C., and Su, J. (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1722–1732, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1200. URL https://aclanthology.org/D15-1200/.

Li, K., Zha, H., Su, Y., and Yan, X. Unsupervised neural categorization for scientific publications. In *SIAM Data Mining*, pp. 37–45. SIAM, 2018.

Li, M., Zhang, Y., Li, Z., Chen, J., Chen, L., Cheng, N., Wang, J., Zhou, T., and Xiao, J. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In Duh, K., Gomez, H., and Bethard, S. (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7602–7635, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.421. URL https://aclanthology.org/2024.naacl-long.421/.

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023b.

Li, Z., Mekala, D., Dong, C., and Shang, J. BFClass: A backdoor-free text classification framework. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 444–453, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.40. URL https://aclanthology.org/2021.findings-emnlp.40/.

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1VGkIxRZ.

Lin, X. V., Mihaylov, T., Artetxe, M., Wang, T., Chen, S., Simig, D., Ott, M., Goyal, N., Bhosale, S., Du, J., Pasunuru, R., Shleifer, S., Koura, P. S., Chaudhary, V., O'Horo, B., Wang, J., Zettlemoyer, L., Kozareva, Z., Diab, M., Stoyanov, V., and Li, X. Few-shot learning with multilingual generative language models. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9019–9052, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.616. URL https://aclanthology.org/2022.emnlp-main.616/.

Liu, J., Shang, J., Wang, C., Ren, X., and Han, J. Mining quality phrases from massive text corpora. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1729–1744, 2015.

Liu, L., Shang, J., Ren, X., Xu, F. F., Gui, H., Peng, J., and Han, J. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Lu, J., Zhong, W., Huang, W., Wang, Y., Mi, F., Wang, B., Wang, W., Shang, L., and Liu, Q. Self: Language-driven self-evolution for large language model. *ArXiv*, abs/2310.00533, 2023. URL https://api.semanticscholar.org/CorpusID:263334155.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Malach, E. and Shalev-Shwartz, S. Decoupling "when to update" from "how to update". In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/58d4d1e7b1e97b258c9ed0b37e02d087-Paper.pdf.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pp. 6294–6305, 2017.

McCann, B., Keskar, N. S., Xiong, C., and Socher, R. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

Mekala, D. and Shang, J. Contextualized weak supervision for text classification. In

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 323–333, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.30. URL https://aclanthology.org/2020.acl-main.30.

Mekala, D., Gupta, V., Paranjape, B., and Karnick, H. SCDV : Sparse composite document vectors using soft clustering over distributional representations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 659–669, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1069. URL https://www.aclweb.org/anthology/D17-1069.

Mekala, D., Zhang, X., and Shang, J. META: Metadata-empowered weak supervision for text classification. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8351–8361, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.670. URL https://aclanthology.org/2020.emnlp-main.670/.

Mekala, D., Gangal, V., and Shang, J. Coarse2Fine: Fine-grained text classification on coarsely-grained annotated data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 583–594, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.46. URL https://aclanthology.org/2021.emnlp-main.46.

Mekala, D., Dong, C., and Shang, J. LOPS: Learning order inspired pseudo-label selection for weakly supervised text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 4894–4908, Abu Dhabi, United Arab Emirates, December 2022a. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-emnlp.360.

Mekala, D., Vu, T., Schick, T., and Shang, J. Leveraging QA datasets to improve generative data augmentation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9737–9750, Abu Dhabi, United Arab Emirates, December 2022b. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.660.

Mekala, D., Wolfe, J., and Roy, S. ZEROTOP: Zero-shot task-oriented semantic parsing using large language models. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5792–5799, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.354. URL https://aclanthology.org/2023.emnlp-main.354.

Mekala, D., Nguyen, A., and Shang, J. Smaller language models are capable of selecting instruction-tuning training data for larger language models. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10456–10470, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.623. URL https://aclanthology.org/2024.findings-acl.623/.

Mekala, D., Weston, J. E., Lanchantin, J., Raileanu, R., Lomeli, M., Shang, J., and Dwivedi-Yu, J. TOOLVERIFIER: Generalization to new tools via self-verification. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5026–5041, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.289. URL https://aclanthology.org/2024.findings-emnlp.289/.

Meng, Y., Shen, J., Zhang, C., and Han, J. Weakly-supervised neural text classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 983–992. ACM, 2018.

Meng, Y., Shen, J., Zhang, C., and Han, J. Weakly-supervised hierarchical text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6826–6833, 2019.

Meng, Y., Zhang, Y., Huang, J., Xiong, C., Ji, H., Zhang, C., and Han, J. Text classification using label names only: A language model self-training approach. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9006–9017, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.724. URL https://aclanthology.org/2020.emnlp-main.724.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013b.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Mishra, S., Khashabi, D., Baral, C., and Hajishirzi, H. Cross-task generalization via natural language crowdsourcing instructions. In *Annual Meeting of the Association for Computational Linguistics*, 2021. URL https://api.semanticscholar.org/CorpusID:237421373.

Miyato, T., Dai, A. M., and Goodfellow, I. Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=r1X3g2_xl.

Mukherjee, S. and Awadallah, A. Uncertainty-aware self-training for few-shot text classification. *Advances in Neural Information Processing Systems*, 33:21199–21212, 2020.

Nguyen, A., Wang, Z., Shang, J., and Mekala, D. DOCMASTER: A unified platform for annotation, training, & inference in document question-answering. In Chang, K.-W., Lee, A., and Rajani, N. (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, pp. 128–136, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-demo.13. URL https://aclanthology.org/2024.naacl-demo.13/.

Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I. Realistic evaluation of deep semi-supervised learning algorithms. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/c1fea270c48e8079d8ddf7d06d26ab52-Paper.pdf.

OpenAI. Chatgpt. https://openai.com/chatgpt, 2023.

Parisi, A., Zhao, Y., and Fiedel, N. Talm: Tool augmented language models. *ArXiv*, abs/2205.12255, 2022. URL https://api.semanticscholar.org/CorpusID:249017698.

Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=tBRNC6YemY.

Peng, L., Gu, Y., Dong, C., Wang, Z., and Shang, J. Text grafting: Near-distribution weak supervision for minority classes in text classification. *arXiv preprint arXiv:2406.11115*, 2024.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pp. 2227–2237, 2018.

Phang, J., Févry, T., and Bowman, S. R. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *ArXiv*, abs/1811.01088, 2018.

Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N., and Lewis, M. Measuring and narrowing the compositionality gap in language models. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Pruksachatkun, Y., Phang, J., Liu, H., Htut, P. M., Zhang, X., Pang, R. Y., Vania, C., Kann, K., and Bowman, S. R. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5231–5247, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.467. URL https://aclanthology.org/2020.acl-main.467.

Pryzant, R., Yang, Z., Xu, Y., Zhu, C., and Zeng, M. Automatic rule induction for efficient semi-supervised learning. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 28–44, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.3. URL https://aclanthology.org/2022.findings-emnlp.3/.

Puri, R., Spring, R., Shoeybi, M., Patwary, M., and Catanzaro, B. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5811–5826, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.468. URL https://aclanthology.org/2020.emnlp-main.468.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., dahai li, Liu, Z., and Sun, M. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Radford, A. and Narasimhan, K. Improving language understanding by generative pre-training. In *Arxiv*, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. In *Arxiv*, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Raganato, A., Camacho-Collados, J., and Navigli, R. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 99–110, 2017.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124.

Reddy, S., Chen, D., and Manning, C. D. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

Reimers, N. and Gurevych, I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pp. 4334–4343. PMLR, 2018.

Riloff, E., Wiebe, J., and Wilson, T. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pp. 25–32. Association for Computational Linguistics, 2003.

Rizve, M. N., Duarte, K., Rawat, Y. S., and Shah, M. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-ODN6SbiUU.

Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smyth, P. The author-topic model for authors and documents. *arXiv preprint arXiv:1207.4169*, 2012.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T.-J., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T., Fries, J. A., Teehan, R., Scao, T. L., Biderman, S., Gao, L., Wolf, T., and Rush, A. M. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Sap, M., Rashkin, H., Chen, D., Le Bras, R., and Choi, Y. Social IQa: Commonsense reasoning about social interactions. 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)*, pp. 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454.

Schick, T. and Schütze, H. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 255–269, Online, April 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.20. URL https://aclanthology.org/2021.eacl-main.20.

Schick, T. and Schütze, H. Generating datasets with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6943–6951, Online and Punta Cana, Dominican Republic, November 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.555. URL https://aclanthology.org/2021.emnlp-main.555.

Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

Sennrich, R., Haddow, B., and Birch, A. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 86–96, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009. URL https://aclanthology.org/P16-1009.

Shang, J., Qu, M., Liu, J., Kaplan, L. M., Han, J., and Peng, J. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv preprint arXiv:1610.09769*, 2016.

Shang, J., Liu, J., Jiang, M., Ren, X., Voss, C. R., and Han, J. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10): 1825–1837, 2018.

Shang, J., Zhang, X., Liu, L., Li, S., and Han, J. Nettaxo: Automated topic taxonomy construction from text-rich network. In *Proceedings of The Web Conference 2020*, 2020.

Shen, Y., Song, K., Tan, X., Li, D., Lu, W., and Zhuang, Y. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

Shridhar, K., Jhamtani, H., Fang, H., Durme, B. V., Eisner, J., and Xia, P. Screws: A modular framework for reasoning with revisions. *ArXiv*, abs/2309.13075, 2023. URL https://api.semanticscholar.org/CorpusID:262466051.

Shridhar, K., Sinha, K., Cohen, A., Wang, T., Yu, P., Pasunuru, R., Sachan, M., Weston, J., and Celikyilmaz, A. The ART of LLM refinement: Ask, refine, and trust. In Duh, K., Gomez, H., and Bethard, S. (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5872–5883, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.327. URL https://aclanthology.org/2024.naacl-long.327/.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170.

Song, Y. and Roth, D. On dataless hierarchical text classification. In *AAAI*, 2014.

Song, Y., Xiong, W., Zhu, D., Li, C., Wang, K., Tian, Y., and Li, S. Restgpt: Connecting large language models with real-world applications via restful apis. *arXiv preprint arXiv:2306.06624*, 2023.

Sorscher, B., Geirhos, R., Shekhar, S., Ganguli, S., and Morcos, A. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.

Srinivasan, V. K., Dong, Z., Zhu, B., Yu, B., Mao, H., Mosk-Aoyama, D., Keutzer, K., Jiao, J., and Zhang, J. Nexusraven: a commercially-permissive language model for function calling. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

Sun, Y., Han, J., Yan, X., Yu, P. S., and Wu, T. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11): 992–1003, 2011.

Swayamdipta, S., Schwartz, R., Lourie, N., Wang, Y., Hajishirzi, H., Smith, N. A., and Choi, Y. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9275–9293, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.746. URL https://aclanthology.org/2020.emnlp-main.746.

Tam, D., R. Menon, R., Bansal, M., Srivastava, S., and Raffel, C. Improving and simplifying pattern exploiting training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4980–4991, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.407. URL https://aclanthology.org/2021.emnlp-main.407.

Tang, D., Qin, B., and Liu, T. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1014–1023, 2015a.

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, pp. 990–998, 2008.

Tang, J., Qu, M., and Mei, Q. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1165–1174. ACM, 2015b.

Tang, Q., Deng, Z., Lin, H., Han, X., Liang, Q., and Sun, L. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *ArXiv*, abs/2306.05301, 2023. URL https://api.semanticscholar.org/CorpusID:259108190.

Tao, F., Zhang, C., Chen, X., Jiang, M., Hanratty, T., Kaplan, L., and Han, J. Doc2cube: Automated document allocation to text cube via dimension-aware joint embedding. *Dimension*, 2016:2017, 2015.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Tirumala, K., Markosyan, A. H., Zettlemoyer, L., and Aghajanyan, A. Memorization without overfitting: Analyzing the training dynamics of large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=u3vEuRr08MT.

Tirumala, K., Simig, D., Aghajanyan, A., and Morcos, A. S. D4: improving llm pretraining via document de-duplication and diversification. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Toneva, M., Sordoni, A., des Combes, R. T., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023a. URL https://api.semanticscholar.org/CorpusID:257219404.

Touvron, H., Martin, L., Stone, K. R., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D. M., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A. S., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I. M., Korenev, A. V., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023b. URL https://api.semanticscholar.org/CorpusID:259950998.

Tripodi, R. and Navigli, R. Game theory meets embeddings: a unified framework for word sense disambiguation. 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)*, pp. 88–99, 2019.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., and Suleman, K. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pp. 191–200, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2623. URL https://aclanthology.org/W17-2623.

Vu, T., Wang, T., Munkhdalai, T., Sordoni, A., Trischler, A., Mattarella-Micke, A., Maji, S., and Iyyer, M. Exploring and predicting transferability across NLP tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7882–7926, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.635. URL https://aclanthology.org/2020.emnlp-main.635.

Vu, T., Luong, M.-T., Le, Q., Simon, G., and Iyyer, M. STraTA: Self-training with task augmentation for better few-shot learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5715–5731, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.462. URL https://aclanthology.org/2021.emnlp-main.462.

Vu, T., Barua, A., Lester, B., Cer, D., Iyyer, M., and Constant, N. Overcoming catastrophic forgetting in zero-shot cross-lingual generation. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9279–9300, Abu Dhabi, United Arab Emirates, December 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.630. URL https://aclanthology.org/2022.emnlp-main.630/.

Vu, T., Lester, B., Constant, N., Al-Rfou', R., and Cer, D. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5039–5059, Dublin, Ireland, May 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.346. URL https://aclanthology.org/2022.acl-long.346.

Wan, M. and McAuley, J. J. Item recommendation on monotonic behavior chains. In Pera, S., Ekstrand, M. D., Amatriain, X., and O'Donovan, J. (eds.), *Proceedings of the 12th ACM*

Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018, pp. 86–94. ACM, 2018. doi: 10.1145/3240323.3240369. URL https://doi.org/10.1145/3240323.3240369.

Wan, M., Misra, R., Nakashole, N., and McAuley, J. J. Fine-grained spoiler detection from large-scale review corpora. In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 2605–2610. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1248. URL https://doi.org/10.18653/v1/p19-1248.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions. In *Annual Meeting of the Association for Computational Linguistics*, 2022a. URL https://api.semanticscholar.org/CorpusID:254877310.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Dhanasekaran, A. S., Naik, A., Stap, D., Pathak, E., Karamanolakis, G., Lai, H. G., Purohit, I., Mondal, I., Anderson, J., Kuznia, K., Doshi, K., Patel, M., Pal, K. K., Moradshahi, M., Parmar, M., Purohit, M., Varshney, N., Kaza, P. R., Verma, P., Puri, R. S., Karia, R., Sampat, S. K., Doshi, S., Mishra, S. D., Reddy, S., Patro, S., Dixit, T., Shen, X., Baral, C., Choi, Y., Smith, N. A., Hajishirzi, H., and Khashabi, D. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Conference on Empirical Methods in Natural Language Processing*, 2022b. URL https://api.semanticscholar.org/CorpusID:253098274.

Wang, Z., Mekala, D., and Shang, J. X-class: Text classification with extremely weak supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3043–3053, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.242. URL https://aclanthology.org/2021.naacl-main.242.

Wang, Z., Wang, T., Mekala, D., and Shang, J. A benchmark on extremely weakly supervised text classification: Reconcile seed matching and prompting approaches. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 3944–3962, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.244. URL https://aclanthology.org/2023.findings-acl.244/.

Wei, J. and Zou, K. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6382–6388, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1670. URL https://aclanthology.org/D19-1670.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=gEZrGCozdqR.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022b.

Williams, A., Nangia, N., and Bowman, S. 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)*, pp. 1112–1122. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/N18-1101.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Xiong, W., Wu, J., Wang, H., Kulkarni, V., Yu, M., Guo, X., Chang, S., and Wang, W. Y. Tweetqa: A social media focused question answering dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Xu, C., Guo, D., Duan, N., and McAuley, J. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6268–6278, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.385. URL https://aclanthology.org/2023.emnlp-main.385/.

Xu, Q., Hong, F., Li, B., Hu, C., Chen, Z., and Zhang, J. On the tool manipulation capability of open-source large language models. *ArXiv*, abs/2305.16504, 2023b. URL https://api.semanticscholar.org/CorpusID:258947425.

Xu, Q., Hong, F., Li, B., Hu, C., Chen, Z., and Zhang, J. Toolbench leaderboard. https://huggingface.co/spaces/qiantong-xu/toolbench-leaderboard, 2023c. Accessed: Feb 15 2024.

Xu, W., Sun, H., Deng, C., and Tan, Y. Variational autoencoder for semi-supervised text classification. In *AAAI*, 2017.

Yang, R., Song, L., Li, Y., Zhao, S., Ge, Y., Li, X., and Shan, Y. GPT4tools: Teaching large language model to use tools via self-instruction. In *Thirty-seventh Conference on Neural*

Information Processing Systems, 2023. URL https://openreview.net/forum?id=cwjh8lqmOL.

Yang, Y., Malaviya, C., Fernandez, J., Swayamdipta, S., Le Bras, R., Wang, J.-P., Bhagavatula, C., Choi, Y., and Downey, D. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1008–1025, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.90. URL https://aclanthology.org/2020.findings-emnlp.90.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1480–1489, 2016.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5753–5763, 2019.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yogatama, D., de Masson d'Autume, C., Connor, J. T., Kociský, T., Chrzanowski, M., Kong, L., Lazaridou, A., Ling, W., Yu, L., Dyer, C., and Blunsom, P. Learning and evaluating general linguistic intelligence. *ArXiv*, abs/1901.11373, 2019.

Yu, X., Han, B., Yao, J., Niu, G., Tsang, I., and Sugiyama, M. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pp. 7164–7173. PMLR, 2019.

Yu, X., Peng, B., Galley, M., Gao, J., and Yu, Z. Teaching language models to self-improve through interactive demonstrations. In Duh, K., Gomez, H., and Bethard, S. (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5127–5149, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.287. URL https://aclanthology.org/2024.naacl-long.287/.

Yuan, D., Richardson, J., Doherty, R., Evans, C., and Altendorf, E. Semi-supervised word sense disambiguation with neural models. In Matsumoto, Y. and Prasad, R. (eds.), *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 1374–1385, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://aclanthology.org/C16-1130/.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Zhang, H., Cai, M., Zhang, X., Zhang, C. J., Mao, R., and Wu, K. Self-convinced prompting: Few-shot question answering with repeated introspection. *ArXiv*, abs/2310.05035, 2023. URL https://api.semanticscholar.org/CorpusID:263831152.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. J. Pegasus: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020a.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M. T., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068, 2022. URL https://api.semanticscholar.org/CorpusID:248496292.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 649–657. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf.

Zhang, Y., Wei, W., Huang, B., Carley, K. M., and Zhang, Y. Rate: Overcoming noise and sparsity of textual features in real-time location estimation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 2423–2426, 2017.

Zhang, Y., Meng, Y., Huang, J., Xu, F. F., Wang, X., and Han, J. Minimally supervised categorization of text with metadata. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pp. 1231–1240, New York, NY, USA, 2020b. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401168. URL https://doi.org/10.1145/3397271.3401168.

Zhong, Z. and Ng, H. T. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 system demonstrations*, pp. 78–83, 2010.

Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., YU, L., Zhang, S., Ghosh, G., Lewis, M., Zettlemoyer, L., and Levy, O. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=KBMOKmX2he.