UNIVERSITY OF CALIFORNIA SAN DIEGO

Towards Data-efficient Machine Learning Systems

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

in

Computer Science

by

Noveen Sachdeva

Committee in charge:

Professor Julian McAuley, Chair
Professor Zhiting Hu
Professor Jingbo Shang
Professor Lily Weng

2024

The Dissertation of Noveen Sachdeva is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

DEDICATION

To the hopefully unpredictable ***entropy*** in this world, universe, and beyond.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would first and foremost like to acknowledge Professor Julian McAuley for his support as the chair of my committee and as my earnest advisor. In addition to your invaluable research and career advice, I greatly appreciate the freedom handed to me for pursuing research interests of my choice, thorough comments on various overleaf documents which made me learn a great deal about formal writing, and am still amazed by how you are always able to carve out time to meet with all of us members of the McAuley lab, on such short notice.

I am also grateful to all of the esteemed and highly knowledgeable members of my committee, Professors Jingbo Shang, Lily Weng, and Zhiting Hu for providing constructive feedback on this dissertation.

Next, I am greatly indebted to my family for their continual support in pursuing my dream. A special thanks to my fiancé, Saanya, for her never-ending love for me, supporting and helping me in each and every step on this journey. She has been a constant source of inspiration and positivity for me. A special thanks to my parents, Neena and Umesh, for giving me the perfect upbringing, education, and moral values since childhood, oftentimes at the expense of innumerable sacrifices of their own. This dissertation is a culmination of all of our combined efforts through the years.

I have utmost gratitude for all of my wonderful and extremely talented collaborators: Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Lichan Hong, Ed Chi, James Caverlee, Derek Zhiyuan Cheng (Google DeepMind); Lequn Wang, Dawen Liang, Nathan Kallus (Netflix); Ziran Wang, Kyungtae Han, Rohit Gupta (Toyota); Carole-Jean Wu (Meta AI); Mehak Preet Dhaliwal (UC Santa Barbara).

I'm grateful for being a part of the McAuley lab and the overall community at UC San Diego, due to which I've had the opportunity to make many lifelong friendships: Canwen Xu, Zhankui He, Zachary Novack, Petros Karypis, An Yan, Hao-Wen Dong, Chengyu Dong, Alex Trevithick, Sean O'Brien, Ahmad Bin Rabiah, Yupeng Hou, Junda Wu, Zhouhang Xie, Hyunsik Jeon, Nafis Sadeq, Jiacheng Li, Yu Wang, Se-eun Yoon, Jessica Echterhoff, Zexue He, Mengting

Wan, Bodhisattwa Prasad Majumder, Shuyang Li, Paarth Neekhara, and many more. Last but not the least, I'm grateful for being a part of *"Something Cool"* — a group of friends I like to call family. They have been an incredible support group, and a set of people I can rely 100% on.

Finally, I would like to thank all of my co-authors who kindly approved the following publications to be included in my dissertation:

Chapter 1, in part, is a reprint of the material as it appears in "On Sampling Collaborative Filtering Datasets." by Noveen Sachdeva, Carole-Jean Wu, and Julian McAuley, in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining. 2022.* The dissertation author was the primary investigator and author of this paper.

Chapter 2, in part, is currently being prepared for submission for publication of the material "How to Train Data-Efficient LLMs." by Noveen Sachdeva, Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Lichan Hong, Ed Chi, James Caverlee, Julian McAuley, and Derek Zhiyuan Cheng. 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, is a reprint of the material as it appears in "Infinite Recommendation Networks: A Data-centric Approach." by Noveen Sachdeva, Mehak Preet Dhaliwal, Carole-Jean Wu, and Julian McAuley, in *Advances in Neural Information Processing Systems. 2022.* The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is currently being prepared for submission for publication of the material "Farzi Data: Autoregressive Data Distillation." by Noveen Sachdeva, Zexue He, Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Derek Zhiyuan Cheng, and Julian McAuley. 2024. The dissertation author was the primary investigator and author of this paper.

VITA

2015-2020 B.Tech., IIIT Hyderabad

2015-2020 M.S., IIIT Hyderabad

2020–2024 Ph.D., University of California San Diego

PUBLICATIONS

"Off-Policy Evaluation for Large Action Spaces via Policy Convolution", The Web Conference, 2024

"Data Distillation: A Survey", Transactions on Machine Learning Research, 2023

"Infinite Recommendation Networks: A Data-Centric Approach", 36th Conference on Neural Information Processing Systems, 2022

"Fast Autoregressive Transformers meet RNNs for Personalized Adaptive Cruise Control", 25th IEEE International Conference on Intelligent Transportation Systems, 2022

"On Sampling Collaborative Filtering Datasets", 15th ACM International Conference on Web Search & Data Mining, 2022

"ECLARE: Extreme Classification with Label Graph Correlations", The Web Conference, 2021

"Off-policy Bandits with Deficient Support", 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (Research Track), 2020

"How Useful are Reviews for Recommendation? A Critical Review & Potential Improvements", 43rd International ACM Conference on Research and Development in Information Retrieval, 2020

"Sequential Variational Autoencoders for Collaborative Filtering", 12th ACM International Conference on Web Search & Data Mining, 2019

"Attentive Neural Architecture Incorporating Song Features For Music Recommendation", 12th ACM International Conference on Recommender Systems, 2018

"Explicit Modelling of Implicit Short Term User Preferences for Music Recommendation", 40th European Conference on Information Retrieval, 2018

ABSTRACT OF THE DISSERTATION

Towards Data-efficient Machine Learning Systems

by

Noveen Sachdeva

Doctor of Philosophy in Computer Science

University of California San Diego, 2024

Professor Julian McAuley, Chair

The amount of data available to train modern machine learning systems has been increasing rapidly, so much so that we're using, *e.g.*, entirety of the publicly available text data to train state-of-the-art (SoTA) large language models (LLMs), interaction data from billions of users to train SoTA recommender systems, *etc.* Training of such large machine learning systems on such large datasets entails a high (i) computational runtime, (ii) economical cost, and (iii) carbon footprint; all of which we aim to minimize for different reasons.

While a large body of literature develops *"model-centric"* techniques to better model a given dataset, in this thesis, we develop a *"data-centric"* viewpoint, where we are interested in techniques that can appropriately *summarize* a given training dataset, such that models can be

trained equally effectively on the data summary *vs.* training on the much larger original dataset. In addition to being more efficient overall, data-efficient techniques further aim to *improve* the trained model's quality by stripping away the low-quality and noisy sources of information in the original dataset.

More specifically, we develop techniques from two disparate data summarization ideologies: (i) data pruning (*a.k.a.* coreset construction) techniques that *sample* the most relevant portions from the dataset using various grounded heuristics, and (ii) data distillation techniques that *generate* synthetic data-points which summarize the underlying information in the dataset, and are optimized end-to-end using meta-learning. We restrict our scope to training (i) language models on textual datasets, and (ii) recommender systems on user-item interaction datasets.

By pushing the frontier of data-efficient training of machine learning systems, we believe our research can effectively contribute to the practical success of such widely-deployed systems, as well as provide a better understanding for the research community to build future work on.

Introduction

Motivation

While the development of optimization frameworks such as gradient descent (Ruder, 2016) for optimizing large parametric function-approximator models such as deep neural networks (LeCun et al., 2015) has seen huge success in practical applications; the next frontier of AI innovation, arguably, lies in improving the sample- or data-efficiency of the underlying *"training"* procedure of such machine learning systems.

This claim of a surprising lack of data-efficiency of modern-day machine learning systems is grounded in reality by a simple comparison between the amount of data seen by an average x year old human and their estimated intelligence *vs.* the amount of data seen by the largest machine learning system ever trained and its estimated intelligence. We routinely find humans (or other animals for that matter) to be orders of magnitude more data-efficient at the fundamental process of *"learning" vs.* state-of-the-art machine learning systems (Lake et al., 2015; Chevalier-Boisvert et al., 2018).

While naïvely scaling up artifacts like (i) the model size, or (ii) total amount of training data might seem to resolve (or at least severely decrease) the data-efficiency issue; there is still a fundamental ceiling to increasing all of these aforementioned quantities. For example, the largest language models have already crossed more than (i) a trillion total parameters (OpenAI, 2023), and are (ii) routinely trained on trillions of tokens of text (Touvron et al., 2023b; Team et al., 2024).

In this dissertation, we are interested in improving the data-efficiency of modern machine

learning systems by creating high-fidelity data summaries of the original dataset, that are much smaller and of higher quality compared to the original dataset. Consequently, training the same model now on the curated data summary, we aim to achieve (i) faster convergence rates due to a smaller dataset, and (ii) better model generalization achieved by removing redundant and noisy sources from the dataset.

Dissertation Organization

Limiting the scope of this dissertation to training (i) recommender systems on user-item interaction datasets, and (ii) language models on textual datasets, we develop techniques from two disparate data summarization ideologies: (i) data pruning (*a.k.a.* coreset construction) techniques that *sample* the most relevant portions from the dataset using various grounded heuristics, and (ii) data distillation techniques that *generate* synthetic data-points which summarize the underlying information in the dataset, and are optimized end-to-end using meta-learning.

First, in Chapter 1, we develop SVP-CF, a data curation strategy for collaborative filtering (recommender systems) data, that aims to preserve the relative performance of models when trained on sampled data, and is especially suited to long-tailed interaction datasets. Characterizing the effect of sampling on downstream algorithm performance for a wide variety of data curation techniques, we develop an *oracle* model, DATA-GENIE, which can suggest the sampling scheme that is most likely to preserve model performance for a given dataset.

Next, in Chapter 2, we develop data-efficient approaches for pre-training large language models. We seek to understand the tradeoffs associated with data curation routines based on (i) expensive-to-compute *data-quality* estimates, and (ii) maximization of *coverage* and diversity-based measures in the feature space. To this end, we develop two data curation strategies, ASK-LLM and DENSITY. Our first technique, ASK-LLM, leverages the zero-shot reasoning capabilities of instruction tuned LLMs to directly assess the quality of a training example. On the other hand, the DENSITY sampler models the dataset distribution to select a diverse sample,

maximizing coverage.

Next, in Chapter 3, we leverage the Neural Tangent Kernel and its equivalence to training infinitely-wide neural networks to devise ∞-AE: an autoencoder with infinitely-wide bottleneck layers. The outcome is a highly expressive yet simplistic recommendation model with a single hyper-parameter and a closed-form solution. Leveraging ∞-AE's simplicity, we further develop DISTILL-CF, a data distillation technique for synthesizing terse and high-fidelity data summaries which distill the most important knowledge from the extremely large and sparse user-item interaction matrix for efficient and accurate subsequent data-usage like model training, inference, architecture search, *etc.*

Finally, in Chapter 4, we study data distillation for auto-regressive machine learning tasks, where the input and output have a strict left-to-right causal structure. More specifically, we propose FARZI, which summarizes an event sequence dataset into a small number of *synthetic* sequences — FARZI DATA — which are optimized to maintain (if not improve) model performance compared to training on the full dataset.

Chapter 1

Dataset Pruning for Recommender Systems

We study the practical consequences of dataset sampling strategies on the ranking performance of recommendation algorithms. Recommender systems are generally trained and evaluated on *samples* of larger datasets. Samples are often taken in a naïve or ad-hoc fashion, *e.g.*, by sampling a dataset randomly or by selecting users or items with many interactions. As we demonstrate, commonly-used data sampling schemes can have significant consequences on algorithm performance. Following this observation, this chapter makes three main contributions:

- *Characterizing* the effect of sampling on algorithm performance, in terms of algorithm and dataset characteristics (*e.g.* sparsity characteristics, sequential dynamics, *etc.*)
- Designing SVP-CF, which is a data-specific sampling strategy, that aims to preserve the relative performance of models after sampling, and is especially suited to long-tailed interaction data.
- Developing an *oracle*, DATA-GENIE, which can suggest the sampling scheme that is most likely to preserve model performance for a given dataset.

The main benefit of DATA-GENIE is that it will allow recommender system practitioners to quickly prototype and compare various approaches, while remaining confident that algorithm performance will be preserved, once the algorithm is retrained and deployed on the complete

data. Detailed experiments show that using DATA-GENIE, we can discard upto 5× more data than any sampling strategy with the same level of performance.

1.1 Introduction

Representative *sampling* of collaborative filtering (CF) data is a crucial problem from numerous stand-points and can be performed at various levels: (1) mining hard-negatives while training complex algorithms over massive datasets (Mittal et al., 2021; Chen et al., 2017); (2) down-sampling the item-space to estimate expensive ranking metrics (Krichene and Rendle, 2020; Cañamares and Castells, 2020); and (3) reasons like easy-sharing, fast-experimentation, mitigating the significant environmental footprint of training resource-hungry machine learning models (Patterson et al., 2021; Wu et al., 2022; Gupta et al., 2021; Schwartz et al., 2019). In this chapter, we are interested in finding a sub-sample of a dataset which has minimal effects on model utility evaluation *i.e.* an algorithm performing well on the sub-sample should also perform well on the original dataset.

Preserving *exactly* the same levels of performance on sub-sampled data over metrics, such as MSE and AUC, is a challenging problem. A simpler yet useful problem is accurately preserving the *ranking* or relative performance of different algorithms on sub-sampled data. For example, a sampling scheme that has low bias but high variance in preserving metric performance values has less utility than a different sampling scheme with high amounts of bias but low variance, since the overall algorithm ranking is still preserved.

Performing ad-hoc sampling such as randomly removing interactions, or making dense subsets by removing users *or* items with few interactions (Sachdeva and McAuley, 2020) can have adverse downstream repercussions. For example, sampling only the head-portion of a dataset—from a fairness and inclusion perspective—is inherently biased against minority-groups and benchmarking algorithms on this biased data is likely to propagate the original sampling bias. Alternatively, simply from a model performance view-point, accurately retaining the

relative performance of different recommendation algorithms on much smaller sub-samples is a challenging research problem in itself.

Two prominent directions toward better and more representative sampling of CF data are: (1) designing principled sampling strategies, especially for user-item interaction data; and (2) analyzing the performance of different sampling strategies, in order to better grasp which sampling scheme performs "better" for which types of data. *In this chapter,* we explore both directions through the lens of expediting the recommendation algorithm development cycle by:

- Characterizing the efficacy of *sixteen* different sampling strategies in accurately benchmarking various kinds of recommendation algorithms on smaller sub-samples.
- Proposing a sampling strategy, SVP-CF, which dynamically samples the "toughest" portion of a CF dataset. SVP-CF is tailor-designed to handle the inherent data heterogeneity and missing-not-at-random properties in user-item interaction data.
- Developing DATA-GENIE, which analyzes the *performance* of different sampling strategies. Given a dataset sub-sample, DATA-GENIE can directly estimate the likelihood of model performance being preserved on that sample.

Ultimately, our experiments reveal that SVP-CF outperforms all other sampling strategies and can accurately benchmark recommendation algorithms with roughly $50-60\%$ of the original dataset size. Furthermore, by employing DATA-GENIE to dynamically select the best sampling scheme for a dataset, we are able to preserve model performance with only 10% of the initial data, leading to a net $5.8\times$ training time speedup.

1.2 Related Work

Sampling CF data. Sampling in CF-data has been a popular choice for three major scenarios. Most prominently, sampling is used for mining hard-negatives while training recommendation algorithms. Some popular approaches include random sampling; using the graph-structure (Ying

et al., 2018; Mittal et al., 2021); and ad-hoc techniques like similarity search (Jain et al., 2019), stratified sampling (Chen et al., 2017), *etc.* Sampling is also generally employed for evaluating recommendation algorithms by estimating expensive to compute metrics like Recall, nDCG, *etc.* (Krichene and Rendle, 2020; Cañamares and Castells, 2020). Finally, sampling is also used to create smaller sub-samples of a big dataset for reasons like fast experimentation, benchmarking different algorithms, privacy concerns, *etc.* However, the consequences of different samplers on any of these downstream applications is under-studied, and is the main research interest of this chapter.

Coreset selection. Closest to our work, a coreset is loosely defined as a subset of data-points that maintains a similar "quality" as the full dataset for subsequent model training. Submodular approaches try to optimize a function $f : \mathbf{V} \mapsto \mathcal{R}_+$ which measures the utility of a subset $\mathbf{V} \subseteq \mathbf{X}$, and use it as a proxy to select the best coreset (Kaushal et al., 2019). More recent works treat coreset selection as a bi-level optimization problem (Borsos et al., 2020b; Krause et al., 2021) and directly optimize for the best coreset for a given downstream task. Selection-via-proxy (Coleman et al., 2020) is another technique which employs a base-model as a proxy to tag the importance of each data-point. Note, however, that most existing coreset selection approaches were designed primarily for classification data, whereas adapting them for CF-data is non-trivial because of: (1) the inherent data heterogeneity; the (2) wide range of recommendation metrics; and (3) the prevalent missing-data characteristics.

Evaluating sample quality. The quality of a dataset sample, if estimated correctly is of high interest for various applications. Short of being able to evaluate the "true" utility of a sample, one generally resorts to either retaining task-dependent characteristics (Morstatter et al., 2013) *or* employing universal, handcrafted features like a social network's hop distribution, eigenvalue distribution, *etc.* (Leskovec and Faloutsos, 2006) as meaningful proxies. Note that evaluating the sample quality with a limited set of handcrafted features might introduce bias in the sampled data, depending on the number and quality of such features.

1.3 Sampling Collaborative Filtering Datasets

Given our motivation of quickly benchmarking recommendation algorithms, we now aim to *characterize* the performance of various commonly-used sampling strategies. We loosely define the performance of a sampling scheme as its ability in effectively retaining the performance-ranking of different recommendation algorithms on the full *vs.* sub-sampled data. In this section, we start by discussing the different recommendation feedback scenarios we consider, along with a representative sample of popular recommendation algorithms that we aim to efficiently benchmark. We then examine popular data sampling strategies, followed by proposing a novel, proxy-based sampling strategy (SVP-CF) that is especially suited for sampling representative subsets from long-tail CF data.

1.3.1 Problem Settings & Methods Compared

To give a representative sample of typical recommendation scenarios, we consider three different user feedback settings. In *explicit feedback*, each user u gives a numerical rating r_i^u to each interacted item i; the model must predict these ratings for novel (test) user-item interactions. Models from this class are evaluated in terms of the Mean Squared Error (MSE) of the predicted ratings. Another scenario we consider is *implicit feedback*, where the interactions for each user are only available for positive items (*e.g.* clicks or purchases), whilst all non-interacted items are considered as negatives. We employ the AUC, Recall@100, and nDCG@10 metrics to evaluate model performance for implicit feedback algorithms. Finally, we also consider *sequential feedback*, where each user u interacts with an ordered sequence of items $\mathcal{S}^u = (\mathcal{S}_1^u, \mathcal{S}_2^u, \ldots, \mathcal{S}_{|\mathcal{S}^u|}^u)$ such that $\mathcal{S}_i^u \in \mathcal{I}$ for all $i \in \{1, \ldots, |\mathcal{S}^u|\}$. Given $\mathcal{S} = \{\mathcal{S}^u \mid \forall u \in \mathcal{U}\}$, the goal is to identify the *next-item* for each sequence $\mathcal{S}^u$ that each user u is most likely to interact with. We use the same metrics as in implicit feedback settings. Note that following recent warnings against sampled metrics for evaluating recommendation algorithms (Krichene and Rendle, 2020; Cañamares and Castells, 2020), we compute both Recall and nDCG by ranking *all* items in the dataset. Further

specifics about the datasets used, pre-processing, train/test splits, *etc.* are discussed in-depth in Section 1.3.5.

Given the diversity of the scenarios discussed above, there are numerous relevant recommendation algorithms. We use the following seven recommendation algorithms, intended to represent the state-of-the-art and standard baselines:

- *PopRec:* A naïve baseline that simply ranks items according to overall train-set popularity. Note that this method is unaffected by the user for which items are being recommended, and has the *same global ranking* of all items.

- *Bias-only:* Another simple baseline that assumes no interactions between users and items. Formally, it learns: (1) a global bias α; (2) scalar biases β_u for each user $u \in \mathcal{U}$; and (3) scalar biases β_i for each item $i \in \mathcal{I}$. Ultimately, the rating/relevance for user u and item i is modeled as $\hat{r}_i^u = \alpha + \beta_u + \beta_i$.

- *Matrix Factorization (MF) (Koren et al., 2009):* Represents both users and items in a common, low-dimensional latent-space by factorizing the user-item interaction matrix. Formally, the rating/relevance for user u and item i is modeled as $\hat{r}_i^u = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$ where $\gamma_u, \gamma_i \in \mathbb{R}^d$ are learned latent representations.

- *Neural Matrix Factorization (NeuMF) (He et al., 2017):* Leverages the representation power of deep neural-networks to capture non-linear correlations between user and item embeddings. Formally, the rating/relevance for user u and item i is modeled as $\hat{r}_i^u = \alpha + \beta_u + \beta_i + f(\gamma_u \,||\, \gamma_i \,||\, \gamma_u \cdot \gamma_i)$ where $\gamma_u, \gamma_i \in \mathbb{R}^d$, '||' represents the concatenation operation, and $f : \mathbb{R}^{3d} \mapsto \mathbb{R}$ represents an arbitrarily complex neural network.

- *Variational Auto-Encoders for Collaborative Filtering (MVAE) (Liang et al., 2018):* Builds upon the Variational Auto-Encoder (VAE) (Kingma and Welling, 2014) framework to learn a low-dimensional representation of a user's consumption history. More specifically, MVAE

encodes each user's bag-of-words consumption history using a VAE and further decodes the latent representation to obtain the completed user preference over all items.

- *Sequential Variational Auto-Encoders for Collaborative Filtering (SVAE) (Sachdeva et al., 2019):* A sequential algorithm that combines the temporal modeling capabilities of a GRU (Chung et al., 2014) along with the representation power of VAEs. Unlike MVAE, SVAE uses a GRU to encode the user's consumption sequence followed by a multinomial VAE at each time-step to model the likelihood of the next item.

- *Self-attentive Sequential Recommendation (SASRec) (Kang and McAuley, 2018):* Another sequential algorithm that relies on the sequence modeling capabilities of self-attentive neural networks (Vaswani et al., 2017) to predict the occurance of the next item in a user's consumption sequence. To be precise, given a user u and their time-ordered consumption history $\mathcal{S}^u = (\mathcal{S}^u_1, \mathcal{S}^u_2, \ldots, \mathcal{S}^u_{|\mathcal{S}^u|})$, SASRec first applies self-attention on $\mathcal{S}^u$ followed by a series of non-linear feed-forward layers to finally obtain the next item likelihood.

We also list models and metrics for each of the three different CF-scenarios in Table 1.1. Since bias-only, MF, and NeuMF can be trained for all three CF-scenarios, we optimize them using the regularized least-squares regression loss for explicit feedback, and the pairwise-ranking (BPR (Rendle et al., 2009)) loss for implicit/sequential feedback. Note however that the aforementioned algorithms are only intended to be a representative sample of a wide pool of recommendation algorithms, and in our pursuit to benchmark recommender systems faster, we are primarily concerned with the *ranking* of different algorithms on the full dataset *vs.* a smaller sub-sample.

1.3.2 Sampling Strategies

Given a user-item CF dataset $\mathcal{D}$, we aim to create a $p\%$ subset $\mathcal{D}^{s,p}$ according to some sampling strategy s. In this chapter, to be comprehensive, we consider a sample of eight popular sampling strategies, which can be grouped into the following three categories:

Table 1.1. Demonstrates the pertinence of each CF-scenario towards each algorithm (left) and each metric (right). Note that we can use ranking metrics for explicit feedback, however, we only use MSE as a design choice and due to it's direct relevance.

CF-scenario	*Algorithm*							*Metric*			
	Bias-only	MF	NeuMF	PopRec	MVAE	SVAE	SASRec	MSE	AUC	Recall@k	nDCG@k
Explicit	Yes	Yes	Yes	×	×	×	×	Yes	×	×	×
Implicit	Yes	Yes	Yes	Yes	Yes	×	×	×	Yes	Yes	Yes
Sequential	Yes	Yes	Yes	Yes	Yes	Yes	Yes	×	Yes	Yes	Yes

Interaction sampling. We first discuss three strategies that sample interactions from $\mathcal{D}$. In *Random Interaction Sampling*, we generate $\mathcal{D}^{s,p}$ by randomly sampling $p\%$ of all the user-item interactions in $\mathcal{D}$. *User-history Stratified Sampling* is another popular sampling technique (see *e.g.* (Sachdeva et al., 2019; X et al., 2011)) to generate smaller CF-datasets. To match the user-frequency distribution amongst $\mathcal{D}$ and $\mathcal{D}^{s,p}$, it randomly samples $p\%$ of interactions from each user's consumption history. Unlike random stratified sampling, *User-history Temporal Sampling* samples $p\%$ of the *most recent* interactions for each user. This strategy is representative of the popular practice of making data subsets from the online traffic of the last x days (Mittal et al., 2021; Jain et al., 2016).

User sampling. Similar to sampling interactions, we also consider two strategies which sample users in $\mathcal{D}$ instead. To ensure a fair comparison amongst the different kinds of sampling schemes used in this chapter, we retain exactly $p\%$ of the *total interactions* in $\mathcal{D}^{s,p}$. In *Random User Sampling*, we retain users from $\mathcal{D}$ at random. To be more specific, we iteratively preserve *all* the interactions for a random user until we have retained $p\%$ of the original interactions. Another strategy we employ is *Head User Sampling*, in which we iteratively remove the user with the least amount of total interactions. This method is representative of commonly used data pre-processing strategies (see *e.g.* (Liang et al., 2018; He et al., 2017)) to make data suitable for parameter-heavy algorithms. Sampling the data in such a way can introduce bias toward users from minority groups which might raise concerns from a diversity and fairness perspective

(Mitchell et al., 2018).

Graph sampling. Instead of sampling directly from $\mathcal{D}$, we also consider three strategies that sample from the inherent user-item bipartite interaction graph $\mathcal{G}$. In *Centrality-based Sampling*, we proceed by computing the pagerank centrality scores (Page et al., 1999) for each node in $\mathcal{G}$, and retain all the edges (interactions) of the *top scoring nodes* until a total $p\%$ of the original interactions have been preserved. Another popular strategy we employ is *Random-walk Sampling* (Leskovec and Faloutsos, 2006), which performs multiple random-walks with restart on $\mathcal{G}$ and retains the edges amongst those pairs of nodes that have been visited at least once. We keep expanding our walk until $p\%$ of the initial edges have been retained. We also utilize *Forest-fire Sampling* (Leskovec et al., 2005), which is a snowball sampling method and proceeds by randomly "burning" the outgoing edges of visited nodes. It initially starts with a random node, and then propagates to a random subset of previously unvisited neighbors. The propagation is terminated once we have created a graph-subset with $p\%$ of the initial edges.

1.3.3 SVP-CF: Selection-Via-Proxy for CF data

Selection-Via-Proxy (SVP) (Coleman et al., 2020) is a leading coreset mining technique for classification datasets like CIFAR10 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009). The main idea proposed is simple and effective, and proceeds by training a relatively inexpensive base-model as a proxy to define the "importance" of a data-point. However, applying SVP to CF-data can be highly non-trivial because of the following impediments:

- *Data heterogeneity:* Unlike classification data over some input space $\mathcal{X}$ and label-space $\mathcal{Y}$, CF-data consists of numerous four-tuples $\{u, i, r_i^u, t_i^u\}$. Such multimodal data adds many different dimensions to sample data from, making it increasingly complex to define meaningful samplers.

- *Defining the importance of a data point:* Unlike classification, where we can measure the performance of a classifier by its empirical risk on held-out data, for recommendation,

there are a variety of different scenarios (Section 1.3.1) along with a wide list of relevant evaluation metrics. Hence, it becomes challenging to adapt importance-tagging techniques like greedy k-centers (Sener and Savarese, 2018), forgetting-events (Toneva et al., 2019), *etc.* for recommendation tasks.

- *Missing data:* CF-data is well-known for (1) its sparsity; (2) skewed and long-tail user/item distributions; and (3) missing-not-at-random (MNAR) properties of the user-item interaction matrix. This results in additional problems as we are now sampling data from skewed, MNAR data, especially using proxy-models trained on the same skewed data. Such sampling in the worst-case might even lead to exacerbating existing biases in the data or even aberrant data samples.

To address these fundamental limitations in applying the SVP philosophy to CF-data, we propose SVP-CF to sample representative subsets from large user-item interaction data. SVP-CF is also specifically devised for our objective of benchmarking different recommendation algorithms, as it relies on the crucial assumption that the "easiest" part of a dataset will generally be easy *for all* algorithms. Under this assumption, even after removing such data we are still likely to retain the overall algorithms' ranking.

Because of the inherent data heterogeneity in user-item interaction data, we can sub-sample in a variety of different ways. We design SVP-CF to be versatile in this aspect as it can be applied to sample users, items, interactions, or combinations of them, by marginally adjusting the definition of importance of each data-point. In this chapter, we limit the discussion to only sampling users and interactions (separately), but extending SVP-CF for sampling across other data modalities should be relatively straightforward.

Irrespective of whether to sample users or interactions, SVP-CF proceeds by training an inexpensive proxy model $\mathcal{P}$ on the full, original data $\mathcal{D}$ and modifies the forgetting-events approach (Toneva et al., 2019) to retain the points with the *highest* importance. To be more specific, for explicit feedback, we define the importance of each data-point *i.e.* $\{u, i, r_i^u, t_i^u\}$

interaction as $\mathcal{P}$'s average MSE (over epochs) of the specific interaction if we're sampling interactions *or* $\mathcal{P}$'s average MSE of u (over epochs) if we're sampling users. Whereas, for implicit and sequential feedback, we use $\mathcal{P}$'s average inverse-AUC while computing the importance of each data-point. For the sake of completeness, we experiment with both Bias-only and MF as two different kinds of proxy-models for SVP-CF. Since both models can be trained for all three CF-scenarios (Table 1.1), we can directly use them to tag the importance for each CF-scenario.

Ultimately, to handle the MNAR and long-tail problems, we also propose SVP-CF-PROP which employs user and item propensities to correct the distribution mismatch while estimating the importance of each datapoint. More specifically, let $p_{u,i} = P(r_i^u = 1 \mid \overset{*}{r}_i^u = 1)$ denote the probability of user u and item i's interaction actually being observed (propensity), E be the total number of epochs that $\mathcal{P}$ was trained for, $\mathcal{P}_e$ denote the proxy model after the e^{th} epoch, $\mathcal{I}_u^+ \coloneqq \{j \mid r_j^u > 0\}$ be the set of positive interactions for u, and $\mathcal{I}_u^- \coloneqq \{j \mid r_j^u = 0\}$ be the set of negative interactions for u; then, the importance function for SVP-CF-PROP, $\mathcal{I}_p$ is defined as follows:

$$\mathcal{I}_p(u \mid \mathcal{P}) \coloneqq \frac{1}{|\mathcal{I}_u^+|} \cdot \sum_{i \in \mathcal{I}_u^+} \mathcal{I}_p(u,i \mid \mathcal{P}) \quad ; \quad \mathcal{I}_p(u,i \mid \mathcal{P}) \coloneqq \frac{\Delta(u,i \mid \mathcal{P})}{p_{u,i}}$$

$$\text{where,} \qquad \Delta(u,i \mid \mathcal{P}) \coloneqq \begin{cases} \sum_{e=1}^{E} \left(\mathcal{P}_e(u,i) - r_i^u\right)^2 \\ \text{(for explicit feedback)} \\ \\ \sum_{e=1}^{E} \sum_{j \sim \mathcal{I}_u^-} \frac{1}{\mathbb{1}\left(\mathcal{P}_e(u,i) > \mathcal{P}_e(u,j)\right)} \\ \text{(for implicit/sequential feedback)} \end{cases}$$

Proposition 1.3.1. *Given an ideal propensity-model $p_{u,i}$; $\mathcal{I}_p(u,i \mid \mathcal{P})$ is an unbiased estimator*

of $\Delta(u,i \mid \mathcal{P})$.

Proof.

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{u\sim\mathcal{U}}\mathop{\mathbb{E}}_{i\sim\mathcal{I}}[\mathcal{I}_p(u,i \mid \mathcal{P})] \\
&\qquad = \frac{1}{|\mathcal{U}||\mathcal{I}|}\sum_{u\sim\mathcal{U}}\sum_{i\sim\mathcal{I}}\mathcal{I}_p(u,i \mid \mathcal{P})\cdot P(r_i^u=1) \\
&\qquad = \frac{1}{|\mathcal{U}||\mathcal{I}|}\sum_{u\sim\mathcal{U}}\sum_{i\sim\mathcal{I}}\frac{\Delta(u,i \mid \mathcal{P})}{p_{u,i}}\cdot(P(\overset{*}{r}{}_i^u=0)\cdot \cancelto{0}{P(r_i^u=1 \mid \overset{*}{r}{}_i^u=0)} \\
&\qquad\qquad + P(\overset{*}{r}{}_i^u=1)\cdot P(r_i^u=1 \mid \overset{*}{r}{}_i^u=1)) \\
&\qquad = \frac{1}{|\mathcal{U}||\mathcal{I}|}\sum_{u\sim\mathcal{U}}\sum_{i\sim\mathcal{I}}\Delta(u,i \mid \mathcal{P})\cdot P(\overset{*}{r}{}_i^u=1) \\
&\qquad = \mathop{\mathbb{E}}_{u\sim\mathcal{U}}\mathop{\mathbb{E}}_{i\sim\mathcal{I}}[\Delta(u,i \mid \mathcal{P})]
\end{aligned}
$$

□

Propensity model. A wide variety of ways exist to model the propensity score of a user-item interaction (Zhu et al., 2020; Schnabel et al., 2016; Sachdeva et al., 2020; Jain et al., 2016). The most common ways comprise using machine learning models like naïve bayes and logistic regression (Schnabel et al., 2016), or by fitting handcrafted functions (Jain et al., 2016). For our problem statement, we make a simplifying assumption that the data noise is one-sided *i.e.* $P(r_i^u=1 \mid \overset{*}{r}{}_i^u=0)$ or the probability of a user interacting with a *wrong* item is *zero*, and model the probability of an interaction going missing to decompose over the user and item as follows:

$$
\begin{aligned}
p_{u,i} &= P(r_i^u=1 \mid \overset{*}{r}{}_i^u=1) \\
&= P(r^u=1 \mid \overset{*}{r}{}^u=1)\cdot P(r_i=1 \mid \overset{*}{r}_i=1) \;=\; p_u\cdot p_i
\end{aligned}
$$

Ultimately, following (Jain et al., 2016), we assume the user and item propensities to lie on the following sigmoid curves:

$$
p_u \coloneqq \frac{1}{1+C_u\cdot e^{-A\cdot log(N_u+B)}} \quad ; \quad p_i \coloneqq \frac{1}{1+C_i\cdot e^{-A\cdot log(N_i+B)}}
$$

Where, N_u and N_i represent the total number of interactions of user u and item i respectively, A and B are two fixed scalars, $C_u = (log(|\mathcal{U}|) - 1) \cdot (B+1)^A$ and $C_i = (log(|\mathcal{I}|) - 1) \cdot (B+1)^A$.

1.3.4 Performance of a sampling strategy

To quantify the performance of a sampling strategy s on a dataset $\mathcal{D}$, we start by creating various $p\%$ subsets of $\mathcal{D}$ according to s and call them $\mathcal{D}^{s,p}$. Next, we train and evaluate all the relevant recommendation algorithms on both $\mathcal{D}$ and $\mathcal{D}^{s,p}$. Let the *ranking* of all algorithms according to CF-scenario f and metric m trained on $\mathcal{D}$ and $\mathcal{D}^{s,p}$ be $\mathcal{R}_{f,m}$ and $\mathcal{R}^{s,p}_{f,m}$ respectively, then the performance measure $\Psi(\mathcal{D}, s)$ is defined as the average correlation between $\mathcal{R}_{f,m}$ and $\mathcal{R}^{s,p}_{f,m}$ measured through Kendall's Tau over all possible CF-scenarios, metrics, and sampling percents:

$$\Psi(\mathcal{D}, s) = \lambda \cdot \sum_{f} \sum_{m} \sum_{p} \tau\left(\mathcal{R}_{f,m}, \mathcal{R}^{s,p}_{f,m}\right)$$

Where λ is an appropriate normalizing constant for computing the average, sampling percent $p \in \{80, 60, 40, 20, 10, 1\}$, CF-scenario f, metric m and their pertinence towards each other can all be found in Table 1.1. Ψ has the same range as Kendall's Tau *i.e.* $[-1, 1]$ and a higher Ψ indicates strong agreement between the algorithm ranking on the full and sub-sampled datasets, whereas a large negative Ψ implies that the algorithm order was effectively reversed.

1.3.5 Experiments

Datasets. To promote dataset diversity in our experiments, we use six public user-item rating interaction datasets with varying sizes, sparsity patterns, and other characteristics. We use the Magazine, Luxury, and Video-games categories of the Amazon review dataset (Ni et al., 2019b), along with the Movielens-100k (Harper and Konstan, 2015), BeerAdvocate (McAuley et al., 2012), and GoodReads Comics (Wan and McAuley, 2018) datasets. A brief set of data statistics is also presented in Table 1.2. We simulate all three CF-scenarios (Section 1.3.1) via

Table 1.2. Data statistics of the *six* datasets used in our experiments.

Dataset	# Interactions	# Users	# Items	Avg. User history length
Amazon Magazine	12.7k	3.1k	1.3k	4.1
ML-100k	100k	943	1.7k	106.04
Amazon Luxury	126k	29.7k	8.4k	4.26
Amazon Video-games	973k	181k	55.3k	5.37
BeerAdvocate	1.51M	18.6k	64.3k	81.45
Goodreads Comics	4.37M	133k	89k	32.72

different pre-processing strategies. For explicit and implicit feedback, we follow a randomized 80/10/10 train-test-validation split for each user's consumption history in the dataset, and make use of the leave-one-last (Meng et al., 2020) strategy for sequential feedback *i.e.* keep the last two interactions in each user's time-sorted consumption history in the validation and test-set respectively. Since we can't control the initial construction of datasets, and to minimize the initial data bias, we follow the least restrictive data pre-processing (Dacrema et al., 2019; Sachdeva and McAuley, 2020). We only weed out the users with less than 3 interactions, to keep at least one occurrence in the train, validation, and test sets.

Training details. We implement all algorithms in PyTorch[1] and train on a single GPU. For a fair comparison across algorithms, we perform hyper-parameter search on the validation set. For the three smallest datasets used in this chapter (Table 1.2), we search the latent size in $\{4, 8, 16, 32, 50\}$, dropout in $\{0.0, 0.3, 0.5\}$, and the learning rate in $\{0.001, 0.006, 0.02\}$. Whereas for the three largest datasets, we fix the learning rate to be 0.006. Note that despite the limited number of datasets and recommendation algorithms used in this study, given that we need to train all algorithms with hyper-parameter tuning for all CF scenarios, % data sampled according to all different sampling strategies discussed in Section 1.3.2, there are a total of $\sim 400k$ unique models trained, equating to a cumulative train time of over $400k \times \sim 1\text{min} \approx 9$ months.

[1]Code is available at https://github.com/noveens/sampling_cf

Data sampling. To compute the Ψ-values as defined in Section 1.3.4, we construct {80, 60, 40, 20, 10, 1} % samples for each dataset and sampling strategy. To keep comparisons as fair as possible, for all sampling schemes, we only sample on the train set and never touch the validation and test sets.

How do different sampling strategies compare to each other? Results with Ψ-values for all sampling schemes on all datasets are in Table 1.3. Even though there are only six datasets under consideration, there are a few prominent patterns. First, the average Ψ for most sampling schemes is around 0.4, which implies a statistically significant correlation between the ranking of algorithms on the full *vs.* sub-sampled datasets. Next, SVP-CF generally outperforms all commonly used sampling strategies by some margin in retaining the ranking of different recommendation algorithms. Finally, the methods that discard the tail of a dataset (head-user and centrality-based) are the worst performing strategies overall, which supports the recent warnings against dense sampling of data (Sachdeva and McAuley, 2020).

How does the relative performance of algorithms change as a function of sampling rate? In an attempt to better understand the impact of sampling on different recommendation algorithms used in this study (Section 1.3.1), we visualize the probability of a recommendation algorithm moving up in the overall method ranking with data sampling. We estimate the aforementioned probability using Maximum-Likelihood-Estimation (MLE) on the experiments already run in computing $\Psi(\mathcal{D}, s)$. Formally, given a recommendation algorithm r, CF-scenario f, and data sampling percent p:

$$P_{MLE}(r \mid f, p) = \lambda \cdot \sum_{\mathcal{D}} \sum_{s} \sum_{m} 0.5 + \frac{\mathcal{R}_{f,m}(r) - \mathcal{R}^{s,p}_{f,m}(r)}{2 \cdot (n-1)}$$

where λ is an appropriate normalizing constant, and n represents the total number of algorithms. A heatmap visualizing P_{MLE} for all algorithms and CF-scenarios is shown in Figure 1.1. We see that simpler methods like Bias-only and PopRec have the highest probability across data

Table 1.3. Ψ-values for all datasets and sampling strategies. Higher Ψ is better. The best Ψ for every dataset is underlined. The Ψ-values for each sampling scheme *averaged over all datasets* is appended to the right.

	Sampling strategy	*Datasets*						***Average***
		Amazon Magazine	**ML 100k**	**Amazon Luxury**	**Amazon V.games**	**Beer Adv.**	**Goodr. Comics**	
Interaction sampling	Random	0.428	0.551	0.409	0.047	0.455	0.552	0.407
	Stratified	0.27	0.499	0.291	-0.01	0.468	0.538	0.343
	Temporal	0.289	0.569	0.416	-0.02	0.539	0.634	0.405
	SVP-CF *w/* MF	0.418	0.674	0.398	0.326	0.425	0.662	0.484
	SVP-CF *w/* Bias-only	0.38	0.684	0.431	0.348	0.365	0.6	0.468
	SVP-CF-PROP *w/* MF	0.381	0.617	0.313	0.305	0.356	0.608	0.43
	SVP-CF-PROP *w/* Bias-only	0.408	0.617	0.351	0.316	0.437	0.617	0.458
User sampling	Random	0.436	0.622	0.429	0.17	0.344	0.582	0.431
	Head	0.369	0.403	0.315	0.11	-0.04	-0.02	0.19
	SVP-CF *w/* MF	0.468	0.578	0.308	0.13	0.136	0.444	0.344
	SVP-CF *w/* Bias-only	0.49	0.608	0.276	0.124	0.196	0.362	0.343
	SVP-CF-PROP *w/* MF	0.438	0.683	0.307	0.098	0.458	0.592	0.429
	SVP-CF-PROP *w/* Bias-only	0.434	0.751	0.233	0.107	0.506	0.637	0.445
Graph	Centrality	0.307	0.464	0.407	0.063	0.011	0.343	0.266
	Random-walk	0.596	0.5	0.395	0.306	0.137	0.442	0.396
	Forest-fire	0.564	0.493	0.415	0.265	0.099	0.454	0.382

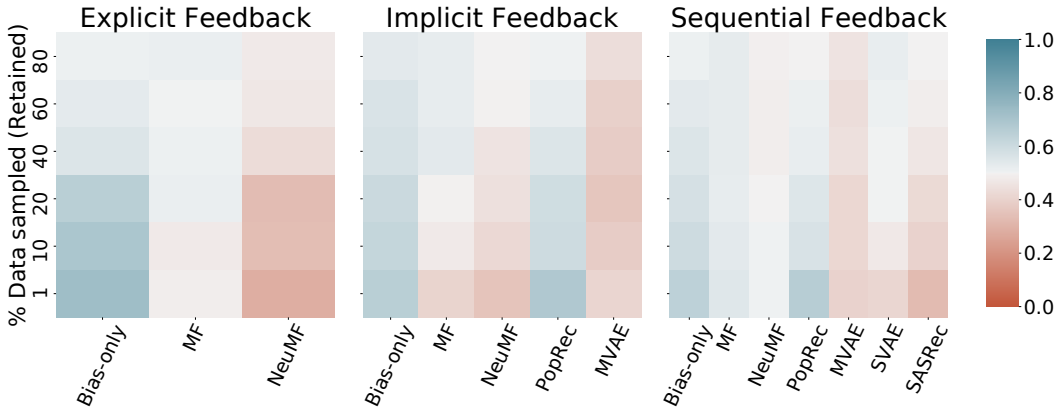

Figure 1.1. Heatmap of the probability of an algorithm moving in the overall ranking with extreme sampling. A high value indicates that the algorithm is most probable to *move up* in the sampled data ranking order, whereas a low value indicates that the algorithm is most probable to *move down*.

scenarios of increasing their ranking order with extreme sampling. Whereas parameter-heavy algorithms like SASRec, SVAE, MVAE, *etc.* tend to decrease in the ranking order, which is indicative of overfitting on smaller data samples.

How much data to sample? Since Ψ is averaged over all $p \in \{80, 60, 40, 20, 10, 1\}\%$ data samples, to better estimate a reasonable amount of data to sample, we stratify Ψ according to each value of p and note the average Kendall's Tau. As we observe from Figure 1.2, there is a steady increase in the performance measure when more data is retained. Next, despite the results in Figure 1.2 being averaged over *sixteen* sampling strategies, we still notice a significant amount of performance retained after sampling just $50-60\%$ of the data.

Are different metrics affected equally by sampling? In an attempt to better understand how the different implicit and sequential feedback metrics (Section 1.3.1) are affected by sampling, we visualize the average Kendall's Tau for all sampling strategies (except SVP-CF for brevity) and all % data sampling choices separately over the AUC, Recall, and nDCG metrics in Figure 1.3. As expected, we observe a steady decrease in the model quality across the accuracy metrics over the different sampling schemes. This is in agreement with the analysis from Figure 1.2.

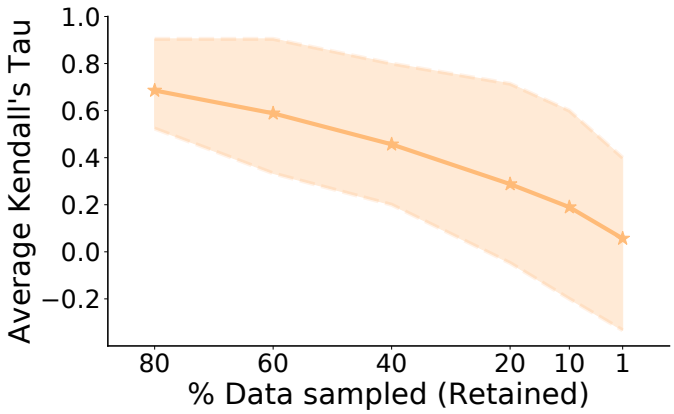

Figure 1.2. Comparison of the average Kendall's Tau with % data sampled. A higher Tau indicates better retaining power of the ranking of different recommendation algorithms.

Next, most sampling schemes follow a *similar* downwards trend in performance for the three metrics with AUC being slightly less affected and nDCG being slightly more affected by extreme sampling.

1.4 DATA-GENIE: Which sampler is best for me?

Although the results presented in Section 1.3.5 are indicative of correlation between the ranking of recommendation algorithms on the full dataset *vs.* smaller sub-samples, there still is no 'one-size-fits-all' solution to the question of *how to best sub-sample a dataset for retaining the performance of different recommendation algorithms?* In this section, we propose DATA-GENIE, that attempts to answer this question from a statistical perspective, in contrast with existing literature that generally has to resort to sensible heuristics (Leskovec and Faloutsos, 2006; Belletti et al., 2019; Chen et al., 2017).

1.4.1 Problem formulation

Given a dataset $\mathcal{D}$, we aim to gauge how a *new* sampling strategy will perform in retaining the performance of different recommendation algorithms. Having already experimented with sixteen different sampling strategies on six datasets (Section 1.3.5), we take a frequentist approach

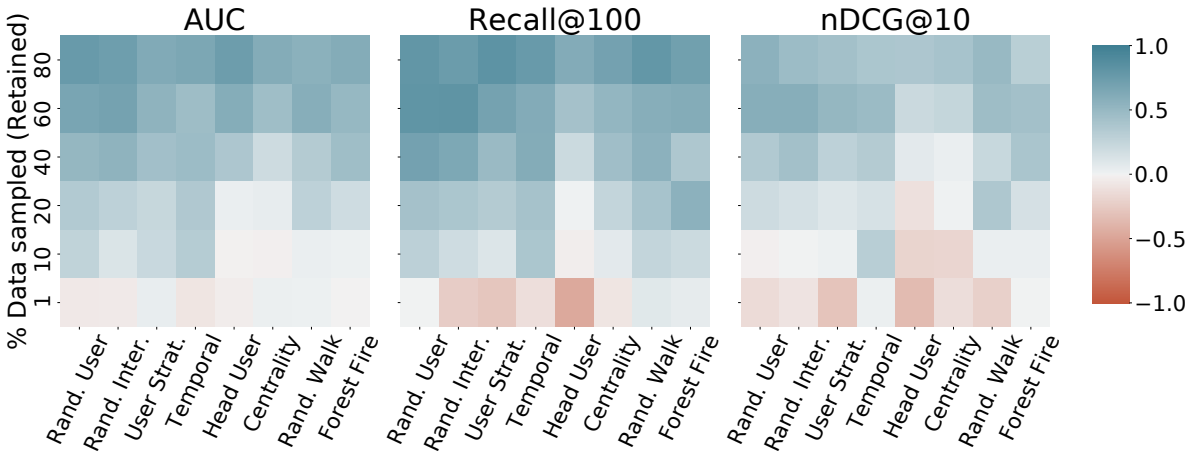

Figure 1.3. Heatmap of the average Kendall's Tau for different samplers stratified over metrics and % data sampled.

in predicting the performance of any sampling scheme. To be precise, to predict the performance of sampling scheme s on dataset $\mathcal{D}$, we start by creating $\mathcal{D}$'s subset according to s and call it $\mathcal{D}^s$. We then represent $\mathcal{D}$ and $\mathcal{D}^s$ in a low-dimensional latent space, followed by a powerful regression model to directly estimate the performance of s on $\mathcal{D}$.

1.4.2 Dataset representation

We experiment with the following techniques of embedding a user-item interaction dataset into lower dimensions:

Handcrafted. For this method, we cherry-pick a few representative characteristics of $\mathcal{D}$ and the underlying user-item bipartite interaction graph $\mathcal{G}$. Inspired by prior work (Leskovec and Faloutsos, 2006), we represent $\mathcal{D}$ as a combination of five features. We first utilize the frequency distribution of all users and items in $\mathcal{D}$. Next, we evaluate the distribution of the top-100 eigenvalues of $\mathcal{G}$'s adjacency matrix. All of these three distributions are generally long-tailed and heavily skewed. Furthermore, to capture notions like the diameter of $\mathcal{G}$, we compare the distribution of the number of hops h *vs.* the number of pairs of nodes in $\mathcal{G}$ reachable at a distance less than h (Palmer et al., 2002). This distribution, unlike others is monotonically

increasing in h. Finally, we also compute the size distribution of all connected components in $\mathcal{G}$, where a connected component is defined to be the maximal set of nodes, such that a path exists between any pair of nodes. Ultimately, we ascertain $\mathcal{D}$'s final representation by concatenating 10 evenly-spaced samples from each of the aforementioned distributions along with the total number of users, items, and interactions in $\mathcal{D}$. This results in a $53-$dimensional embedding for each dataset. Note that unlike previous work of simply *retaining* the discussed features as a proxy of the quality of data subsets (Leskovec and Faloutsos, 2006), DATA-GENIE instead uses these features to learn a regression model *on-top* which can dynamically establish the importance of each feature in the performance of a sampling strategy.

Unsupervised GCN. With the recent advancements in the field of Graph Convolutional Networks (Kipf and Welling, 2017b) to represent graph-structured data for a variety of downstream tasks, we also experiment with a GCN approach to embed $\mathcal{G}$. We modify the InfoGraph framework (Sun et al., 2020a), which uses graph convolution encoders to obtain patch-level representations, followed by sort-pooling (Zhang et al., 2018) to obtain a fixed, low-dimensional embedding for the entire graph. Since the nodes in $\mathcal{G}$ are the union of all users and items in $\mathcal{D}$, we randomly initialize $32-$dimensional embeddings using a Xavier-uniform prior (Glorot and Bengio, 2010). Parameter optimization is performed in an unsupervised fashion by maximizing the mutual information (Nowozin et al., 2016) amongst the graph-level and patch-level representations of nodes in the same graph. We validate the best values of the latent dimension and number of layers of the GCN from $\{4, 8, 16, 32\}$ and $\{1, 3\}$ respectively.

1.4.3 Training & Inference

Having discussed different representation functions $\mathcal{E} : \mathcal{D} \mapsto \mathbb{R}^d$ to embed a CF-dataset in Section 1.4.2, we now discuss DATA-GENIE's training framework agnostic to the actual details about $\mathcal{E}$.

Optimization problem. As a proxy of the performance of a sampler on a given dataset, we re-use the Kendall's Tau for each CF-scenario, metric, and sampling percent used while computing the $\Psi(\mathcal{D}, s)$ in Section 1.3.5. To be specific, given $\mathcal{D}_f^{s,p}$ which is a $p\%$ sample of $f-$type feedback data $\mathcal{D}_f$, sampled according to sampling strategy s, we aim to estimate $\tau(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p})$ without ever computing the actual ranking of algorithms on either the full or sampled datasets:

$$\hat{\tau}\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p}\right) = \Phi\left(\mathcal{E}(\mathcal{D}_f), \mathcal{E}(\mathcal{D}_f^{s,p}), m\right), \tag{1.1}$$

where Φ is an arbitrary neural network, and m is the metric of interest (see Table 1.1). We train Φ by either (1) regressing on the Kendall's Tau computed for each CF scenario, metric, and sampling percent used while computing the $\Psi(\mathcal{D}, s)$ scores in Section 1.3.5; or (2) performing BPR-style (Rendle et al., 2009) pairwise ranking on two sampling schemes $s_i \succ s_j \iff \tau(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s_i,p}) > \tau(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s_j,p})$. Formally, the two optimization problems are defined as follows:

$$\arg\min_{\Phi} \sum_{\mathcal{D}_f} \sum_{s} \sum_{p} \sum_{m} \left(\tau\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p}\right) - \hat{\tau}\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p}\right)\right)^2$$

(DATA-GENIE-regression)

$$\arg\min_{\Phi} \sum_{\mathcal{D}_f} \sum_{s_i \succ s_j} \sum_{p} \sum_{m} -\ln \sigma\left(\hat{\tau}\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s_i,p}\right) - \hat{\tau}\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s_j,p}\right)\right)$$

(DATA-GENIE-ranking)

$$\text{where, } \hat{\tau}\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p}\right) = \Phi\left(\mathcal{E}(\mathcal{D}_f), \mathcal{E}(\mathcal{D}_f^{s,p}), m\right).$$

The critical differences between the two aforementioned optimization problems are the down-stream use-case and Φ's training time. If the utility of DATA-GENIE is to rank different sampling schemes for a given dataset, then DATA-GENIE-ranking is better suited as it is robust to the noise in computing $\tau(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s_i,p})$ like improper hyper-parameter tuning, local minima, *etc.* On the other hand, DATA-GENIE-regression is better suited for the use-case of estimating the exact values of τ for a sampling scheme on a given dataset. Even though both optimization problems

Table 1.4. Results for predicting the best sampling scheme for a particular dataset over a germane metric. The MSE-value next to randomly choosing the sampling scheme represents the variance of the test-set. Best values are underlined.

$\mathcal{E}$	Φ	**MSE**	**P@1**
Random		0.2336	25.2
User sampling *w/* Bias-only SVP-CF-PROP		–	30.6
Handcrafted	Least squares regression	0.1866	31.7
”	XGBoost regression	0.1163	43.9
”	DATA-GENIE-regression	<u>0.1008</u>	51.2
”	DATA-GENIE-ranking	–	51.2
Unsupervised GCN	Least squares regression	0.1838	39.1
”	XGBoost regression	0.1231	43.9
”	DATA-GENIE-regression	0.1293	48.8
”	DATA-GENIE-ranking	–	<u>53.7</u>

converge in less than 2 minutes given the data collected in Section 1.3.5, the complexity of optimizing DATA-GENIE-ranking is still squared *w.r.t.* the total number of sampling schemes, whilst that of DATA-GENIE-regression is linear.

Architecture. To compute $\Phi(\mathcal{E}(\mathcal{D}_f), \mathcal{E}(\mathcal{D}_f^{s,p}), m)$ we concatenate $\mathcal{E}(\mathcal{D}_f)$, $\mathcal{E}(\mathcal{D}_f^{s,p})$, one-hot embedding of m; and pass it through two relu-activated MLP projections to obtain $\hat{\tau}(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p})$. For DATA-GENIE-regression, we also pass the final output through a tanh activation, to reflect the range of Kendall's Tau *i.e.* $[-1, 1]$.

Inference. Since computing both $\mathcal{E}$ and Φ are agnostic to the datasets and the sampling schemes, we can simply use the trained $\mathcal{E}$ and Φ functions to rank *any* sampling scheme for *any* CF dataset. Computationally, given a trained DATA-GENIE, the utility of a sampling scheme can be computed simply by computing $\mathcal{E}$ twice, along with a single pass over Φ, completing in the order of milliseconds.

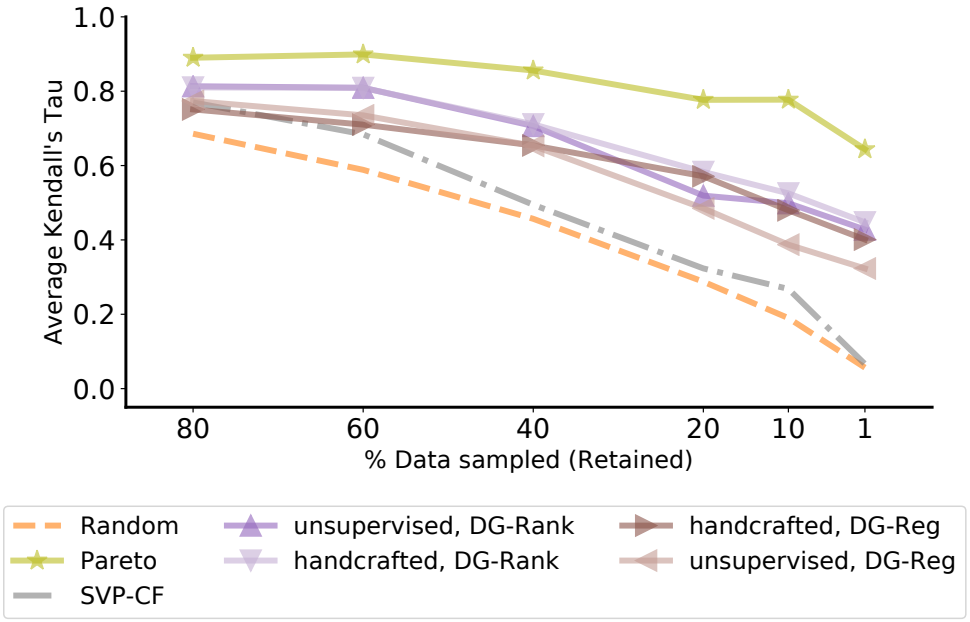

Figure 1.4. Comparison of the average Kendall's Tau with % data sampled for different sampling-selection strategies. A higher Tau indicates better retaining power of the ranking of different recommendation algorithms.

1.4.4 Experiments

Setup. We first create a train/validation/test split by randomly splitting all possible metrics and sampling % pairs (m, p) into $70/15/15\%$ proportions. Subsequently for each dataset $\mathcal{D}$, CF-scenario f, and (m, p) in the validation/test-set, we ask DATA-GENIE to rank all 16 samplers (Table 1.3) for $p\%$ sampling of $f-$type feedback for $\mathcal{D}$ and use metric m for evaluation by sorting $\hat{\tau}$ for each sampler, as defined in Equation (1.1). To evaluate DATA-GENIE, we use the P@1 metric between the actual sampler ranking computed while computing $\Psi-$scores in Section 1.3.5, and the one estimated by DATA-GENIE.

How accurately can DATA-GENIE predict the best sampling scheme? In Table 1.4, we compare all dataset representation choices $\mathcal{E}$, and multiple Φ architectures for the task of predicting the best sampling strategy. In addition to the regression and ranking architectures discussed in Section 1.4.3, we also compare with linear least-squares regression and XGBoost

regression (Chen and Guestrin, 2016) as other choices of Φ. In addition, we compare DATA-GENIE with simple baselines: (1) randomly choosing a sampling strategy; and (2) the best possible static sampler choosing strategy—always predict user sampling *w/* Bias-only SVP-CF-PROP. First and foremost, irrespective of the $\mathcal{E}$ and Φ choices, DATA-GENIE outperforms both baselines. Next, both the handcrafted features and the unsupervised GCN features perform quite well in predicting the best sampling strategy, indicating that the graph characteristics are well correlated with the final performance of a sampling strategy. Finally, DATA-GENIE-regression and DATA-GENIE-ranking both perform better than alternative $\Phi-$choices, especially for the P@1 metric.

Can we use DATA-GENIE to sample more data without compromising performance? In Figure 1.4, we compare the impact of DATA-GENIE in sampling more data by dynamically choosing an appropriate sampler for a given dataset, metric, and % data to sample. More specifically, we compare the percentage of data sampled with the Kendall's Tau averaged over all datasets, CF-scenarios, and relevant metrics for different sampling strategy selection approaches. We compare DATA-GENIE with: (1) randomly picking a sampling strategy averaged over 100 runs; and (2) the Pareto frontier as a skyline which always selects the best sampling strategy for any CF-dataset. As we observe from Figure 1.4, DATA-GENIE is better than predicting a sampling scheme at random, and is much closer to the Pareto frontier. Next, pairwise ranking approaches are marginally better than regression approaches irrespective of $\mathcal{E}$. Finally, DATA-GENIE can appraise the best-performing recommendation algorithm with a suitable amount of confidence using only 10% of the original data. This is significantly more efficient compared to having to sample $50-60\%$ if we were to always sample using a fixed strategy.

1.5 Discussion

In this work, we discussed two approaches for representative sampling of CF-data, especially for accurately retaining the *relative* performance of different recommendation algorithms.

Table 1.5. CO_2 emissions comparison (Strubell et al., 2019a).

Consumption	CO_2e (lbs.)
1 person, NY↔SF flight	2k
Human life, 1 year avg.	11k
Weekly RecSys development cycle	20k
" *w/* DATA-GENIE	3.4k

First, we proposed SVP-CF, which is better than commonly used sampling strategies, followed by introducing DATA-GENIE which *analyzes* the performance of different samplers on different datasets. Detailed experiments suggest that DATA-GENIE can confidently estimate the downstream utility of any sampler within a few milliseconds, thereby enabling practitioners to benchmark different algorithms on 10% data sub-samples, with an average 5.8× time speedup.

To realize the real-world environmental impact of DATA-GENIE, we discuss a typical weekly RecSys development cycle and its carbon footprint. Taking the Criteo Ad dataset as inspiration, we assume a common industry-scale dataset to have $\sim$ 4B interactions. We assume a hypothetical use case that benchmarks for *e.g.* 25 different algorithms, each with 40 different hyper-parameter variations. To estimate the energy consumption of GPUs, we scale the 0.4 minute MLPerf (Mattson et al., 2020) run of training NeuMF (He et al., 2017) on the Movielens-20M dataset over an Nvidia DGX-2 machine. The total estimated run-time for all experiments would be $25 \times 40 \times \frac{4B}{20M} \times \frac{0.4}{60} \approx 1340$ hours; and following Strubell et al. (2019a), the net CO_2 emissions would roughly be $10 \times 1340 \times 1.58 \times 0.954 \approx 20k$ lbs. To better understand the significance of this number, a brief CO_2 emissions comparison is presented in Table 1.5. Clearly, DATA-GENIE along with saving a large amount of experimentation time and cloud compute cost, can also significantly reduce the carbon footprint of this *weekly process* by more than an average human's *yearly* CO_2 emissions.

Despite having significantly benefited the run-time and environmental impacts of benchmarking algorithms on massive datasets, our analysis heavily relied on the experiments of

training seven recommendation algorithms on six datasets and their various samples. Despite the already large experimental cost, we strongly believe that the downstream performance of DATA-GENIE could be further improved by simply considering more algorithms and diverse datasets. In addition to better sampling, analyzing the fairness aspects of training algorithms on sub-sampled datasets is an interesting research direction, which we plan to explore in future work.

Chapter 1, in part, is a reprint of the material as it appears in "On Sampling Collaborative Filtering Datasets." by Noveen Sachdeva, Carole-Jean Wu, and Julian McAuley, in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining. 2022.* The dissertation author was the primary investigator and author of this paper.

Chapter 2

Dataset Pruning for Pretraining Large Language Models

The training of large language models (LLMs) is expensive. In this chapter, we study data-efficient approaches for pre-training LLMs, *i.e.*, techniques that aim to optimize the Pareto frontier of model quality and training resource/data consumption. We seek to understand the tradeoffs associated with data selection routines based on (i) expensive-to-compute *data-quality* estimates, and (ii) maximization of *coverage* and diversity-based measures in the feature space.

Our first technique, ASK-LLM, leverages the zero-shot reasoning capabilities of instruction tuned LLMs to directly assess the quality of a training example. To target coverage, we propose DENSITY sampling, which models the data distribution to select a diverse sample.

Testing the effect of 22 different data curation techniques on the pre-training of T5-style of models, involving hundreds of pre-training runs and post fine-tuning evaluation tasks, we find that ASK-LLM and DENSITY are the best methods in their respective categories. Coverage sampling can *recover* the performance of the full data, while models trained on ASK-LLM data consistently *outperform* full-data training—even when we reject 90% of the original dataset, while converging up to 70% faster.

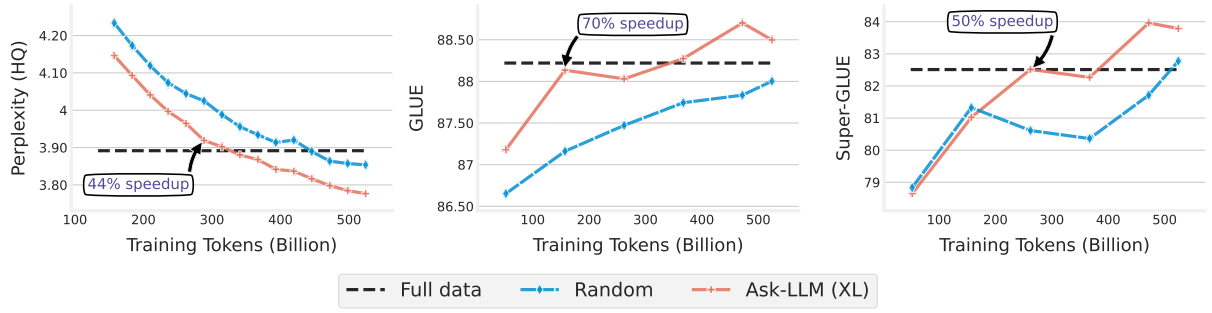

Figure 2.1. Data-efficient pre-training run of T5-Large (800M) using ASK-LLM with Flan-T5-XL as the data quality scorer. Training on 60% of the original dataset, ASK-LLM is able to train T5-Large both better and 70% faster, compared to training on 100% of the dataset.

2.1 Introduction

Large language model (LLM) pre-training is perhaps the most data- and compute-intensive task attempted by the machine learning community to date, with impressive capabilities primarily being accomplished by training massive transformer architectures on trillions of tokens of text (OpenAI, 2023; Gemini et al., 2023; Touvron et al., 2023b).

But even these incredibly capable LLMs are subject to empirical scaling laws, which predict sharply diminishing returns from a linear increase in model- or data-size (Hoffmann et al., 2022; Kaplan et al., 2020). Power-law scaling therefore acts as a soft limit on model quality, beyond which it is prohibitively expensive to drive performance by scaling up the data or model. At the same time, Sorscher et al. (2022)—in the context of vision pre-training—show that we can significantly improve the power law constants in the aforementioned scaling laws if we prioritize *important* training examples using some robust notion of data quality or impact.

A similar call for data-curation is also apparent in the context of training LLMs, where our largest models are quickly approaching their capacity and data thresholds. LIMA (Zhou et al., 2023) showed that LLaMA-65B (Touvron et al., 2023a) can be better aligned with human preferences when trained on a set of 1,000 carefully selected fine-tuning prompts, compared to training on as much as 52,000 unfiltered examples. Tirumala et al. (2023) recently conducted a large-scale data-efficient pre-training evaluation, showing that a 6.7B OPT model (Zhang

et al., 2022) can converge up to 20% faster on data curated by a technique based on stratified cluster sampling. The Phi-2 experiments also suggest that when data curation is performed at a human-expert level (*e.g.*, by textbook editors), models can outperform baselines that are up to 25x larger (Javaheripi et al., 2023).

Data curation routines can be fundamentally characterized as selecting training samples for quality, coverage, or some mixture of both (Figure 2.2). In this work, we seek to understand how quality and coverage affect the data efficiency of LLM pre-training. Our core research question is:

> *"Are cheap-to-compute heuristics like maximum-coverage enough to pre-train a SoTA LLM, or are there real benefits from costly samplers that carefully evaluate the quality of each example?"*

This question is crucial to answer because data-curation algorithms can improve the Pareto frontier of the data-quantity↔model-quality tradeoff, directly addressing the bottleneck of power-law scaling by enabling higher-quality models to be trained using less data. Data curation also unlocks new tradeoffs between training time, inference cost, data collection effort, and downstream performance. For example, if we consider the compute-constrained (single-epoch) regime, a data-efficient LLM training routine may reach the desired performance using only X% of the data (corresponding to a $\leq$X% training speedup).

Despite considerable interest from the community for building data-efficient training methods (Sorscher et al., 2022; Paul et al., 2021; Coleman et al., 2020; Jiang et al., 2019; Katharopoulos and Fleuret, 2018), large-scale analyses of data pruning strategies are rare because of the extreme computational cost—especially in the context of LLM pre-training. To be more specific, an extensive comparative study necessarily entails pre-training (i) various sizes of LLMs, (ii) for a variety of data sampling rates, (iii) obtained through various pruning strategies. Further, downstream evaluations for LLMs also frequently involve fine-tuning, which is resource intensive in itself.

2.1.1 Contributions

We hypothesize that the roles of coverage and quality depend on the stage of training, size of the model, and the sampling rate. To understand the coverage/quality design choice better, we develop new data-efficiency routines that independently (and solely) target quality and coverage. Our ASK-LLM sampler prioritizes high-quality and informative training samples by asking a *proxy* LLM. Our DENSITY sampler seeks to maximize the coverage of latent topics in the input dataset through a diversified sampling procedure. To summarize, our contributions are as follows:

ASK-LLM sampling. We find that ASK-LLM can train better models (*vs.* training on the *entire dataset*) even after removing up to 90% of training samples, while also consistently beating well-established data curation routines. Furthermore, we find ASK-LLM also promotes data-efficiency during training (Figure 2.1).

Exhaustive benchmark. We implement 22 different sampling strategies for pre-training T5-Large (800M) and T5-Small (60M) on 524B tokens and evaluate them on 111 downstream evaluation tasks. This leads to a total of 220 pre-training and $1,100$ distinct fine-tuning runs.

New insights. By analyzing the differences between ASK-LLM and DENSITY sampling, we study the role of coverage, quality, and sampling cost in LLM pre-training. We support our conclusions with additional studies of the convergence rate, correlations between sampler outputs, and impact of sampling cost on downstream performance.

Takeaway. Our results show that while coverage sampling can *recover* the performance of the full data, ASK-LLM (quality filtering) can often *exceed* it. These experiments suggest that LLM-based quality raters are a worthwhile and effective way to drive performance in pre-training.

2.2 Related Work

Data selection is a classical problem with well-established literature on coresets, sketching, importance sampling, filtering, denoising, and a host of other algorithms with similar goals. While we cannot possibly catalog the entire sampling literature, we hope to provide an overview of the principles behind common data selection algorithms. We also describe how these algorithms have been applied to machine learning, with a focus on language model training.

2.2.1 Coverage Sampling

The first class of methods maximize the *coverage* of the sample by selecting points that are evenly distributed across the entire input domain, e.g., an ε-net for a Lipschitz function (Phillips, 2017). When training language models, coverage sampling is motivated by the intuition that we ought to show the model the full breadth of genres, topics, and languages (Longpre et al., 2023b). Coverage sampling is typically accomplished by embedding examples into a metric space and selecting points which are mutually far from each other (Lee et al., 2023).

Cluster sampling algorithms group inputs based on embedding similarity and select representatives from each group. These algorithms are popular, scalable, interpretable, and enjoy strong theoretical support – k-means sampling provably approximates the SVM objective (Tukan et al., 2021) and many others (Feldman et al., 2020). However, there are also recent techniques based on submodular optimization of a coverage score (Chen et al., 2012; Indyk et al., 2014; Borsos et al., 2020a), models of the data distribution (Coleman et al., 2022), discrepancy minimization (Karnin and Liberty, 2019), and deduplication through token matching / locality-sensitive hashing (Lee et al., 2022).

Many variations of cluster sampling have been applied to vision and language model training. Sorscher et al. (2022) propose the "SSL prototypes" method for vision models, which removes points that fall too close to the nearest k-means centroid. SemDeDup (Abbas et al., 2023) also removes points based on this distance, but targets pairs of nearby examples, or

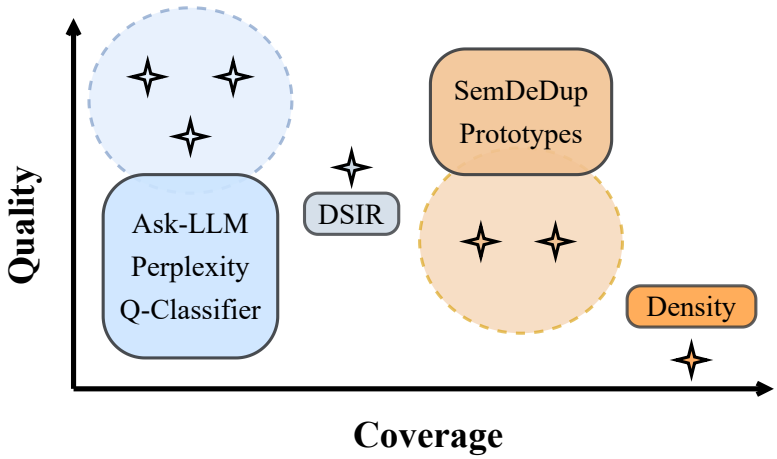

Figure 2.2. While there is no inherent tradeoff between coverage and quality, samplers target these metrics on a spectrum (up and to the left indicates a more aggressive prioritization). See Section 2.4.3 for a description of the plotted samplers.

"semantic duplicates," and prefers points close to the centroid. The D4 sampler chains MinHash deduplication, SemDeDup, and SSL prototypes together to prune both high-variance, sparse regions and prototypical, dense regions of LLM pre-training datasets (Tirumala et al., 2023). Coleman et al. (2020) considers a k-centers submodular selection routine on the last-layer embeddings of ResNet vision models.

2.2.2 Quality-score Sampling

Another class of methods are based on *quality scores*, where a scoring algorithm rates every example and the sampler preferentially selects points with high scores. Even though this framework was originally developed for importance sampling (Hastings, 1970), the machine learning community has expanded the theoretical "score-and-sample" framework to include a variety of practical heuristics.

For example, the selection-via-proxy (SVP) algorithm determines the importance of an input using the validation loss and uncertainty scores of a pre-trained model on the input (Coleman et al., 2020; Sachdeva et al., 2021). Paul et al. (2021) sample according to an "EL2N score" formed by ensembling the losses of 10 lightly-trained models. Ensemble prediction variance has also been used as the scoring metric (Chitta et al., 2021), as have ensemble disagreement

rates (Meding et al., 2021). Other scores measure whether an example is likely to be forgotten (Toneva et al., 2019), memorized (Feldman and Zhang, 2020), or un-learnable (Mindermann et al., 2022).

In the context of pre-training LLMs, there exist a few different schools-of-thought for scoring the quality of training samples. The first (and arguably most used) camp is perplexity-filtering, where we prioritize samples with *low* perplexity and filter out highly surprising examples (Wenzek et al., 2019; Marion et al., 2023; Muennighoff et al., 2023). Notably, recent advancements in cheaper to run model-based *training-run simulators* for LLMs can be used to *estimate* the perplexity of a training sample instead of running an LLM inference (Guu et al., 2023). Another group of methods selects training data that minimizes the *distance* between the distribution of selected data and a handcrafted high-quality data source (typically wikipedia and books). Typical ways are to do this in a feature space (Xie et al., 2023b) or by training a contrastive-style classifer (Radford et al., 2019; Anil et al., 2023; Javaheripi et al., 2023). Similar ideas have also been explored for optimizing the data mixture weights for pre-training (Xie et al., 2023a).

Unlike using LLMs for quality evaluation, in concurrent work, Maini et al. (2024) also show promise for an LLM-based data re-writing approach. Engstrom et al. (2024) consider a quality evaluation based on datamodels, though their analysis suggests that this approach selects for strongly model-dependent notions of quality.

2.3 Methods

We propose two samplers, ASK-LLM and DENSITY. These samplers have significantly different costs—ASK-LLM requires an LLM inference call for each training sample, whereas DENSITY is based on a diversified sampling routine that is cheaper than even clustering the dataset. They also exhibit substantially different selection behavior: ASK-LLM conducts a highly *nuanced* and *contextual* quality evaluation for each sample, while DENSITY asks whether we have already sampled many similar examples. By studying samplers on extreme ends of this

Figure 2.3. The prompt for obtaining the sampling score for each training sample in ASK-LLM.

spectrum, we hope to better understand the salient factors for LLM data curation.

2.3.1 ASK-LLM Sampling

Intuition. Our intuition is that humans can easily identify commonly occurring failure modes in state-of-the-art data quality scorers. Hence, it should be possible to correct these mistakes using the reasoning capabilities of modern instruction-tuned LLMs.

To do so, in ASK-LLM, we directly prompt an instruction-tuned *proxy* LLM with the prospective training example for the sampling decision (see Figure 2.3 for the prompt). We take the softmax probability of the token "`yes`" as the estimated data-quality score. Consider the following common failure modes of perplexity filtering (*i.e.*, keeping the lowest perplexity examples), which the ASK-LLM scoring model fixes (see more qualitative examples in Section 2.6):

- *Contextuality.* Perplexity filters often select samples that lack context, *e.g.*, containing

Algorithm 1. ASK-LLM Sampling

Input: Dataset $\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$ s.t. $x_i \in \mathcal{X}$ is the training sample in plain-text, sample size k, scoring model $\mathcal{M} : \mathcal{X}; \mathcal{X} \mapsto \mathbb{R}$
Output: Sampled data

Initialize list of scores $S = []$.
for $n = 1 \rightarrow N$ **do**
    $\text{prompt}_n \leftarrow \text{make\_prompt}(x_n)$ ▷ Make ASK-LLM prompts as in Figure 2.3
    Append $\mathcal{M}(\text{“}\texttt{yes}\text{”} \mid \text{prompt}_n)$ to S ▷ Use $\mathcal{M}$ to score x_n
return k elements from $\mathcal{D}$ with top-k scores in S, without replacement.

questions without answers (Examples 5, 6). ASK-LLM correctly identifies that these examples do not provide new information.

- *Nonsense.* Perplexity filters can select examples that endlessly repeat the same phrases / words , likely because these word combinations are common (resulting in high likelihood).

- *Niche examples.* Perplexity filters can reject niche topics that are otherwise informative, well-written, and contain useful *tail knowledge* of the world. Example 7 contains detailed information about a Manchester art installation but is assigned a high perplexity, likely because it contains uncommon (but valid) word combinations.

Discussion on the cost of ASK-LLM scoring. Even though ASK-LLM sampling results in impressive performance and training efficiency improvements compared to training on the full-dataset (Figure 2.6), the data quality scoring cost might seem prohibitive. On the other hand, on top of the improved results, we argue the following to be compelling points in justifying ASK-LLM's one-time-amortized data scoring cost:

- ASK-LLM only requires *forward passes* on the entire dataset. This is much cheaper than (i) training the model itself which requires both forward and backward passes on multiple repetitions of the entire dataset, (ii) gradient-based data-curation techniques (Sachdeva and McAuley, 2023; Sachdeva et al., 2023) that also require backward passes, *etc.*

- An additional benefit of the ASK-LLM framework is the ability to leverage memory-efficient, quantized LLM inference setups (Dettmers et al., 2022). This is strictly not possible, *e.g.*, for pre-training LLMs. Notably, quantization isn't the only ASK-LLM-friendly technique. All the recent (and future) advances in efficient *inference* techniques for LLMs (Weng, 2023) directly reduce the amortization cost of the ASK-LLM framework.

- Another benefit of ASK-LLM is the ability to naïvely parallelize quality scoring. To be more specific, we can simply scale-up the amount of *small & independent* inference resources, and run inference calls for various training samples parallely. Note that inference hardware has much smaller requirements compared to, *e.g.*, pre-training or fine-tuning requirements. This is primarily true because of no batch size requirement for inference *vs.* large batch size requirement while training. This enables scaling-up hardware to happen via a large number of small-compute setups (*e.g.*, 4 interconnected GPUs per node) versus increasing the number of large-compute setups (*e.g.*, 1000s of interconnected GPUs per node).

- ASK-LLM also uses strictly less compute compared to teacher-student knowledge distillation based training setups (Agarwal et al., 2023). This is true simply because knowledge distillation require (i) bigger teacher model's softmax predictions (ii) for each token in our training data. On the other hand, ASK-LLM requires just the score of the token "`yes`" given the prompt.

2.3.2 DENSITY Sampling

Intuition. Our intuition is that the data distribution provides a strong coverage signal. High-probability regions contain "prototypical" examples—ones with many near-duplicates and strong representation in the dataset. Low-probability regions will contain outliers, noise, and unique/rare inputs. If we wish to maximize topic coverage, we should boost the signal from under-represented portions of the input domain and downsample redundant, high-density information.

The key difficulty for our DENSITY sampler is to accurately estimate an example's local density. Like Tirumala et al. (2023) (D4), we assume access to embeddings from a pre-trained

Algorithm 2. Inverse Propensity Sampling (IPS) via Kernel Density Estimation (KDE)

Input: Dataset $\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$ of embeddings, sample size k, kernel k with corresponding locality-sensitive hash family $\mathcal{H}$ (see Coleman and Shrivastava (2020)), hash range B, rows R, random seed s
Output: Sampled data

Initialize: KDE sketch $\mathcal{S} \leftarrow 0^{R \times B}$
Generate R independent hash functions $h_1, \ldots, h_R$ from $\mathcal{H}$ with range B and random seed s.
for $n = 1 \rightarrow N$ **do** ▷ Construct KDE estimator for D.
    for $r = 1 \rightarrow R$ **do** ▷ Add x_n to the KDE estimator.
        $\mathcal{S}_{r,h_r(x_n)} += 1$
Initialize list of scores $S = []$.
for $n = 1 \rightarrow N$ **do** ▷ Score each example x_n
    score $= 0$
    for $r = 1 \rightarrow R$ **do** ▷ Compute approximate KDE using $\mathcal{S}$
        score$+= \mathcal{S}[r, h_r(x_n)]$
    Append score$/R$ to S
return k elements from $\mathcal{D}$ with probability $p = \frac{S}{\sum S}$ without replacement.

LLM. However, we depart from the traditional approach of clustering and opt to sample based on kernel sums. Given a dataset D of embeddings and a kernel $k(x,y)$, we estimate the density using the following score.

$$\text{score}(y) = \sum_{x \in D} k_\lambda(x,y).$$

λ is a smoothing parameter called the *kernel bandwidth* that controls the scale of the points' effects. To reduce the complexity from $O(N^2)$ to $O(N \log N)$, we use recent breakthroughs from the algorithm community to approximate the sum (Siminelakis et al., 2019; Coleman and Shrivastava, 2020). Our density sampler is adapted from that of Coleman et al. (2022), with a few critical departures:

- We use a two-pass procedure that allows for more rigorous theoretical guarantees (and different sampling behavior).
- We conduct the density estimation in the model's latent space rather than using Jaccard similarity over n-grams.

Improvements. Jaccard similarities are sufficient to construct a reasonable sampling distribution for genomics applications, which are significantly more structured than natural language. However, this is not the case with text — we found that sampling based on Jaccard density is no better than random. For this reason, we must use different kernels (p-stable rather than MinHash) and different input representations (embedding rather than n-grams).

However, our more interesting departure from Coleman et al. (2022) is our two-pass sampling procedure, which changes the behavior of the algorithm and allows for more rigorous theoretical guarantees. The original method was only able to demonstrate convergence of cluster populations in the sampled dataset. While this leads to (weak) convergence for some measures of diversity, it also requires strong assumptions about the cluster structure.

Theoretical Analysis. We use a recent result that demonstrates consistent sketch-based estimation of the kernel sum (Theorem 3.3 of Liu et al. (2023)), which we paraphrase below.

Lemma 2.3.1. *Let $P(x)$ denote a probability density function. Let $\mathcal{D} \underset{\text{iid}}{\sim} P(x)$ denote a dataset. Let $k(x,y)$ be a positive definite LSH kernel, and let S be the* DENSITY *score. Then $S(x)$ is a consistent estimator for the kernel sum.*

$$S(x) \underset{\text{i.p.}}{\to} \frac{1}{N} \sum_{x_i \in \mathcal{D}} k(x_i, q)$$

with convergence rate $O(\sqrt{\log R / R})$.

If we perform inverse propensity sampling using the score in Lemma 2.3.1, we obtain a sampling procedure that outputs a uniformly-distributed sample.

Theorem 2.3.2. *Let $Q(x)$ be the distribution formed by (i) drawing N samples i.i.d. from a distribution P,* e.g. *$\mathcal{D} = \{x_1, \ldots x_N\} \sim P$, and (ii) keeping x with probability proportional to $\frac{1}{S(x)}$. Under the conditions of Lemma 2.3.1, $Q(x) \underset{\text{i.p.}}{\to} U(x)$, where $U(x)$ is the uniform distribution.*

Proof. Under the conditions of Wied and Weißbach (2012) (specifically, positive-definiteness and ℓ_1 integrability / bounded domain), the kernel sum is a consistent estimator of the density.

That is, the sum converges in probability to $P(x)$.

$$\frac{1}{N}\sum_{x_i \in \mathcal{D}} k(x_i, q) \underset{\text{i.p.}}{\to} P(x)$$

Lemma 2.3.1 shows that $S(x)$ converges in probability to the sum (and thus to $P(x)$). By Slutsky's Theorem, $\frac{1}{S(x)} \to \frac{1}{P(x)}$ for all x in the support of the distribution (i.e. $P(x) \neq 0$). The probability of generating x as part of the sample is:

$$Q(x) = \Pr[\text{Select} x \cap \text{Generate} x] = \Pr[\text{Select} x]\Pr[\text{Generate} x] = \frac{1}{S(x)}P(x)$$

Because $\frac{1}{S(x)} \to \frac{c}{P(x)}$ for some constant c, we have that $Q(x) \to c$. □

Theorem 2.3.2 demonstrates that our DENSITY sampler outputs a uniformly-distributed collection of points over the input space (latent LLM representation space).

Cost. Like SemDeDup, D4, and SSL prototypes, our DENSITY sampler requires access to embeddings for each example in the training corpus. However, by eliminating the expensive clustering step, we eliminate a significant computational overhead. Our DENSITY sampling routine required just 80MB of memory and two linear passes through the dataset to score all 364M embeddings. This is significantly less expensive than clustering.

Tuning. We also eliminate a large number of hyperparameters, improving tuning. Cluster-based samplers must choose the number of clusters, clustering optimizer and objective, and per-cluster sampling rate or deduplication similarity. Kernel density estimation, on the other hand, has just *two* hyperparameters: the choice of kernel and the bandwidth. We did not observe a significant performance variation among different bandwidth and kernel choices (e.g., the L2 and cosine kernels of Coleman and Shrivastava (2020) perform nearly identically). This is likely because all positive-definite kernels enjoy strong guarantees on the distribution approximation error (Devroye, 1983).

2.3.3 Sampling Techniques

DENSITY and ASK-LLM are both *scoring* methods that reduce an example to a floating point value that measures coverage or quality. Once we have scores for a complete dataset of training examples (sentences, paragraphs, etc.), we can make score-based decisions about which examples to include in the training set.

Top / Bottom K. The simplest method is to sort examples by score and accept the top or bottom K, depending on whether a higher *or* lower score signifies better data-quality. While straightforward, this approach is supported by the "permutation" theory of Sorscher et al. (2022), and sensitivity score sampling (a softened version) is the core subroutine for many coresets (Mai et al., 2021). When applied to DENSITY and perplexity scores, top-K sampling selects for the head of the data distribution (similar to SSL prototypes). Bottom-K sampling selects the tail and removes common items.

Inverse Propensity Sampling. Inverse propensity sampling (IPS) selects items proportional to their reweighted and normalized *inverse* score (Rosenbaum and Rubin, 1983). When applied to DENSITY or perplexity scores, IPS implements a form of *diversified sampling* that uniformizes the distribution of selected inputs (Theorem 2.3.2).

In our experiments, the DENSITY sampler uses IPS to maximize the coverage of the dataset.[1] For our ASK-LLM filter, we adopt top-k sampling because we expect the "yes" probability to be a reliable and strong measure of quality.

2.3.4 Relationships Between Methods

DENSITY, Perplexity, and Loss. When a language model is trained to minimize perplexity, the LLM itself is a data distribution model. Therefore, the perplexity and loss filtering approaches of Marion et al. (2023), Muennighoff et al. (2023), and other authors can be viewed as model-

[1] We also implemented top-K and bottom-K sampling, but these samplers do not maintain coverage and perform poorly.

based density sampling. However, our sampler measures the density of the training dataset in a latent geometric space, while perplexity measures the likelihood under the scoring model. The samplers also differ in terms of decision complexity. Thanks to the capacity of the LLM, a perplexity filter can make highly-nuanced decisions between two texts on the same topic. On the other hand, our DENSITY sampler is constructed from a simple nonparametric density model (Rosenblatt, 1956) that does not have the capacity to distinguish examples at such a granular level.

ASK-LLM and Perplexity. Perplexity filters exhibit a strong in-distribution bias, making decisions based on the data used to train the scoring model (not the dataset we wish to sample). By using the LLM for quality evaluation rather than likelihood estimation, our sampler can escape this bias because the additional context and alternative task change the sampling distribution. This occurs even when the ASK-LLM and perplexity models are the same size.

DENSITY and Clustering. The kernel sum procedure at the core of our DENSITY sampler operates on embedding-similarity relationships in a similar way to D4, SemDeDup, and SSL prototypes. Indeed, near-duplicate detection can be viewed as a discretized version of similarity-based density estimation (Kirsch and Mitzenmacher, 2006). Outlier rejection, which motivates the "nearest-to-centroid" heuristic of SSL prototypes, also has intimate connections with density estimation (Schubert et al., 2014).

Intuition. Perplexity should be viewed as a "difficulty" or "quality" score rather than as a coverage-maximizing score. Our ASK-LLM sampler should be viewed as a contextualized quality score that incorporates reasoning.[2] Our DENSITY sampler is a pure "coverage" score in the latent representation space, while SemDeDup, and SSL Prototypes all incorporate quality / outlier filtering to some extent (*e.g.*, by preferring points near / far from a centroid).

[2]Note that ASK-LLM may also incidentally improve coverage because it does not suffer from in-distribution bias.

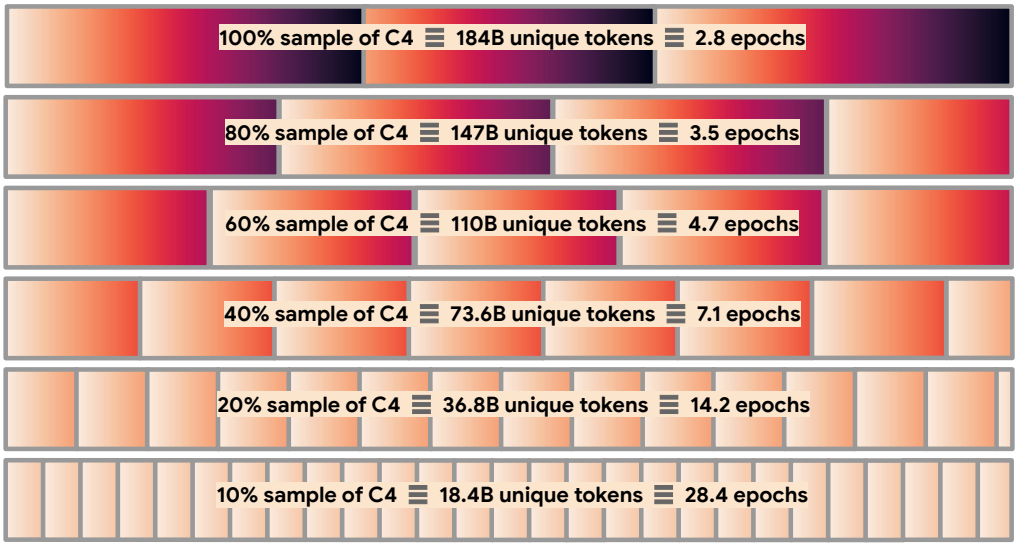

Figure 2.4. We consider a setup where all of our models are trained on exactly 524B tokens, causing us to repeat the same examples for more epochs when we downsample. We borrow the format of this graphic from Muennighoff et al. (2023), who consider a similar setting.

2.4 Empirical Setup

2.4.1 Models

We pre-train T5-style models (Raffel et al., 2020), which belong to the encoder-decoder family of Transformer models and offer competitive performance on many tasks (Shen et al., 2023). See Phuong and Hutter (2022) for a formal introduction to various Transformer model configurations. We train T5-Small (60M) and T5-Large (800M), reusing all of the training settings from the original T5 implementation except the batch size ($2048 \rightarrow 1024$). We train on batches of 1024 sequences of length 512 for 1M steps.

2.4.2 Dataset

We use the C4 dataset (Raffel et al., 2019), which was also used for pre-training the original T5. The C4 dataset is a version of the Common Crawl—a publicly available archive of web-text—that has been pre-processed using several heuristics (Raffel et al., 2020, Section 2.2). In its entirety, C4 contains 184B tokens. We use our algorithms (see Section 2.4.3 for a list) to

sample $\{10, 20, 40, 60, 80\}\%$ of C4.

Because a low sampling ratio yields exceedingly small datasets, we choose to train in the iso-compute setting, *i.e.*, training all models for exactly 524B tokens. This results in more epochs (repetitions) at smaller sampling rates. We believe this gives each data curation method an equal chance to maximize model performance, and not penalize methods that sample a small number of high-quality repeatable tokens *vs.* large number of non-repeatable tokens. See Figure 2.4 for a demonstration of this process.

2.4.3 Baseline Samplers

Random Sampling. The de-facto standard for obtaining samples of large datasets where we sample training examples uniformly at random. Notably, random sampling has also been accompanied with strong results in a variety of applications in the data-curation literature primarily due to its unbiased sampling (Ayed and Hayou, 2023b; Guo et al., 2022b).

DENSITY Sampling. See Section 2.3.2 for technical details about the DENSITY sampler. We use Sentence-T5-Base (Ni et al., 2021) as our embedding model for training samples, primarily due to its contrastive training, giving confidence for computing distances amongst its 768-dim embeddings. We use the PStable hash (Datar et al., 2004) to hash the embeddings, along with a $[1,000 \times 20,000]$ sketch matrix.

SemDeDup. The key idea is to perform (coverage maximizing) semantic deduplication inside clusters of the original dataset (Abbas et al., 2023). We re-use the Sentence-T5-Base embeddings of data-points used for DENSITY sampling, and perform k-means clustering to obtain $10,000$ clusters of the entire dataset.

SSL Prototypes. They key idea is to remove *prototypical* points in a dataset (Sorscher et al., 2022). As a meaningful proxy, this method removes the points closest to cluster centroids of a dataset. For brevity, we use the name "Prototypes" when reporting our results. We re-use the same embeddings and clustering for both SemDeDup and Prototypes.

Perplexity Filtering. A popular quality-filtering approach in the literature is to use the perplexity of proxy language models to filter data-points with a high-perplexity under that language model. While the literature historically used small language models for perplexity filtering (Wenzek et al., 2019; Muennighoff et al., 2023), recent work (Marion et al., 2023) suggests improved filtering performance when using LLMs for this task. To this end, we employ perplexity filtering with T5-{Small, Base, Large, XL, XXL} models; as well as intermediate checkpoints during the course of training T5-Large: {20k, 100k, 300k, 500k, 700k}.

Text-quality Classifier (Q-Classifier). First proposed by GPT-3 for curating its pretraining dataset (Brown et al., 2020), and later used by various state-of-the-art LLMs at their time (Chowdhery et al., 2023; Anil et al., 2023; Gao et al., 2020; Du et al., 2022) another popular quality filtering approach is to train a linear classifier for distinguishing web-scrape data *vs.* known reference high-quality data. Consistent with existing usage of this technique (Gao et al., 2020; Xie et al., 2024; Brown et al., 2020; Chowdhery et al., 2023; Du et al., 2022), we train a hashing-based linear classifier with a hash size of 262k trained to classify if a document is either from (negative) C4 or (positive) Wikipedia + BookCorpus. We train this classifier for a total of 218k steps (equivalent to 14 Trillion unigrams), and based on recent evidence (Xie et al., 2024) we sample the documents with the highest score according to this classifier.

Data Selection with Importance Resampling (DSIR). Proposed by Xie et al. (2024), DSIR performs importance sampling using a bag-of-words estimator (we use unigram and bigram features) over some "high-quality" target data-source. This approach is, in spirit, quite similar to the aforementioned text-quality classification approach but performs distribution-matching in a non-parametric way, and without the hassle of training a classifier on large piles of data. To be consistent, we use Wikipedia + BookCorpus as the target source for DSIR as well. We re-use the official public implementation for DSIR[3].

[3]https://github.com/p-lambda/dsir

ASK-LLM Sampling. See Section 2.3.1 for technical details about the ASK-LLM sampler. Since ASK-LLM relies on the reasoning capabilities of instruction-tuned models, we use the Flan-T5-{Small, Base, Large, XL, XXL} (Longpre et al., 2023a) and instruction tuned Gemma-7B (Team et al., 2024) models for obtaining the quality scores in ASK-LLM.

2.4.4 Evaluation Tasks

We use 111 downstream evaluation tasks to assess diverse performance indicators for pre-trained LLMs as listed below.

Perplexity. Defined as the exponentiated average negative log-likelihood of an average sequence in the dataset; we report the perplexity computed over the target tokens in T5's denoising objective (Raffel et al., 2020) over the default validation set provided by C4. Note that C4's validation set is a random sample of the dataset, so it is prone to be of much lower quality than curated sources, and hence, a less reliable indicator of true model quality.

HQ Perplexity. As our best effort to devise an inexpensive-to-compute metric that is better aligned with model quality than perplexity on C4's validation set, inspired by the evaluation conducted in Tirumala et al. (2023), we construct a *high-quality* validation set from non web-scrape sources. We collate the validation sets from (1) English portion of wiki40b (Guo et al., 2020), (2) realnews and webtext subsets of C4, and (3) news commentary from the LM1B dataset (Chelba et al., 2013).

GLUE. A popular natural language understanding meta-benchmark comprising of eleven different tasks (Wang et al., 2018a). Note that we report the average score for all individual tasks, after finetuning on the concatenation of all individual tasks' training sets, as is done in the original T5 implementation.

SuperGLUE. A harder meta-benchmark (*vs.* GLUE) built to further test the natural language understanding abilities of language models (Wang et al., 2019). Similar to GLUE, we report the

average score of all tasks, and conduct fine-tuning on all tasks' concatenated train-set.

CNN/DM. We use the CNN/DM dataset (Hermann et al., 2015) for testing our models' abstractive summarization abilities. Like the T5 original setting, we finetune on the train-set, and report the ROUGE-2 scores.

SQuAD. A popular dataset (Rajpurkar et al., 2016) used to evaluate question-answering capabilities of language models, we compare the finetuned performance of our models using exact-match as the metric.

FLAN Instruction Tuning. A popular application of LLMs has been instruction-following, and chatting capabilities. To test our model's quality on this front, we finetune our models on the FLANv2 dataset (Longpre et al., 2023a), and test the instruction-tuned models' performance from four fronts:

- 5-shot MMLU (Hendrycks et al., 2020): a popular benchmark consiting of exam questions from 57 tasks.
- 3-shot Big Bench Hard (BBH) (Srivastava et al., 2022): a popular set of 23 hardest tasks from big bench.
- Reasoning: macro-average 8-shot performance on GSM8k (Cobbe et al., 2021), SVAMP (Patel et al., 2021), ASDIV (Miao et al., 2021), and StrategyQA (Geva et al., 2021) benchmarks.
- QA: macro-average 0-shot performance on UnifiedQA (Khashabi et al., 2020), BoolQ (Clark et al., 2019), Arc-Easy and Arc-Challenge (Clark et al., 2018) benchmarks.
- Average: macro-average of all the four benchmarking suites listed above: MMLU, BBH, Reasoning, and Q/A.

Please note that all of our reported numbers are based on *single checkpoint* evaluations, *i.e.*, we first select the best checkpoint during FLAN finetuning using the *average* performance on all tasks, and report the individual task performance on that checkpoint.

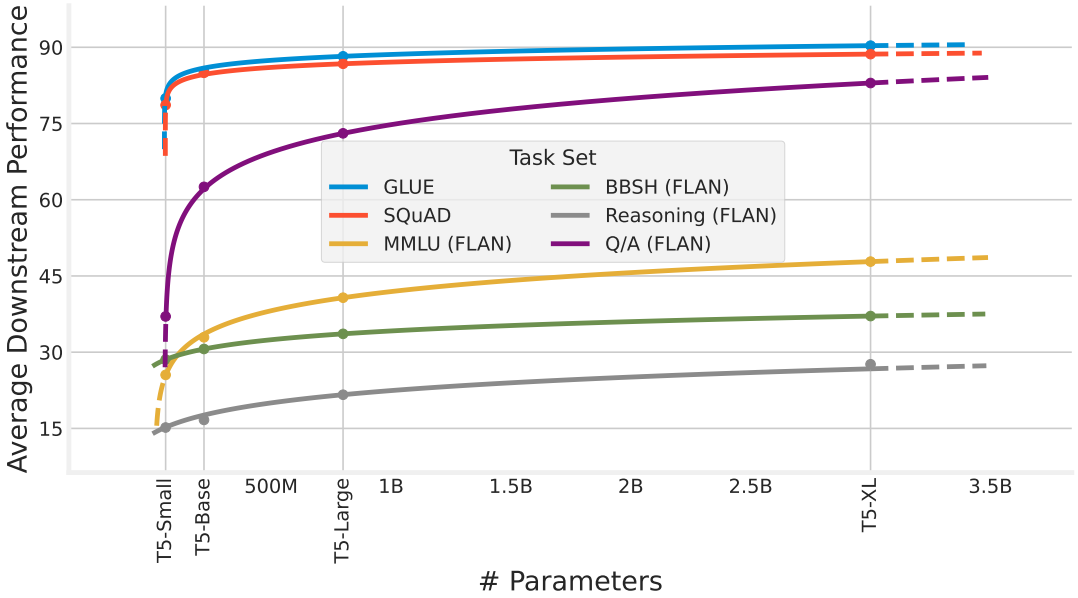

Figure 2.5. Empirical scaling laws for T5-models trained on the entire C4 dataset for various downstream tasks.

2.4.5 Effective Model Size

In addition to these individual tasks, to compare a *normalized average performance improvement* over all downstream evaluations, we devise a metric called "Effective Model Size." Inspired by the LLM scaling-law literature (Hoffmann et al., 2022; Muennighoff et al., 2023), the **Effective Model Size** metric measures the effective model size by extrapolating on a parametric fit of the "number of parameters *vs.* downstream eval" trend, averaged over various downstream tasks.

It is challenging to concisely summarize performance using a single measure, primarily because our evaluation consists of 111 individual tasks, all of which respond at different rates to data and model optimizations. To provide a holistic view of performance, we fit a parametric model, *a.k.a* scaling law (Hoffmann et al., 2022; Muennighoff et al., 2023), of the "model-size $\leftrightarrow$ quality" curve for the original T5 models (*i.e.*, trained on the full C4 dataset) over various downstream tasks.

More specifically, we fit functions of the following form, for each downstream task

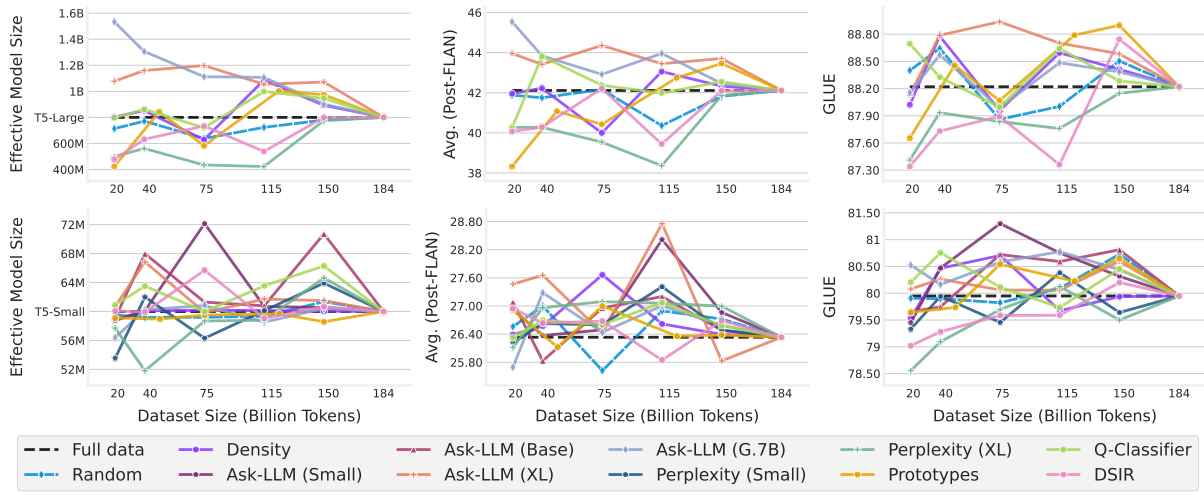

Figure 2.6. Tradeoff between data quantity (number of unique tokens in the sampled dataset) and model quality for (top) T5-Large and (bottom) T5-Small pre-training. Each point corresponds to a converged pre-training run over a sub-sample.

separately:

$$\text{ModelSize} = A + \exp(B + \text{EvalPerformance} * C) \ , \tag{2.1}$$

and use `scipy.optimize.curve_fit` to estimate the A, B, C parameters based on the evaluations of T5-{Small, Base, Large, XL}. See Figure 2.5 for a visual interpretation of the parametric models we fit.

Finally, given the performance of a model trained on downsampled data, the **effective model size** is defined as the predicted model size by plugging in the observed downstream performance into the parametric scaling law estimated via Equation (2.1), taking a median over various downstream evaluation tasks listed in Figure 2.5. To summarize intuitively, this metric is a principled estimate to the following question:

> *"If our ablations (in this chapter, data sampling) lead to x performance, what sized LLM should I have trained in the original setting (in this chapter, the full dataset) to achieve the same x performance?"*

Table 2.1. Comparison of sampling algorithms at a fixed sample size. For each sampling strategy, we sample the dataset to X% of the original size and pre-train T5-Small and T5-Large for 524B tokens. This table is a cross-section of Figure 2.6 but with more metrics.

LLM	Training config.		Effective Model Size	Downstream tasks				FLAN Instruction Tuning			
	Sampler	# Tokens		GLUE	Super GLUE	CNN/ DM	SQuAD	MMLU	BBH	Reasoning	QA
T5-Small	—	184B	60M	79.9	58.6	**18.6**	78.6	25.5	28.5	15.2	37.0
T5-Small	Random	36B	59.2M	79.9	58.3	**18.6**	78.1	26.9	27.8	15.2	**38.1**
T5-Small	Density	36B	60.1M	80.5	59.7	18.5	78.5	28.1	**30.3**	14.5	33.4
T5-Small	Prototypes	46B	58.9M	79.7	58.8	18.5	78.0	26.8	27.7	15.7	34.2
T5-Small	Perplexity (Small)	36B	62.0M	79.9	58.5	18.5	77.5	28.2	28.2	15.0	35.1
T5-Small	DSIR	36B	59.9M	79.3	61.0	18.5	78.9	26.9	28.0	15.5	36.1
T5-Small	Q-Classifier	36B	63.5M	**80.8**	**62.7**	**18.6**	**79.6**	27.1	28.2	15.0	36.4
T5-Small	Ask-LLM (XL)	36B	**66.9M**	80.3	59.8	**18.6**	79.1	**29.9**	28.5	**15.8**	36.4
T5-Large	—	184B	800M	88.2	82.5	20.8	86.7	40.7	33.6	21.6	73.0
T5-Large	Random	18B	713M	88.4	82.3	20.8	85.9	41.8	33.6	20.6	71.5
T5-Large	Density	18B	802M	88.0	80.5	**20.9**	86.9	42.6	35.5	19.1	70.6
T5-Large	Prototypes	18B	423M	87.7	80.5	20.4	86.6	36.7	33.0	17.6	66.0
T5-Large	Perplexity (Small)	18B	301M	87.6	80.2	20.5	85.2	36.8	33.8	17.7	60.9
T5-Large	DSIR	18B	476M	87.3	81.7	20.7	85.4	39.8	33.3	22.2	65.0
T5-Large	Q-Classifier	18B	797M	**88.7**	**83.6**	20.8	87.7	40.5	35.0	20.2	65.4
T5-Large	Ask-LLM (G.7B)	18B	**1.5B**	88.2	82.5	20.8	87.8	**44.2**	**37.1**	**22.7**	**78.2**

2.5 Experiments

2.5.1 Does reasoning improve data efficiency?

Figure 2.6 shows that ASK-LLM trains 800M models to an equivalent performance, as if we were to train 1.5B models on the original C4 dataset. ASK-LLM consistently outperforms perplexity filtering (and coverage-maximizing baselines), despite having access to a scoring model of the same model capacity (XL). Similar findings hold true for training efficiency (Figure 2.7). ASK-LLM converges faster than perplexity filters, both in terms of the average (expected final performance over all proxy model sizes) and pointwise for the best configuration (Small and XL for training T5-Small and T5-Large).

Figure 2.9 further demonstrates that prompting adds critical information to the sampler not present in perplexity: ASK-LLM scores show *no correlation* with the perplexity scores.

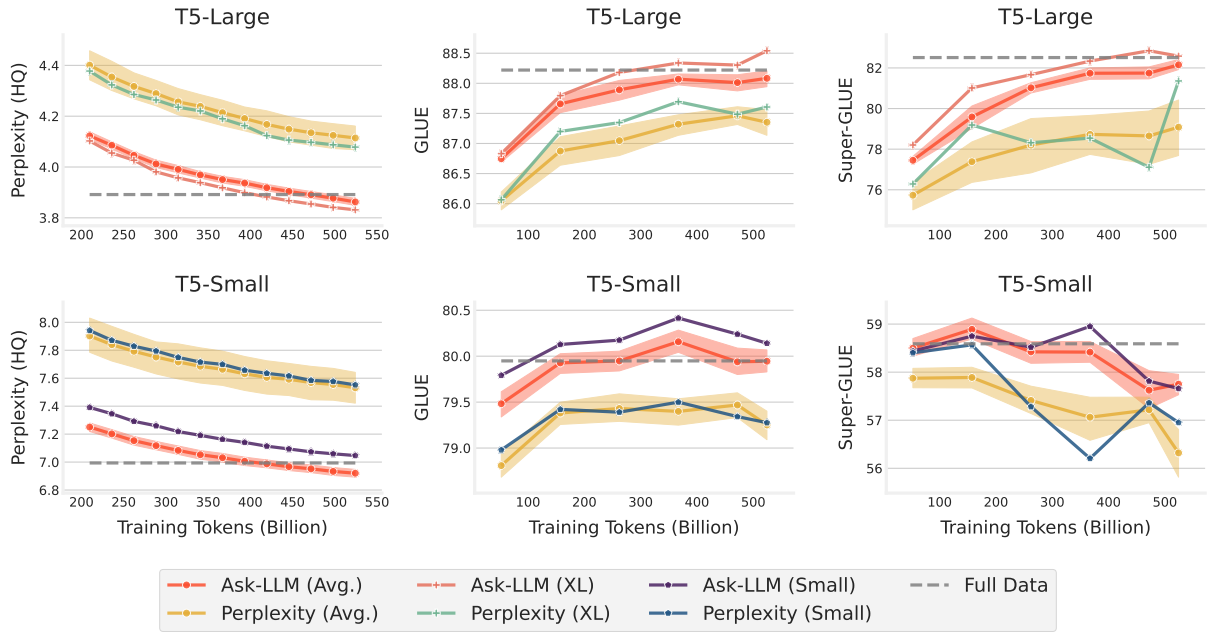

Figure 2.7. Training efficiency comparison between two quality-score based samplers: ASK-LLM and Perplexity filtering. ASK-LLM (Avg.) and Perplexity filtering (Avg.) represent the training run *averaged* across (i) proxy model sizes, *i.e.*, T5-{Small, Base, Large, XL, XXL}; and (ii) sampling ratios, *i.e.*, {10, 20, 40, 60, 80}%. The training runs for ASK-LLM and perplexity filtering with T5-{Small, XL} specifically are averaged only over the sampling ratios. Each point in this plot is the (averaged) performance of an intermediate checkpoint during the course of training on sampled data.

Based on this clear behavioral difference, we conclude that reasoning and context are crucial ingredients. We expect prompting techniques such as chain-of-thought reasoning (Wei et al., 2022) to further drive performance.

2.5.2 When are expensive quality scores justified?

Observing the effective model size while training T5-Large, Figure 2.6 suggests that other samplers start performing well only at larger sample ratios ($\geq 60\%$), with performance very close to ASK-LLM. On the other hand, at smaller sampling ratios, ASK-LLM tends to significantly outperform both coverage-based samplers, as well as cheaper alternatives for data-quality scoring like Q-Classifier and DSIR (Section 2.4.3). Hence, the higher costs of LLM-based filters are most justified in two scenarios: (i) improving full-data performance, where quality filtering by

removing the lowest-quality data is the main way to push the upper limit of model performance; or (ii) in the low-data regime, where keeping only the highest-quality data drives the most model performance compared to other sampling strategies.

We also observe that random sampling is a strong baseline, aligning with recent observations in the literature. Guo et al. (2022a) found that only three methods outperformed random sampling in a computer vision benchmark of 15 algorithms. Ayed and Hayou (2023a) prove the existence of adversarial problem instances where score-based sampling cannot outperform random sampling. These results only serve to highlight the significance of ASK-LLM's gains.

2.5.3 Effect of quality-scoring model capacity

Figure 2.8 demonstrates a clear scaling trend for ASK-LLM's quality-scoring model: larger scoring models are increasingly beneficial as the scale of the to-be-trained LLM increases. Perplexity filters do not seem to exhibit such trends. The strongly consistent scaling for ASK-LLM also suggests an interesting performance-recipe: to improve downstream data-efficiency, use better quality-scoring models. Creating better quality scorers for ASK-LLM (via fine-tuning, chain-of-thought prompting, more capable scoring models, *etc.*) is thus an exciting direction for future work.

2.5.4 Do samplers prioritize different examples?

To understand whether different algorithms prioritize different examples, we sorted examples by score and computed the Kendall Tau rank correlation between samplers (Figure 2.9). We find that samplers differ in significant and interesting ways. For example, the "T5-Large" row shows that (i) T5-Large outputs perplexity scores similar to T5-Small early in training, but becomes progressively more nuanced on the path from 20k to 700k training steps, and (ii) perplexity and ASK-LLM select for wildly different criteria, with almost no ranking correlation.

DENSITY prioritizes coverage over de-noising, maintaining the in-distribution test perplexity better than any other strategy (Figure 2.6), other than random sampling in which case

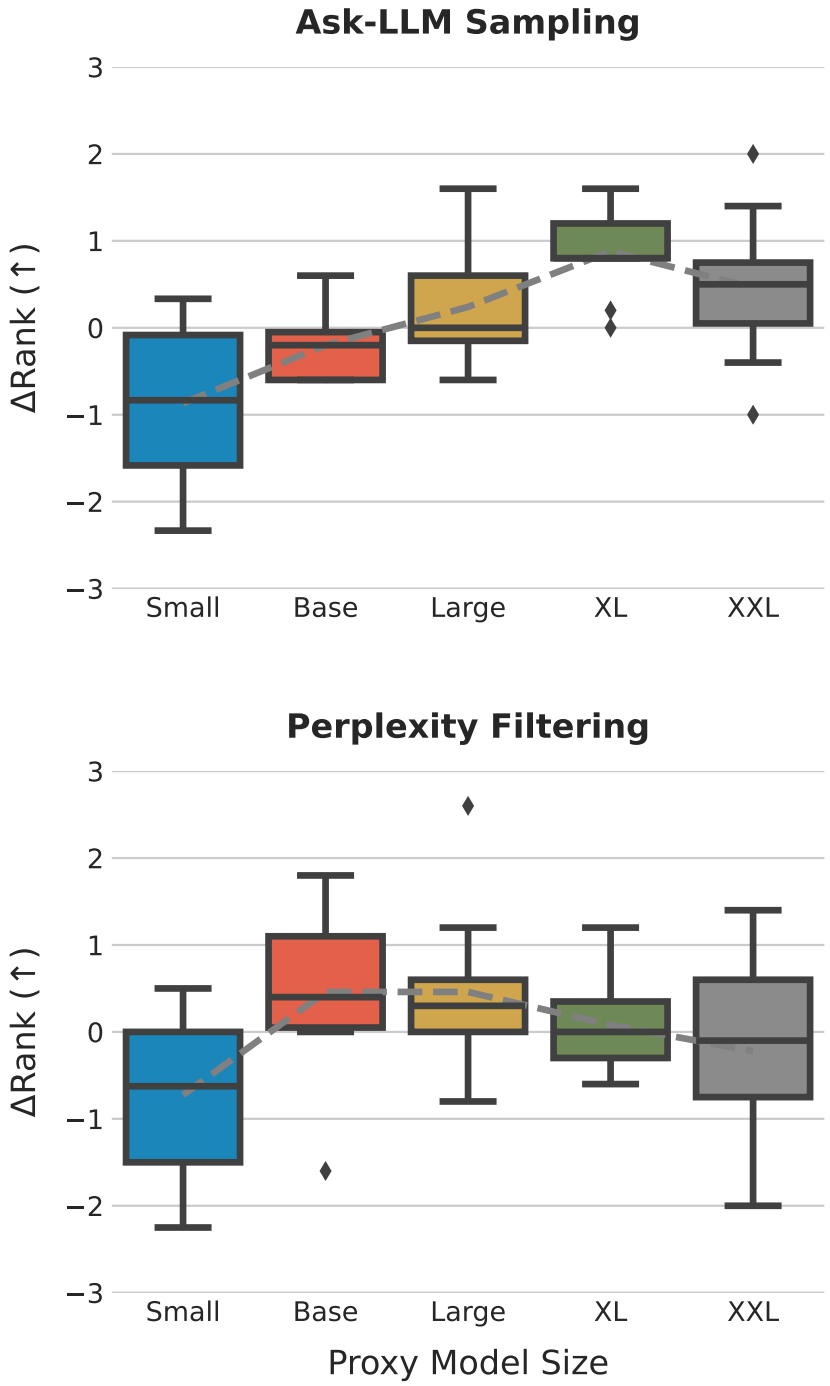

Figure 2.8. We investigate the change in *ranking* of quality-scoring models when pre-training different LLMs. A positive ΔRank indicates that the scorer's task-averaged rank within {Small, Base, Large, XL, XXL} increased when training T5-Large *vs.* T5-Small.

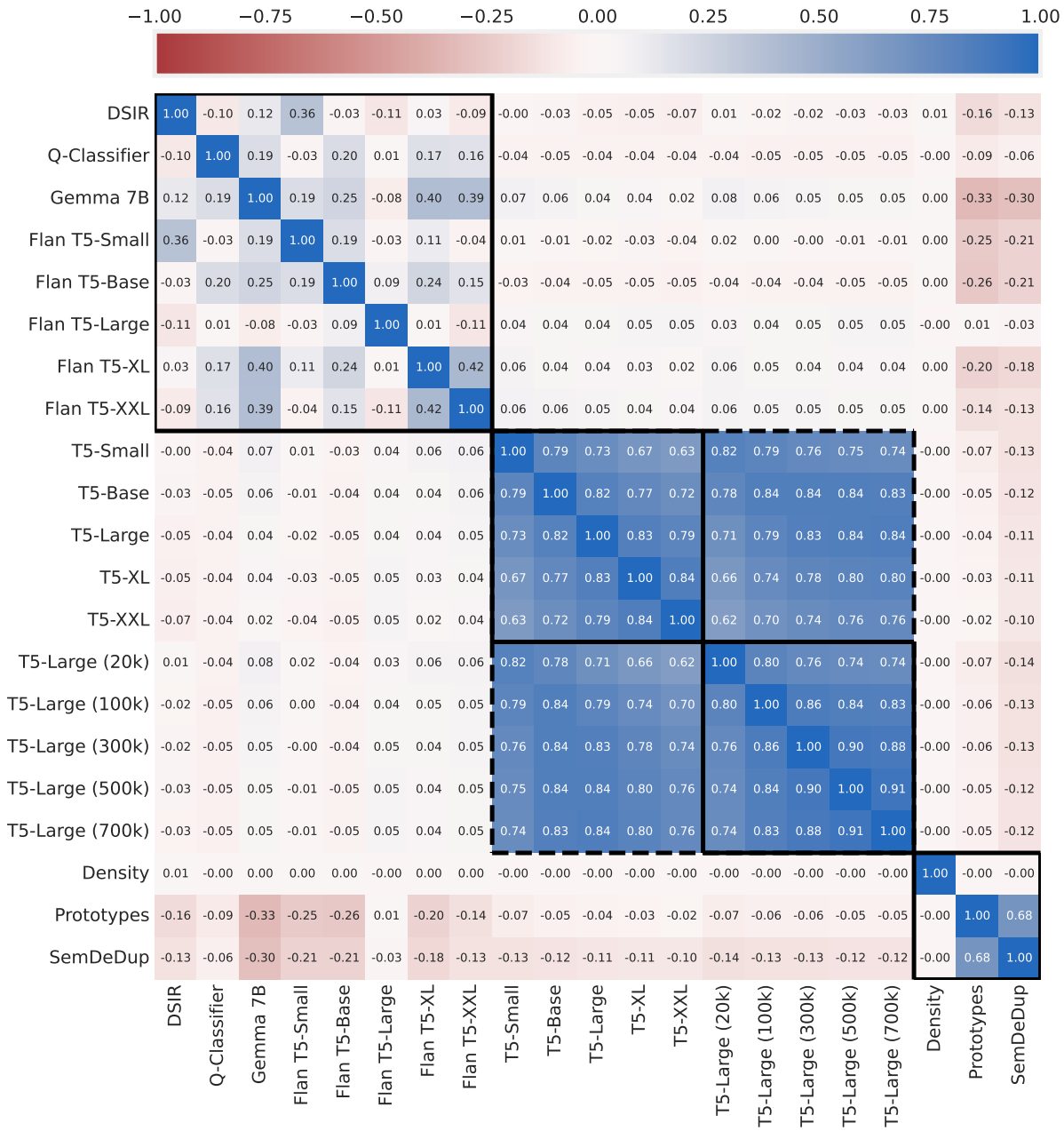

Figure 2.9. Kendall's Tau correlation amongst the scores from quality filters (first 8), perplexity filters (next 10), and coverage-based samplers (last 3) over 500k randomly selected training samples.

the validation data is exactly in-distribution *w.r.t.* the training data. This suggests that coverage sampling preserves the objective function, in contrast with other methods that preferentially select for quality in addition to diversity.

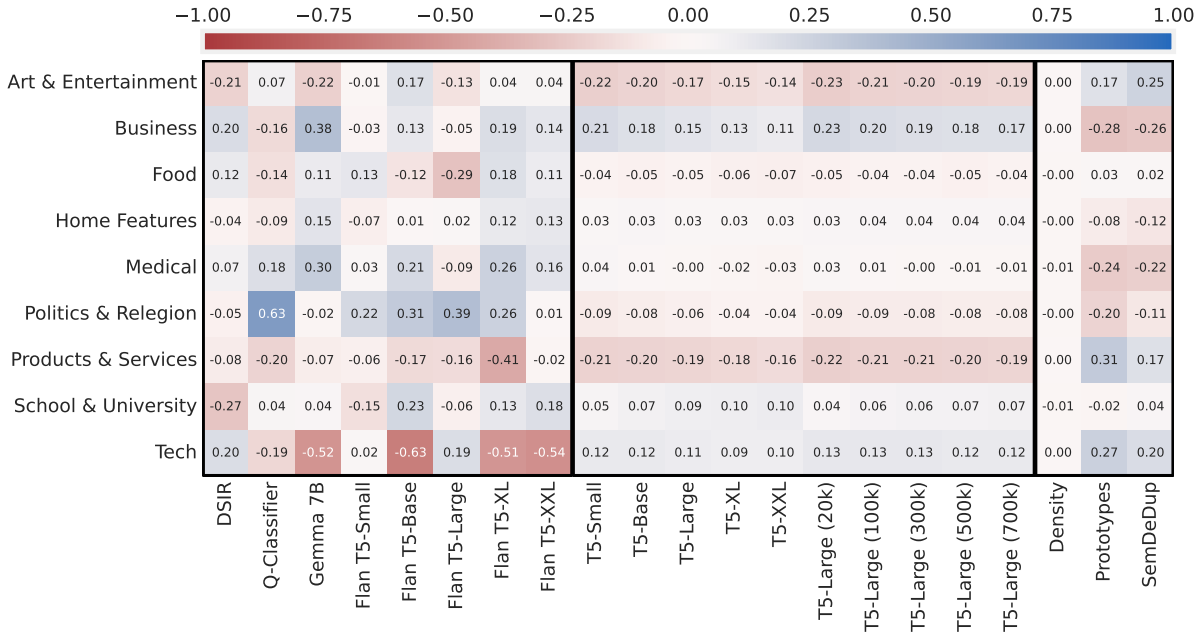

Figure 2.10. Estimated topic affinity for quality filters (first 8), perplexity filters (next 10), and coverage-based samplers (last 3) over 500k randomly selected training samples.A higher score represents more affinity. All effects significant at the $p < 0.01$ level.

2.5.5 Analysis of Different Samplers' Affinity to Different Topics

Since the different sampling strategies explored in this chapter operate with different implicit biases, we try to understand if certain samplers exhibit more affinity to certain topics in the data compared to others. To visualize this phenomenon, we conduct the following procedure:

1. Load a random sample of 500k datapoints, along with their respective data-quality scores.

2. Perform topic-modeling (via LDA) on the 500k datapoints with 9 topics.

3. Manually inspect the most common word associations in each of the 9 topics, and label a "high-level description" for each topic.

4. Assign each of the 500k datapoints to the LDA topic with the highest likelihood and analyze the differences between the distribution of scores within each topic and the global score distribution.

5. We conducted a one-way ANOVA, one-vs-rest style, to determine whether the averages were statistically significant. Because $N = 500$k, all effects were significant at the $p < 0.01$ level. We measure the effect size using Cohen's d and report results in Figure 2.10.

From the topic affinity analysis in Figure 2.10, we can observe a few interesting common trends:

- The perplexity filters have relatively low variance in their scores, indicating a much less biased sampling. This is expected, because perplexity filtering primarily biases toward "well-written text" which is relatively task/topic agnostic.
- The quality-based samplers (ASK-LLM, DSIR, Q-Classifier) exhibit a much stronger variance in their scores, with a common liking towards business, political, and religious content; and a common disliking towards tech, art, and entertainment content.
- Consistent with the score correlations in Figure 2.9, the prototypes and SemDeDup samplers exhibit inverse correlation with most other samplers when comparing topic affinity too.
- Density sampling, as expected, exhibits no special affinity to any particular topic because it's objective is to only maximize coverage.

2.6 Qualitative Analysis

In this section we look at some qualitative training samples, sorted according to various criteria of data-quality scores. Along with the textual content of each training sample, we also list the estimated data-quality percentile for ASK-LLM and perplexity filtering samplers, *i.e.*, the percentile of the given data-point's quality score amongst the entire training set. A high percentile represents that the sampler estimates this training sample to have higher quality compared to other training samples in the dataset. We manually don't include any NSFW examples to the best of our knowledge.

2.6.1 High-quality Samples Identified by ASK-LLM

We look at the training samples that *all* ASK-LLM scoring models, on average, think are good (*i.e.*, have a high percentile). To the best of our understanding, the overarching conclusions we make by observing these qualitative samples are:

- ASK-LLM doesn't seem to have any length bias for good examples.
- ASK-LLM can accurately tag high-quality training samples that contain a lot of proper nouns and named entities. Perplexity filtering gets these kind of samples wrong.
- Even looking at this slice of only the highest-quality data tagged by ASK-LLM, perplexity filtering scores don't seem to correlate well with ASK-LLM scores as suggested by Figure 2.9.

Example 1: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					**Perplexity Filtering**				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
93.33%	88.21%	88.11%	100.0%	99.99%	50.29%	30.34%	32.56%	31.61%	25.62%

What constitutes overtime for a part-time employee? Question: What is overtime for a part-time employee? Overtime for a part-time employee is time that is beyond the part-time employee's ordinary hours of work or outside the agreed number of hours of work, as specified in their employment contract.

Example 2: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					Perplexity Filtering				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
87.08%	89.33%	95.26%	99.13%	99.94%	98.09%	97.52%	98.83%	97.39%	97.38%

Bonelli, N.; Giordano, S.; Procissi, G. Enif-Lang: A Specialized Language for Programming Network Functions on Commodity Hardware. J. Sens. Actuator Netw. 2018, 7, 34. Bonelli N, Giordano S, Procissi G. Enif-Lang: A Specialized Language for Programming Network Functions on Commodity Hardware. Journal of Sensor and Actuator Networks. 2018; 7(3):34. Bonelli, Nicola; Giordano, Stefano; Procissi, Gregorio. 2018. "Enif-Lang: A Specialized Language for Programming Network Functions on Commodity Hardware." J. Sens. Actuator Netw. 7, no. 3: 34.

Example 3: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					Perplexity Filtering				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
96.41%	86.03%	97.38%	95.91%	90.8%	34.7%	44.8%	56.87%	60.15%	77.25%

"What is your number one secret to productivity?" In recording their responses, Kruse came across some fascinating suggestions. What follows are some of my favorites. They focus on minutes, not hours. Most people default to hour and half-hour blocks on their calendar; highly successful people know that there are 1,440 minutes in every day and that there is nothing more valuable than time. Money can be lost and made again, but time spent can never be reclaimed. As legendary Olympic gymnast Shannon Miller told Kevin, "To this day, I keep a schedule that is almost minute by minute." You must master your minutes to master your life.

...

Energy is everything. You can't make more minutes in the day, but you can increase your energy to increase your attention, focus, and productivity. Highly successful people don't skip meals, sleep, or breaks in the pursuit of more, more, more. Instead, they view food as fuel, sleep as recovery, and breaks as opportunities to recharge in order to get even more done. Author of #1 bestselling book, Emotional Intelligence 2.0, and president of TalentSmart, world's leading provider of emotional intelligence.

2.6.2 Low-quality Samples Identified by ASK-LLM

We look at the training samples that *all* ASK-LLM scoring models, on average, think are bad (*i.e.*, have a low percentile). To the best of our understanding, the overarching conclusions we make by observing these qualitative samples are:

- ASK-LLM doesn't seem to have any length bias for bad examples.
- ASK-LLM filters hateful or toxic examples that might hurt LLM training.
- ASK-LLM rejects non-contextual samples, *e.g.*, having only questions with no answers, repeated non-sensical content, *etc.* Notably, perplexity filtering performs bad in these cases, as these low quality examples tend to have a low perplexity score.

Example 4: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					**Perplexity Filtering**				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
5.41%	3.86%	0.49%	0.8%	6.24%	62.97%	75.91%	86.3%	85.26%	88.11%

You were a good daughter the first day or two. Now, you are only showing the worst sides of yourself. I can only be sad and disappointed in you.

Example 5: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					**Perplexity Filtering**				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
1.08%	0.41%	6.16%	2.46%	1.44%	35.97%	24.13%	31.46%	51.15%	38.19%

Kids can help you enrich your life? Be a better person? Learn to think about someone else? Apparently whoever said these things has never had children because from everything we have seen and experienced, kids are flat out horrible. College can't come fast enough.

Example 6: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					Perplexity Filtering				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
1.89%	3.58%	3.11%	6.02%	0.09%	18.09%	22.8%	25.61%	19.14%	47.01%

EventsThis is how you can go ice skating with real penguinsGrab your tickets before they sell out! Can you spot anyone you know in these fun pics? EventsHow do I get tickets for Wimbledon 2018?

2.6.3 Increasing-quality Samples Identified by ASK-LLM

We look at the training samples that ASK-LLM scoring models *disagree on* as we go from Flan-T5-Small → Flan-T5-XXL. Specifically, we look at training samples that Flan-T5-Small thinks are of low quality, whereas Flan-T5-XXL thinks otherwise. To the best of our understanding, our overarching conclusions by observing these qualitative samples are:

- Larger scoring models in ASK-LLM are able to identify training samples containing *tail-end* of knowledge, *e.g.*, rare world-events, rare named entities, *etc.*
- The increasing quality trend going from Flan-T5-Small → Flan-T5-XXL isn't correlated with the quality scoring model size in perplexity filtering.

Example 7: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					Perplexity Filtering				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
7.67%	30.45%	57.41%	78.17%	97.41%	15.56%	31.02%	24.14%	50.59%	49.64%

The historic city of Manchester now features one of the most interesting public art installations that art lovers have ever witnessed. Design studio, Acrylicize installed five giant lamps in Piccadilly Place that represent the many historic periods that the city has gone through, including; Art Deco, Art Nouveau, Victorian, mid-century, and contemporary. The installation is without any doubt, a great piece of art but unlike other artworks, these are absolutely functional as well. Each lamp provides the many visitors with seating, shelter, light and even heat in the winters. The admirers can also witness the historic stories of Manchester via graphic illustrations on the lamps.

Example 8: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					Perplexity Filtering				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
10.48%	31.26%	54.17%	84.17%	97.93%	30.52%	39.49%	35.79%	30.89%	25.39%

The Cokin Yellow and Pink Center Spot filter has a clear center and diffused yellow and pink edges. Theses diffused edges will be produce blur while leaving the center sharp. The filter effect is directly influenced by the f-stop and the focal length. A lens shot at f/1.4 will see a greater blurring effect than f/8.0 and a 85mm lens will see more blur than a 28mm. Additionally, a longer focal length lens will visually increase the size of the center spot area because it sees less of the filter area.

Example 9: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					Perplexity Filtering				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
0.98%	24.84%	53.36%	88.98%	98.18%	2.3%	1.48%	2.03%	2.1%	3.07%

The Disknet is a networking solution which uses the external floppy drive port of the Amiga. It uses the same coax cabling as 10Base2 Ethernet (RG-58U/50Ohm) but is NOT compatible and is capable of transferring at around 45k/sec. The Disknet may be the same device as the AmigaLink, but this has not been confirmed.

2.6.4 Decreasing-quality Samples Identified by ASK-LLM

We look at the training samples that ASK-LLM scoring models *disagree on* as we go from Flan-T5-Small → Flan-T5-XXL. Specifically, we look at training samples that Flan-T5-XXL thinks are of low quality, whereas Flan-T5-Small thinks otherwise. To the best of our understanding, our overarching conclusions by observing these qualitative samples are:

- Smaller quality-scoring models sometimes mislabel non-informative training samples, that contain, *e.g.*, non-informative content, or repeated content.
- The decreasing quality trend going from Flan-T5-Small → Flan-T5-XXL isn't correlated with the quality scoring model size in perplexity filtering.

Example 10: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					Perplexity Filtering				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
64.05%	46.39%	35.92%	25.29%	9.63%	4.3%	10.21%	3.47%	3.34%	3.35%

one filled with goodwill and cheer. who have supported me thru the year. I wouldn't be changing careers. instead of on strange people's rears. Wishes You a Healthy, Happy Holidays! Ah, how the mighty have fallen! And a Merry fave to you ... and a happy new rear. From one Xmas humor story to another, enjoyed this! Thanks Jack & Susan! Doug, I checked him out–wonderful stuff! Will pass along the good word. Fun and funny–as always! Thanks for the cheer! I can only fave this once, but I've looked at it repeatedly over what has been a bizarre week– and each time you've given me a laugh. That's a gift Bob and I'm grateful! Best of holidays to you and a great New Year!

Example 11: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					Perplexity Filtering				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
82.64%	75.2%	63.2%	29.51%	8.94%	78.34%	82.07%	91.01%	87.78%	88.02%

You can now configure the minimum TLS protocol level for client connections and connections to other servers. Refer to the following page for more information: Advanced TLS. You can now set an Integrated Capture Point (ICP) to stopped mode by changing the state of the corresponding configuration object to disabled; changing the state to enabled restarts the inbound cycle of the ICP. You can now set the minimum TLS protocol level for the Web Service Capture Point by configuring the option <sec-protocol> in the section <settings> of the Capture Point object.

...

Support for the following databases. See the Supported Operating Environment: eServices page for more detailed information and a list of all supported databases. No special procedure is required to upgrade to release 8.5.201.05. Retrieved from "https://docs.genesys.com/Documentation:RN:mm-ixn-svr85rn:mm-ixn-svr8520105:8.5.x (2019-04-21 22:59:48)" This page was last modified on November 8, 2018, at 08:48.

Example 12: Estimated Data-Quality (Percentile – Higher is better)

ASK-LLM					Perplexity Filtering				
Small	Base	Large	XL	XXL	Small	Base	Large	XL	XXL
62.21%	54.71%	35.73%	22.64%	6.76%	64.82%	85.95%	94.65%	93.35%	85.29%

are willing to provide you with perfect services and striding for Display Stand For Boutique , Display Stand for Boutique , Display Stand for Phone , Our product quality is one of the major concerns and has been produced to meet the customer's standards. "Customer services and relationship" is another important area which we understand good communication and relationships with our customers is the most significant power to run it as a long term business. "We have quite a few great team customers very good at internet marketing, QC, and dealing with kinds of troublesome trouble while in the output approach for Display Stand For Boutique , Display Stand for Boutique , Display Stand for Phone , We set a strict quality control system. We've got return and exchange policy and you can exchange within 7 days after receive the wigs if it is in new station and we service repairing free for our solutions. You should feel free to contact us for further information and we are going to give you competitive price list then.

2.7 Discussion

Amortized scoring. The ASK-LLM and perplexity scorers require considerable computation—one LLM inference call for every training sample—which is concerning from both a carbon-emissions and cost perspective (Strubell et al., 2019b). However, we argue that the scoring costs are *amortized over many pre-training runs*, which together cost significantly more than the ASK-LLM inference calls (Luccioni et al., 2023). In practical systems, cheaper samplers / scoring models can also pre-filter examples for our more expensive scorers. While LLM pre-training is often thought of as a one-time cost, this has historically not been the case. We therefore view quality scores as a long-term investment.

LLM-Based Data Refinement. Recursively training on model-generated data causes degradation in both diffusion models and LLMs, inciting concerns about whether the internet will remain a viable source of training data (Shumailov et al., 2023; Alemohammad et al., 2023;

Briesch et al., 2023). It is therefore somewhat surprising that LLMs are so effective at deciding which training data to consume. Our ASK-LLM results raise important questions about whether LLM-based filters can function as an intervention in the self-consumption loop, allowing LLMs to self-improve.

2.8 Conclusion

We studied the performance of sampling algorithms that select high-quality data through highly-capable proxies and maximize coverage through embedding similarity. Our experiments reveal that LLM-based quality filtering yields a Parteo optimal efficiency tradeoff between data quantity and model quality, with important implications for training cost, self-improvement, and LLM training data curation.

Chapter 2, in part, is currently being prepared for submission for publication of the material "How to Train Data-Efficient LLMs." by Noveen Sachdeva, Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Lichan Hong, Ed Chi, James Caverlee, Julian McAuley, and Derek Zhiyuan Cheng. 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Data Distillation for Recommender Systems

We leverage the Neural Tangent Kernel and its equivalence to training infinitely-wide neural networks to devise ∞-AE: an autoencoder with infinitely-wide bottleneck layers. The outcome is a highly expressive yet simplistic recommendation model with a single hyper-parameter and a closed-form solution.

Leveraging ∞-AE's simplicity, we also develop DISTILL-CF for synthesizing tiny, high-fidelity data summaries which distill the most important knowledge from the extremely large and sparse user-item interaction matrix for efficient and accurate subsequent data-usage like model training, inference, architecture search, *etc.* This takes a data-centric approach to recommendation, where we aim to improve the quality of logged user-feedback data for subsequent modeling, independent of the learning algorithm. We particularly utilize the concept of differentiable Gumbel-sampling to handle the inherent data heterogeneity, sparsity, and semi-structuredness, while being scalable to datasets with hundreds of millions of user-item interactions.

Both of our proposed approaches significantly outperform their respective state-of-the-art and when used together, we observe $96-105\%$ of ∞-AE's performance on the full dataset with as little as 0.1% of the original dataset size, leading us to explore the counter-intuitive question: *Is more data what you need for better recommendation?*

3.1 Introduction

The Neural Tangent Kernel (NTK) has recently advanced the theoretical understanding of how neural networks learn (Arora et al., 2019; Jacot et al., 2018). Notably, performing Kernelized Ridge Regression (KRR) with the NTK has been shown to be equivalent to training infinitely-wide neural networks for an infinite number of SGD steps. Owing to KRR's closed-form solution, these networks can be trained in a fast and efficient manner whilst not sacrificing expressivity. While the application of infinite neural networks is being explored in various learning problems (Radhakrishnan et al., 2022; Arora et al., 2020), detailed comparative analyses demonstrate that deep, finite-width networks tend to perform significantly better than their infinite-width counterparts for a variety of standard computer-vision tasks (Lee et al., 2020).

On the contrary, for recommendation tasks, there always has been a debate of linear *vs.* non-linear networks (Kong et al., 2022; Xu et al., 2021), along with the importance of increasing the width *vs.* depth of the network (Cheng et al., 2016; Naumov et al., 2019). At a high level, the general conclusion is that a well-tuned, wide and linear network can outperform shallow and deep neural networks for recommendation (Rendle et al., 2020). We extend this debate by introducing our model ∞-AE, an autoencoder with infinitely wide bottleneck layers and examine its behavior on the recommendation task. Our evaluation demonstrates significantly improved results over state-of-the-art (SoTA) models across various datasets and evaluation metrics.

A rising challenge in recommender systems research has been the increased cost of training models on massive datasets which can involve billions of user-item interaction logs, in terms of time, computational resources, as well as the downstream carbon footprint. Moreover, despite anonymization efforts, privacy risks have been associated with publicly released user data (Narayanan and Shmatikov, 2008). To this end, we further explore recommendation from a data-centric viewpoint (Strickland, 2022), which we loosely define as:

Definition 3.1.1. (Data-centric AI) Systematic methods for improving the collected data's quality, thereby shifting the focus from merely acquiring large quantities of data; implicitly

helping in a learning algorithm's generalization, scalability, and eco-sustainability.

Previous work on data-centric techniques generally involve constructing terse data summaries of large datasets, and has focused on domains with continuous, real-valued features such as images (Zhao and Bilen, 2021; Nguyen et al., 2021b), which are arguably more amenable to data optimization approaches. Due to the heterogeneity, sparsity, and semi-structuredness of recommendation data, such methods are not directly applicable. Common approaches for scaling down such recommendation datasets typically include heuristics such as random, head-user, or k-core sampling. More principled approaches include coreset construction (Sachdeva et al., 2022c) that focus on optimizing for *picking* the set of data-points that are the most representative from a given dataset, and are generally shown to out-perform various heuristic sampling strategies. However, *synthesizing* informative summaries for recommendation datasets still remains a challenge.

Consequently, we propose DISTILL-CF, a data distillation framework for collaborative filtering (CF) datasets that utilizes ∞-AE in a bilevel optimization objective to create highly compressed, informative, and generic data summaries. DISTILL-CF employs efficient multi-step differentiable Gumbel-sampling (Jang et al., 2017) at each step of the optimization to encompass the heterogeneity, sparsity, and semi-structuredness of recommendation data. We further provide an analysis of the denoising effect observed when training the model on the synthesized versus the full dataset. To summarize, *in this chapter*, we make the following contributions:

- We develop ∞-AE: an infinite-width autoencoder for recommendation, that aims to reconstruct the incomplete preferences in a user's item consumption sequence. We demonstrate its efficacy on four datasets, and demonstrate that ∞-AE outperforms complicated SoTA models with only a single fully-connected layer, closed-form optimization, and a single hyper-parameter. We believe our work to be the first to demonstrate that an infinite-width network can outperform their finite-width SoTA counterparts for practical scenarios like recommendation.

- We subsequently develop DISTILL-CF: a novel data distillation framework that can synthesize

tiny yet accurate data summaries for a variety of modeling applications. We empirically demonstrate similar performance of models trained on the full dataset versus training the same models on $2-3$ orders smaller data summaries synthesized by DISTILL-CF. Notably, DISTILL-CF and ∞-AE are complementary for each other's practicality, as ∞-AE's closed-form solution enables DISTILL-CF to scale to datasets with hundreds of millions of interactions; whereas, DISTILL-CF's succinct data summaries help improving ∞-AE's restrictive training complexity, and achieving SoTA performance when trained on these tiny data summaries.

- Finally, we also note that DISTILL-CF has a strong data denoising effect, validated with a counter-intuitive observation — if there is noise in the original data, models trained on *less* data synthesized by DISTILL-CF can be better than the same model trained on the *entire* original dataset. Such observations, along with the strong data compression results attained by DISTILL-CF, reinforce our data-centric viewpoint to recommendation, encouraging the community to think more about the quality of collected data, rather than its quantity.

3.2 Related Work

3.2.1 Autoencoders for recommendation

Recent approaches to implicit feedback recommendation involve building models that can re-construct an incomplete user preference list using autoencoders (Liang et al., 2018; Steck, 2019; Sachdeva et al., 2019; Ma et al., 2019). The CDAE model (Wu et al., 2016) first introduced this idea and used a standard denoising autoencoder for recommending new items to users. MVAE (Liang et al., 2018) later extended this idea to use variational autoencoders, and provided a principled approach to perform variational inference for this model architecture. More recently, EASE (Steck, 2019) proposed using a shallow autoencoder and estimates only an item-item similarity matrix by performing ordinary least squares regression on the relevance matrix, resulting in closed-form optimization.

3.2.2 Infinite neural networks

The Neural Tangent Kernel (NTK) (Jacot et al., 2018) has gained significant attention because of its equivalence to training infinitely-wide neural networks by performing KRR with the NTK. Recent work also demonstrated the double-descent risk curve (Belkin et al., 2018) that extends the classical regime of train *vs.* test error for overparameterized neural networks, and plays a crucial role in the generalization capabilities of such infinite neural networks. However, despite this emerging promise of utilizing NTK for learning problems, detailed comparative analyses (Novak et al., 2019; Lee et al., 2020; Arora et al., 2019) for computer vision tasks demonstrate that finite-width networks are still significantly more accurate than infinite-width ones. Looking at recommendation systems, (Xu et al., 2021) performed a theoretical comparison between Matrix Factorization (MF) and Neural MF (He et al., 2017) by studying their expressivity in the infinite-width limit, comparing the NTK of both of these algorithms. Notably, their settings involved the typical (user-ID, item-ID) input to the recommendation model, and observed results that were equivalent to a random predictor. (Radhakrishnan et al., 2022) performed a similar study that performed matrix completion using the NTK of a single layer fully-connected neural network, but assumed meaningful feature-priors available beforehand.

3.2.3 Data sampling & distillation

Data sampling is ubiquitous — sampling negatives while contrastive learning (Robinson et al., 2021; Kalantidis et al., 2020), sampling large datasets for fast experimentation (Sachdeva et al., 2022c), sampling for evaluating expensive metrics (Krichene and Rendle, 2020), *etc.* In this chapter, we primarily focus on sampling at the dataset level, principled approaches of which can be categorized into: (1) coreset construction methods that aim to *pick* the most useful datapoints for subsequent model training (Borsos et al., 2020a; Killamsetty et al., 2021; S et al., 2021; Kazemi et al., 2021). These methods typically assume the availability of a submodular set-function $f : \mathbf{V} \mapsto \mathbb{R}_+ \; \forall \; \mathbf{V} \subseteq \mathbf{X}$ for a given dataset $\mathbf{X}$ (see (Bilmes, 2022) for a systematic

review on submodularity), and use this set-function f as a proxy to guide the search for the most informative subset; and (2) dataset distillation: instead of picking the most informative data-points, dataset distillation techniques aim to *synthesize* data-points that can accurately distill the knowledge from the entire dataset into a small, synthetic data summary. Originally proposed in (Wang et al., 2018b), the authors built upon the notions of gradient-based hyper-parameter optimization (Maclaurin et al., 2015a) to synthesize representative images for model training. Subsequently, a series of works (Zhao et al., 2021; Zhao and Bilen, 2021; Nguyen et al., 2021a;b) propose various subtle modifications to the original framework, for improving the sample-complexities of models trained on data synthesized using their algorithms. Note that such distillation techniques focused on continuous data like images, which are trivial to optimize in the original data-distillation framework. More recently, (Jin et al., 2022b) proposed a distillation technique for synthesizing fake graphs, but also assumed to have innate node representations available beforehand, prohibiting their method's application for recommendation data.

3.3 ∞-AE: Infinite-width Autoencoders for Recommendation

3.3.1 Notation

Given a recommendation dataset $\mathcal{D} \coloneqq \{(\text{user}_i, \text{item}_i, \text{relevance}_i)\}_{i=1}^{n}$ consisting of n user-item interactions defined over a set of users $\mathcal{U}$, set of items $\mathcal{I}$, and operating over a binary relevance score (implicit feedback); we aim to best model user preferences for item recommendation. The given dataset $\mathcal{D}$ can also be viewed as an interaction matrix, $X \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ where each entry $X_{u,i}$ either represents the observed relevance for item i by user u or is missing. Note that X is typically extremely sparse, *i.e.*, $n \ll |\mathcal{U}| \times |\mathcal{I}|$. More formally, we define the problem of recommendation as accurate likelihood modeling of $\text{P}(i \mid u, \mathcal{D})\ \forall u \in \mathcal{U},\ \forall i \in \mathcal{I}$.

Algorithm 3. ∞-AE model training

Input: User set $\mathcal{U}$; dataset $\mathbf{X} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$; NTK $\mathbb{K} : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}$; regularization $\lambda \in \mathbb{R}$
Output: Dual parameters $\alpha \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$

procedure FIT($\mathcal{U}, \mathbf{X}, \mathbb{K}$)
  $\mathbf{K} \leftarrow [0]_{|\mathcal{U}|\times|\mathcal{U}|}$ ▷ Zero Initialization
  $\mathbf{K}_{u,v} \leftarrow \mathbb{K}(\mathbf{X}_u, \mathbf{X}_v) \quad \forall u \in \mathcal{U}, v \in \mathcal{U}$
  $\alpha \leftarrow (\mathbf{K} + \lambda I)^{-1} \cdot \mathbf{X}$
  return α

Algorithm 4. ∞-AE inference

Input: User set $\mathcal{U}$; dataset $\mathbf{X} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$; NTK
$\mathbb{K} : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}$; dual params. $\alpha \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$; inference user history $\hat{\mathbf{X}}_u \in \mathbb{R}^{|\mathcal{I}|}$
Output: Prediction $\hat{y} \in \mathbb{R}^{|\mathcal{I}|}$

procedure PREDICT($\mathcal{U}, \mathbf{X}, \hat{\mathbf{X}}_u, \mathbb{K}, \alpha$)
  $\mathbf{K} \leftarrow [0]_{|\mathcal{U}|}$ ▷ Zero Initialization
  $\mathbf{K}_v \leftarrow \mathbb{K}(\hat{\mathbf{X}}_u, \mathbf{X}_v) \quad \forall v \in \mathcal{U}$
  $\hat{y} \leftarrow \text{softmax}(\mathbf{K} \cdot \alpha)$
  return $\hat{y}$

3.3.2 Model

∞-AE takes an autoencoder approach to recommendation, where the all of the bottleneck layers are infinitely-wide. Firstly, to make the original bi-variate problem of which *item* to recommend to which *user* more amenable for autoencoders, we make a simplifying assumption that a user can be represented only by their historic interactions with items, *i.e.*, the much larger set of users lie in the linear span of items. This gives us two kinds of modeling advantages: (1) we no longer have to find a unique latent representation of users; and (2) allows ∞-AE to be trivially applicable for any user not in $\mathcal{U}$. More specifically, for a given user u, we represent it by the sparse, bag-of-words vector of historical interactions with items $X_u \in \mathbb{R}^{|\mathcal{I}|}$, which is simply the u^{th} row in X. We then employ the Neural Tangent Kernel (NTK) (Jacot et al., 2018) of an autoencoder that takes the bag-of-items representation of users as input and aims to reconstruct it. Due to the infinite-width correspondence (Jacot et al., 2018; Arora et al.,

Algorithm 5. Data synthesis using DISTILL-CF

Input: User set $\mathcal{U}$; dataset $\mathbf{X} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$; NTK
$\mathbb{K} : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}$; support user size $\mu \in \mathbb{R}$; gumbel softmax temperature
$\tau \in \mathbb{R}$; reg. const. $\lambda_2 \in \mathbb{R}$; SGD batch-size b, step-size $\eta \in \mathbb{R}$
Output: Synthesized data $\mathbf{X^s} \in \mathbb{R}^{\mu\times|\mathcal{I}|}$

procedure SAMPLE($n, \mathcal{U}, \mathbf{X}$)
  $\mathcal{U}' \sim \mathcal{U}$ ▷ Randomly sample n users from $\mathcal{U}$
  $\mathbf{X}' \leftarrow \mathbf{X_u} \quad \forall u \in \mathcal{U}'$ ▷ Retrieve corresponding rows from $\mathbf{X}$
  return $\mathcal{U}', \mathbf{X}'$
procedure SYNTHESIZE($\mathcal{U}, \mathbf{X}, \mathbb{K}$)
  $\mathcal{U}^s, \mathbf{X^s} \leftarrow$ SAMPLE($\mu, \mathcal{U}, \mathbf{X}$) ▷ Sample support data
  for $steps = 0 \ldots \xi$ **do**
    $\hat{\mathbf{X}}^\mathbf{s} \leftarrow \sigma\left[\sum_{i=1}^{\gamma} \text{gumbel}_\tau(\text{softmax}(\mathbf{X^s}))\right]$
    $\alpha^s \leftarrow$ FIT($\mathcal{U}^s, \hat{\mathbf{X}}^\mathbf{s}, \mathbb{K}$) ▷ Fit ∞-AE on support data
    $\mathcal{U}^b, \mathbf{X^b} \leftarrow$ SAMPLE($b, \mathcal{U}, \mathbf{X}$)
    $\tilde{\mathbf{X}} \leftarrow [0]_{b\times|\mathcal{I}|}$
    $\tilde{\mathbf{X}}_\mathbf{u} \leftarrow$ PREDICT($\mathcal{U}^s, \hat{\mathbf{X}}^\mathbf{s}, \mathbf{X^b_u}, \mathbb{K}, \alpha^s$) $\quad \forall u \sim \mathcal{U}^b$ ▷ Predict for sampled users
    $L \leftarrow \mathbf{X^b} \cdot log(\tilde{\mathbf{X}}) + (1 - \mathbf{X^b}) \cdot log(1 - \tilde{\mathbf{X}}) + \lambda_2 \cdot ||\hat{\mathbf{X}}^\mathbf{s}||_1$ ▷ Re-construction loss
    $\mathbf{X^s} \leftarrow \mathbf{X^s} - \eta \cdot \dfrac{\partial L}{\partial \mathbf{X^s}}$ ▷ SGD on $\mathbf{X^s}$
  return $\mathbf{X^s}$

2019), performing Kernelized Ridge Regression (KRR) with the estimated NTK is equivalent to training an infinitely-wide autoencoder for an infinite number of SGD steps. More formally, given the NTK, $\mathbb{K} : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}$ over an RKHS $\mathcal{H}$ of a single-layer autoencoder with an activation function σ (see (Radhakrishnan, 2022) for the NTK derivation of a fully-connected neural network), we reduce the recommendation problem to KRR as follows:

$$\begin{aligned} &\underset{[\alpha_j]_{j=1}^{|\mathcal{U}|}}{\arg\min} \quad \sum_{u\in\mathcal{U}} \|f(X_u \mid \alpha) - X_u\|_2^2 + \lambda \cdot \|f\|_{\mathcal{H}}^2 \\ \text{s.t.} \quad & f(X_u \mid \alpha) = \sum_{j=1}^{|\mathcal{U}|} \alpha_j \cdot \mathbb{K}(X_u, X_{u_j}) \;\; ; \;\; \mathbb{K}(X_u, X_v) = \tilde{\sigma}(X_u^T X_v) + \tilde{\sigma}'(X_u^T X_v) \cdot X_u^T X_v \end{aligned} \tag{3.1}$$

Where λ is a fixed regularization hyper-parameter, $\alpha := [\alpha_1; \alpha_2 \cdots; \alpha_{|\mathcal{U}|}] \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ are

the set of dual parameters we are interested in estimating, $\tilde{\sigma}$ represents the dual activation of σ (Daniely et al., 2016), and $\tilde{\sigma'}$ represents its derivative. Defining a gramian matrix $K \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{U}|}$ where each element can intuitively be seen as the *similarity* of two users, *i.e.*, $K_{i,j} \coloneqq \mathbb{K}(X_{u_i}, X_{u_j})$, the optimization problem defined in Equation (3.1) has a closed-form solution given by $\hat{\alpha} = (K + \lambda I)^{-1} \cdot X$. We can subsequently perform inference for any novel user as follows: $\hat{\mathrm{P}}(\cdot \mid u, \mathcal{D}) = \mathrm{softmax}(f(X_u \mid \hat{\alpha}))$. We also provide ∞-AE's training and inference pseudo-codes in Algorithms 3 and 4.

3.3.3 Scalability

We carefully examine the computational cost of ∞-AE's training and inference. Starting with training, ∞-AE has the following computationally-expensive steps: (1) computing the gramian matrix K; and (2) performing its inversion. The overall training time complexity thus comes out to be $\mathcal{O}(|\mathcal{U}|^2 \cdot |\mathcal{I}| + |\mathcal{U}|^{2.376})$ if we use the Coppersmith-Winograd algorithm (Coppersmith and Winograd, 1987) for matrix inversion, whereas the memory complexity is $\mathcal{O}(|\mathcal{U}| \cdot |\mathcal{I}| + |\mathcal{U}|^2)$ for storing the data matrix X and the gramian matrix K. As for inference for a single user, both the time and memory requirements are $\mathcal{O}(|\mathcal{U}| \cdot |\mathcal{I}|)$. Observing these computational complexities, we note that ∞-AE can be difficult to scale-up to larger datasets naïvely. To this effect, we address ∞-AE's scalability challenges in DISTILL-CF (Section 3.4), by leveraging a simple observation from support vector machines: not all datapoints (users in our case) are important for model learning. Additionally, in practice, we can perform all of these matrix operations on accelerators like GPU/TPUs and achieve a much higher throughput.

3.4 DISTILL-CF

3.4.1 Motivation

Representative sampling of large datasets has numerous downstream applications. Consequently, in this section we develop DISTILL-CF: a data distillation framework to *synthesize*

small, high-fidelity data summaries for collaborative filtering (CF) data. We design DISTILL-CF with the following rationales: (1) mitigating the scalability challenges in ∞-AE by training it only on the terse data summaries generated by DISTILL-CF; (2) improving the sample complexity of existing, finite-width recommendation models; (3) addressing the privacy risks of releasing user feedback data by releasing their synthetic data summaries instead; and (4) abating the downstream CO_2 emissions of frequent, large-scale recommendation model training by estimating their parameters only on much smaller data summaries synthesized by DISTILL-CF.

3.4.2 Challenges

Existing work in data distillation has focused on continuous domain data such as images (Nguyen et al., 2021a;b; Zhao et al., 2021; Zhao and Bilen, 2021), and are not directly applicable to heterogeneous and semi-structured domains such as recommender systems and graphs. This problem is further exacerbated since the data for these tasks is severely sparse. For example, in recommendation scenarios, a vast majority of users interact with very few items (Jain et al., 2016). Likewise, the nodes in a number of graph-based datasets tend to have connections with very small set of nodes (Zheng et al., 2022). We will later show how our DISTILL-CF framework is elegantly able to circumvent both these issues while being accurate, and scalable to large datasets.

3.4.3 Methodology

Given a recommendation dataset $\mathcal{D}$, we aim to synthesize a smaller, support dataset $\mathcal{D}^s$ that can match the performance of recommendation models $\phi : \mathcal{U} \times \mathcal{I} \mapsto \mathbb{R}$ when trained on $\mathcal{D}$ versus $\mathcal{D}^s$. We take inspiration from (Nguyen et al., 2021a), which is also a data distillation technique albeit for images. Formally, given a recommendation model ϕ, a held-out validation set $\mathcal{D}^{\text{val}}$, and a differentiable loss function $l : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ that measures the correctness of a prediction with the actual user-item relevance, the data distillation task can be viewed as the following bilevel optimization problem:

$$\arg\min_{\mathcal{D}^s} \quad \mathbb{E}_{(u,i,r)\sim\mathcal{D}^{\text{val}}} \left[l\left(\phi^*_{\mathcal{D}^s}(u,i), r\right)\right] \tag{3.2}$$

$$\text{s.t.} \quad \phi^*_{\mathcal{D}^s} \triangleq \arg\min_{\phi} \quad \mathbb{E}_{(u,i,r)\sim\mathcal{D}^s} \left[l(\phi(u,i), r)\right] \tag{3.3}$$

This optimization problem has an *outer loop* which searches for the most informative support dataset $\mathcal{D}^s$ given a fixed recommendation model ϕ^*, whereas the *inner loop* aims to find the optimal recommendation model for a fixed support set. In DISTILL-CF, we use ∞-AE as the model of choice at each step of the inner loop for two reasons: (1) as outlined in Section 3.3, ∞-AE has a closed-form solution with a single hyper-parameter λ, making the inner-loop extremely efficient; and (2) due to the infinite-width correspondence (Jacot et al., 2018; Arora et al., 2019), ∞-AE is equivalent to training an infinitely-wide autoencoder on $\mathcal{D}^s$, thereby not compromising on performance.

For the outer loop, we focus only on synthesizing *fake users* (given a fixed user budget μ) through a learnable matrix $X^s \in \mathbb{R}^{\mu\times|\mathcal{I}|}$ which stores the interactions for each fake user in the support dataset. However, to handle the discrete nature of the recommendation problem, instead of directly optimizing for X^s, DISTILL-CF instead learns a *continuous prior* for each user-item pair, denoting the importance of sampling that interaction in our support set (similar to the notion of propensity (Sachdeva et al., 2020; Schnabel et al., 2016)). We then sample $\hat{X}^s \sim X^s$ to get our final, discrete support set.

Instead of keeping this sampling operation post-hoc, *i.e.*, after solving for the optimization problem in Equation (3.2); we perform differentiable Gumbel-sampling (Jang et al., 2017) on each row (user) in X^s at every optimization step, thereby ensuring search only over sparse and discrete support sets. A notable property of recommendation data is that each user can interact with a variable number of items (this distribution is typically long-tailed due to Zipf's law (Wu et al., 2020)). We circumvent this dynamic user sparsity issue by taking multiple Gumbel-samples

for each user, with replacement. This implicitly gives DISTILL-CF the freedom to control the user and item popularity distributions in the generated data summary $\hat{X}^s$ by adjusting the entropy in the prior-matrix X^s.

Having controlled for the discrete and dynamic nature of recommendation data by the multi-step Gumbel-sampling trick, we further focus on maintaining the sparsity of the synthesized data. To do so, in addition to the outer-loop reconstruction loss, we add an L1-penalty over $\hat{X}^s$ to promote and explicitly control its sparsity (Hastie et al., 2009). Furthermore, tuning the number of Gumbel samples we take for each fake user, gives us more control over the sparsity in our generated data summary. More formally, the final optimization objective in DISTILL-CF can be written as:

$$\begin{aligned} \arg\min_{X^s} \quad & \mathop{\mathbb{E}}_{u\sim\mathcal{U}}\Big[X_u \cdot \log(K_{X_u\hat{X}^s}\cdot\alpha) + (1-X_u)\cdot\log(1-K_{X_u\hat{X}^s}\cdot\alpha)\Big] + \lambda_2\cdot||\hat{X}^s||_1 \\ \text{s.t.} \quad & \alpha = (K_{\hat{X}^s\hat{X}^s}+\lambda I)^{-1}\cdot\hat{X}^s \quad ; \quad \hat{X}^s = \sigma\left(\sum_{i=1}^{\gamma}\text{Gumbel}_\tau(\text{softmax}(X^s))\right) \end{aligned} \tag{3.4}$$

Where, λ_2 represents an appropriate regularization hyper-parameter for minimizing the L1-norm of the sampled support set $\hat{X}^s$, K_{XY} represents the gramian matrix for the NTK of ∞-AE over X and Y as inputs, τ represents the temperature hyper-parameter for Gumbel-sampling, γ denotes the number of samples to take for each fake user in X^s, and σ represents an appropriate activation function which clips all values over 1 to be exactly 1, thereby keeping $\hat{X}^s$ binary. We use hard-tanh in our experiments. We also provide DISTILL-CF's pseudo-code in Algorithm 5.

Further notes on multi-step Gumbel Sampling. We now discuss the Gumbel sampling procedure described in Equation (3.4), in more detail. Given the sampling prior matrix X^s, that intuitively denotes the importance of sampling a specific user-item interaction, we intend to sample $\hat{X}^s$ which will finally be used for downstream model applications. Note that for each row (user) in X^s, we need multiple, but variable number of samples to conform to the Zipfian law for

user and item popularity. This requirement in itself rejects the possibility to use top-K sampling which will sample the same number of items for each row. Furthermore, to keep $\hat{X}^s \sim X^s$ sampling part of the optimization procedure, we need to compute the gradients of the logistic objective in Equation (3.4) with respect to X^s, and hence need the entire process to be differentiable. This requirement prohibits the usage of other popular strategies like Nucleus sampling (Holtzman et al., 2020), which is non-differentiable. To workaround all the requirements, we devise a multi-step Gumbel sampling strategy where for each row (user) we take a fixed number of Gumbel samples (γ), with replacement, followed by taking a union of all the sampled user-item interactions. Note that the union operation ensures that due to sampling with replacement, if a user-item interaction is sampled multiple times, we sample it only once. Hence, the number of sampled interactions is strictly upper-bounded by $\gamma \times |\mathcal{I}|$. To be precise, the sampling procedure is formalized below:

$$\hat{X}^s_{u,i} = \sigma \left[\sum^{\gamma} \frac{exp(\frac{\log(X^s_{u,i}) + g_{u,i}}{\tau})}{\sum_{j \in \mathcal{I}} exp(\frac{\log(X^s_{u,j}) + g_{u,j}}{\tau})} \right] \quad \text{s.t.} \quad g_{u,i} \sim \text{Gumbel}(\mu = 0, \beta = 1) \quad \forall u \in \mathcal{U}, i \in \mathcal{I}$$

Where σ represents an appropriate function which clamps all values between 0 and 1. In our experiments, we use hard-tanh.

3.4.4 Scalability

We now analyze the time and memory requirements for optimizing Equation (3.4). The inner loop's major component clearly shares the same complexity as ∞-AE. However, since the parameters of ∞-AE (α in Equation (3.1)) are now being estimated over the much smaller support set $\hat{X}^s$, the time complexity reduces to $\mathcal{O}(\mu^2 \cdot |\mathcal{I}|)$ and memory to $\mathcal{O}(\mu \cdot |\mathcal{I}|)$, where μ typically only lies in the range of hundreds for competitive performance. On the other hand, for performing multi-step Gumbel-sampling for each synthetic user, the memory complexity of a naïve implementation would be $\mathcal{O}(\gamma \cdot \mu \cdot |\mathcal{I}|)$ since AutoGrad stores all intermediary variables

for backward computation. However, since the gradient of each Gumbel-sampling step is independent of other sampling steps and can be computed individually, using `jax.custom_vjp`, we reduced its memory complexity to $\mathcal{O}(\mu \cdot |\mathcal{I}|)$, adding nothing to the overall inner-loop complexity.

For the outer loop, we optimize the logistic reconstruction loss using SGD where we randomly sample a batch of b users from $\mathcal{U}$ and update X^s directly. In totality, for an ξ number of outer-loop iterations, the time complexity to run DISTILL-CF is $\mathcal{O}(\xi \cdot (\mu^2 + b + b \cdot \mu) \cdot |\mathcal{I}|)$, and $\mathcal{O}(b \cdot \mu + (\mu + b) \cdot |\mathcal{I}|)$ for memory. In our experiments, we note convergence in only hundreds of outer-loop iterations, making DISTILL-CF scalable even for the largest of the publicly available datasets and practically useful.

3.5 Empirical Setup

3.5.1 Datasets

We use four recommendation datasets with varying sizes and sparsity characteristics. A brief set of data statistics can be found in Table 3.1. For each user in the dataset, we randomly split their interaction history into $80/10/10\%$ train-test-validation splits. Following recent warnings against unrealistic dense preprocessing of recommender system datasets (Sachdeva and McAuley, 2020; Sachdeva et al., 2022c), we only prune users that have fewer than 3 interactions to enforce at least one interaction per user in the train, test, and validation sets. No such preprocessing is followed for items.

3.5.2 ∞-AE Baselines

We compare ∞-AE with various baseline and SoTA competitors as surveyed in recent comparative analyses (Anelli et al., 2022; Dacrema et al., 2019). We provide a high-level overview of the competitor models used in our experiments:

- **PopRec:** This implicit-feedback baseline simply recommends the most *popular* items in the

Table 3.1. Brief set of statistics of the datasets used in this chapter.

Dataset	# Users	# Items	# Interactions	Sparsity
Amazon Magazine (Ni et al., 2019b)	3k	1.3k	12k	99.7%
ML-1M (Harper and Konstan, 2015)	6k	3.7k	1M	95.6%
Douban (Zhu et al., 2019)	2.6k	34k	1.2M	98.7%
Netflix (Bennett and Lanning, 2007)	476k	17k	100M	98.9%

catalog irrespective of the user. Popularity of an item is estimated by their empirical frequency in the logged train-set.

- **Bias-only:** This baseline learns scalar user and item biases for all users and item in the train-set, optimized by solving a least-squares regression problem between the predicted and observed relevance. More formally, given a user u and an item i, the relevance is predicted as $\hat{r}_{u,i} = \alpha + \beta_u + \beta_i$, where $\alpha \in \mathbb{R}$ is a global offset bias, and $\beta_u, \beta_i \in \mathbb{R}$ are the user and item specific biases respectively. This model doesn't consider any cross user-item interactions, and hence lacks expressivity.

- **MF:** Building on top of the bias-only baseline, the Matrix Factorization algorithm tries to represent the users and items in a shared latent space, modeling their relevance by the dot-product of their representations. More formally, given a user u and an item i, the relevance is predicted as $\hat{r}_{u,i} = \alpha + \beta_u + \beta_i + (\gamma_u \cdot \gamma_i)$, where α, β_u, β_i are global, user, and item biases respectively, and $\gamma_u, \gamma_i \in \mathbb{R}^d$ represent the learned user and item representations. The biases and latent representations in this model are estimated by optimizing for the Bayesian Personalized Ranking (BPR) loss (Rendle et al., 2009).

- **NeuMF (He et al., 2017):** As a neural extension to MF, Neural Matrix Factorization aims to replace the linear cross-interaction between the user and item representations with an arbitrarily complex, non-linear neural network. More formally, given a user u and an item i, the relevance is predicted as $\hat{r}_{u,i} = \alpha + \beta_u + \beta_i + \phi(\gamma_u, \gamma_i)$, where α, β_u, β_i are global, user,

and item biases respectively, $\gamma_u, \gamma_i \in \mathbb{R}^d$ represent the learned user and item representations, and $\phi : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ is a neural network. The parameters for this model are again optimized using the BPR loss.

- **MVAE (Liang et al., 2018):** This method proposed using variational auto-encoders for the task of collaborative filtering. Their main contribution was to provide a principled approach to perform variational inference for the task of collaborative filtering.

- **LightGCN (He et al., 2020):** This simplistic Graph Convolution Network (GCN) framework removes all the steps in a typical GCN (Kipf and Welling, 2017a), only keeping a linear neighbourhood aggregation step. This *light* model demonstrated promising results for the collaborative filtering scenario, despite its simple architecture. We use the official public implementation[1] for our experiments.

- **EASE (Steck, 2019):** This linear model proposed doing ordinary least squares regression by estimating an item-item similarity matrix, that can be viewed as a zero-depth auto-encoder. Performing regression gives them the benefit of having a closed-form solution. Despite its simplicity, EASE has been shown to out-perform most of the deep non-linear neural networks for the task of collaborative filtering.

3.5.3 DISTILL-CF Baselines

Given a recommendation dataset $\mathcal{D} \coloneqq \{(\text{user}_i, \text{item}_i, \text{relevance}_i)\}_{i=1}^{n}$ consisting of n user-item interactions defined over a set of users $\mathcal{U}$, set of items $\mathcal{I}$, and operating over a binary relevance score (implicit feedback); we aim to make a $p\%$ sub-sample of $\mathcal{D}$, defined in terms of number of interactions. Below are the different sampling strategies we used in comparison with DISTILL-CF:

- **Interaction-RNS:** Randomly sample $p\%$ interactions from $\mathcal{D}$.

[1] https://github.com/gusye1234/LightGCN-PyTorch

- **User-RNS:** Randomly sample a user $u \sim \mathcal{U}$, and add all of its interactions into the sampled set. Keep repeating this procedure until the size of sampled set is less than $\frac{p \times n}{100}$.

- **Head user:** Sample the user u from $\mathcal{U}$ with the most number of interactions, and add all of its interactions into the sampled set. Remove u from $\mathcal{U}$. Keep repeating this procedure until the size of sampled set is less than $\frac{p \times n}{100}$.

- **SVP-CF (user):** This coreset mining technique was introduced in Chapter 1, and proceeds by first training a proxy model on $\mathcal{D}$ for e epochs. SVP-CF then modifies the forgetting events approach (Toneva et al., 2019), and counts the inverse AUC for each user in $\mathcal{U}$, averaged over all e epochs. Just like head-user sampling, we now iterate over users in the order of their forgetting count, and keep sampling users until we exceed our sampling budget of $\frac{p \times n}{100}$ interactions. We use the bias-only model as the proxy.

3.5.4 Evaluation metrics

We evaluate all models on a variety of pertinent ranking metrics, namely AUC, HitRate (HR@k), Normalized Discounted Cumulative Gain (nDCG@k), and Propensity-scored Precision (PSP@k), each focusing on different components of the algorithm performance. A notable addition to our list of metrics compared to the literature is the PSP metric (Jain et al., 2016), which we adapt to the recommendation use case as an indicator of performance on tail items. We now present formal definitions of all the aforementioned ranking metrics:

- **AUC:** Intuitively defined as a threshold independent classification performance measure, AUC can also be interpreted as the expected probability of a recommender system ranking a positive item over a negative item for any given user. More formally, given a user u from the user set $\mathcal{U}$ with its set of positive interactions $\mathcal{I}_u^+ \subseteq \mathcal{I}$ with a similarly defined set of negative interactions $\mathcal{I}_u^- = \mathcal{I} \backslash \mathcal{I}_u^+$, the AUC for a relevance predictor $\phi(u,i)$ is defined as:

$$\text{AUC}(\phi) \coloneqq \mathop{\mathbb{E}}_{u \sim \mathcal{U}} \left[\mathop{\mathbb{E}}_{i^+ \sim \mathcal{I}_u^+} \left[\mathop{\mathbb{E}}_{i^- \sim \mathcal{I}_u^-} \left[\phi(u,i^+) > \phi(u,i^-) \right] \right] \right]$$

Table 3.2. List of all the hyper-parameters grid-searched for ∞-AE, DISTILL-CF, and baselines.

Hyper-Parameter	Model	Amz. Magazine	ML-1M	Douban	Netflix
Latent size	MF NeuMF LightGCN MVAE	{4, 8, 16, 32, 50, 64, 128}			
# Layers	MF NeuMF LightGCN MVAE	{1, 2, 3}			
	∞-AE	{1}			
Learning rate	MF NeuMF LightGCN MVAE	{0.001, 0.006, 0.01}			{0.006}
	DISTILL-CF	{0.04}			
Dropout	MF NeuMF LightGCN MVAE	{0.0, 0.3, 0.5}			
λ	EASE	{1, 10, 100, 1K, 10K}			
	∞-AE	{0.0, 1.0, 5.0, 20.0 50.0, 100.0}			
	DISTILL-CF	{1e-5, 1e-3, 0.1, 1.0, 5.0, 50.0}			
λ_2	DISTILL-CF	$\frac{\{1e\text{-}3,\ 10.0\}}{\text{avg. \# of interactions per user}}$			
τ	DISTILL-CF	{0.3, 0.5, 0.7, 5.0}			
γ	DISTILL-CF	{50, 100, 200}	{200, 500, 700}	{500, 1K, 2K}	{500, 700}

- **HitRate (HR@k):** Another name for the recall metric, this metric estimates how many positive items are predicted in a top-k recommendation list. More formally, given recommendation lists $\hat{Y}_u \subseteq \mathcal{I}^K \quad \forall u \in \mathcal{U}$ and the set of positive interactions $\mathcal{I}_u^+ \subseteq \mathcal{I} \quad \forall u \in \mathcal{U}$:

$$\text{HR@k} := \mathop{\mathbb{E}}_{u \sim \mathcal{U}} \left[\frac{|\hat{Y}_u \cap \mathcal{I}_u^+|}{|\mathcal{I}_u^+|} \right]$$

- **Normalized Discounted Cumulative Gain (nDCG@k):** Unlike HR@k which gives equal importance to all items in the recommendation list, the nDCG@k metric instead gives a higher importance to items predicted higher in the recommendation list and performs logarithmic discounting further down. More formally, given *sorted* recommendation lists $\hat{Y}_u \subseteq \mathcal{I}^K \quad \forall u \in \mathcal{U}$ and the set of positive interactions $\mathcal{I}_u^+ \subseteq \mathcal{I} \quad \forall u \in \mathcal{U}$:

$$\text{nDCG@k} := \mathop{\mathbb{E}}_{u \sim \mathcal{U}} \left[\frac{\text{DCG}_u}{\text{IDCG}_u} \right] \; ; \; \text{DCG}_u := \sum_{i=1}^{k} \frac{\hat{Y}_u^i \in \mathcal{I}_u^+}{log_2(i+1)} \; ; \; \text{IDCG}_u := \sum_{i=1}^{|\mathcal{I}_u^+|} \frac{1}{log_2(i+1)}$$

- **Propensity-scored Precision (PSP@k):** Originally introduced in (Jain et al., 2016) for extreme classification scenarios (Prabhu et al., 2018; Jain et al., 2019; Mittal et al., 2021), the PSP@k metric intuitively accounts for missing labels (items in the case of recommendation) by dividing the true relevance of an item (binary) with a propensity correction term. More formally, given recommendation lists $\hat{Y}_u \subseteq \mathcal{I}^K \quad \forall u \in \mathcal{U}$, the set of positive interactions $\mathcal{I}_u^+ \subseteq \mathcal{I} \quad \forall u \in \mathcal{U}$, and a propensity model $\phi : \mathcal{I} \mapsto \mathbb{R}$:

$$\text{PSP@k} := \mathop{\mathbb{E}}_{u \sim \mathcal{U}} \left[\frac{\text{uPSP}_u}{\text{mPSP}_u} \right] \; ; \; \text{uPSP}_u := \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{\hat{Y}_u^i \in \mathcal{I}_u^+}{\phi(\hat{Y}_u^i)} \; ; \; \text{mPSP}_u := \sum_{i \in \mathcal{I}_u^+} \frac{1}{\phi(i)}$$

  For ϕ, we adapt the propensity model proposed in (Jain et al., 2016) for the case of recommendation as:

$$\phi(i) \equiv \mathop{\mathbb{E}}_{u\sim\mathcal{U}}\left[\mathrm{P}(r_{u,i}=1|r^*_{u,i}=1)\right] = \frac{1}{1+C\cdot e^{-A\cdot log(n_i+B)}}\ ;\ C=(logN-1)\cdot(B+1)^A$$

Where, N represents the total number of interactions in the dataset, and n_i represents the empirical frequency of item i in the dataset. We use $A = 0.55$ and $B = 1.5$ for our experiments.

3.5.5 Training details

We implement both ∞-AE and DISTILL-CF using `JAX` (Bradbury et al., 2018) along with the Neural-Tangents package (Novak et al., 2020) for the relevant NTK computations.[2,3] We re-use the official implementation of LightGCN, and implement the remaining competitors ourselves. To ensure a fair comparison, we conduct a hyper-parameter search for all competitors on the validation set. More details on the hyper-parameters for ∞-AE, DISTILL-CF, and all competitors can be found in Table 3.2. All of our experiments are performed on a single RTX 3090 GPU, with a random-seed initialization of 42.

We also make use of a validation-set, and evaluate the performance of the ∞-AE model trained in DISTILL-CF's inner-loop to perform hyper-parameter search, as well as early exit. Note that we don't validate the inner-loop's λ at every outer-loop iteration, but keep changing it on-the-fly at each validation cycle. We notice this trick gives us a much faster convergence compared to keeping λ fixed for the entire distillation procedure, and validating for it like other hyper-parameters.

3.6 Experiments

3.6.1 Does ∞-AE outperform existing methods?

We compare the performance of ∞-AE with various baseline and competitor methods in Table 3.3. We also include the results of training ∞-AE on data synthesized by DISTILL-CF with an additional constraint of having a budget of only $\mu = 500$ synthetic users. For the sake

[2]Our implementation for ∞-AE is available at https://github.com/noveens/infinite_ae_cf

[3]Our implementation for DISTILL-CF is available at https://github.com/noveens/distill_cf

Table 3.3. Comparison of ∞-AE with different methods on various datasets. All metrics are better when higher. Brief set of data statistics can be found in Table 1.2. **Bold** values represent the best in a given row, and underlined represent the second-best. Results for ∞-AE on the Netflix dataset (marked with a *) consist of random user-sampling with a max budget of 25K $\equiv 5.4\%$ users, and results for DISTILL-CF + ∞-AE have a user-budget of 500 for all datasets.

Dataset	Metric	PopRec	Bias only	MF	NeuMF	Light GCN	EASE	MVAE	∞-AE	DISTILL-CF + ∞-AE
Amazon Magazine	AUC	0.8436	0.8445	0.8475	0.8525	0.8141	0.6673	0.8507	0.8539	**0.8584**
	HR@10	14.35	14.36	18.36	18.35	27.12	26.31	17.94	27.09	**28.27**
	HR@100	59.5	59.35	58.94	59.3	58.00	48.36	57.3	60.86	**61.78**
	NDCG@10	8.42	8.33	13.1	13.6	22.57	22.84	12.18	23.06	**23.81**
	NDCG@100	19.38	19.31	21.76	21.13	29.92	28.27	19.46	30.75	**31.75**
	PSP@10	6.85	6.73	9.24	9.00	13.2	12.96	8.81	13.22	**13.76**
ML-1M	AUC	0.8332	0.8330	0.9065	0.9045	0.9289	0.9069	0.8832	**0.9457**	0.9415
	HR@10	13.07	12.93	24.63	23.25	27.43	28.54	21.7	**31.51**	31.16
	HR@100	30.38	29.63	53.26	51.42	55.61	57.28	52.29	**60.05**	58.28
	NDCG@10	13.84	13.74	25.65	24.44	28.85	29.88	22.14	**32.82**	32.52
	NDCG@100	19.49	19.13	35.62	33.93	38.29	40.16	33.82	**42.53**	41.29
	PSP@10	1.10	1.07	2.41	2.26	2.72	3.06	2.42	**3.22**	3.15
Douban	AUC	0.8945	0.8932	0.9288	0.9258	0.9391	0.8570	0.9129	**0.9523**	0.9510
	HR@10	11.06	10.71	12.69	12.79	15.98	17.93	15.36	**23.56**	22.98
	HR@100	17.07	16.63	20.29	19.69	22.38	25.41	22.82	**28.37**	27.20
	NDCG@10	11.63	11.24	13.21	13.33	16.68	19.48	16.17	**24.94**	24.20
	NDCG@100	12.63	12.27	14.96	14.39	17.20	19.55	17.32	**23.26**	22.21
	PSP@10	0.52	0.50	0.63	0.63	0.86	1.06	0.87	**1.28**	1.24
Netflix	AUC	0.9249	0.9234	0.9234	0.9244		0.9418	0.9495	0.9663*	**0.9728**
	HR@10	12.14	11.49	11.69	11.06		26.03	20.6	**29.69***	29.57
	HR@100	28.47	27.66	27.72	26.76	Timed	50.35	44.53	**50.88***	49.24
	NDCG@10	12.34	11.72	12.04	11.48	Out	26.83	20.85	**30.59***	30.54
	NDCG@100	17.79	16.95	17.17	16.40		35.09	29.22	**36.59***	35.58
	PSP@10	1.45	1.28	1.31	1.21		3.59	2.77	**3.75***	3.62

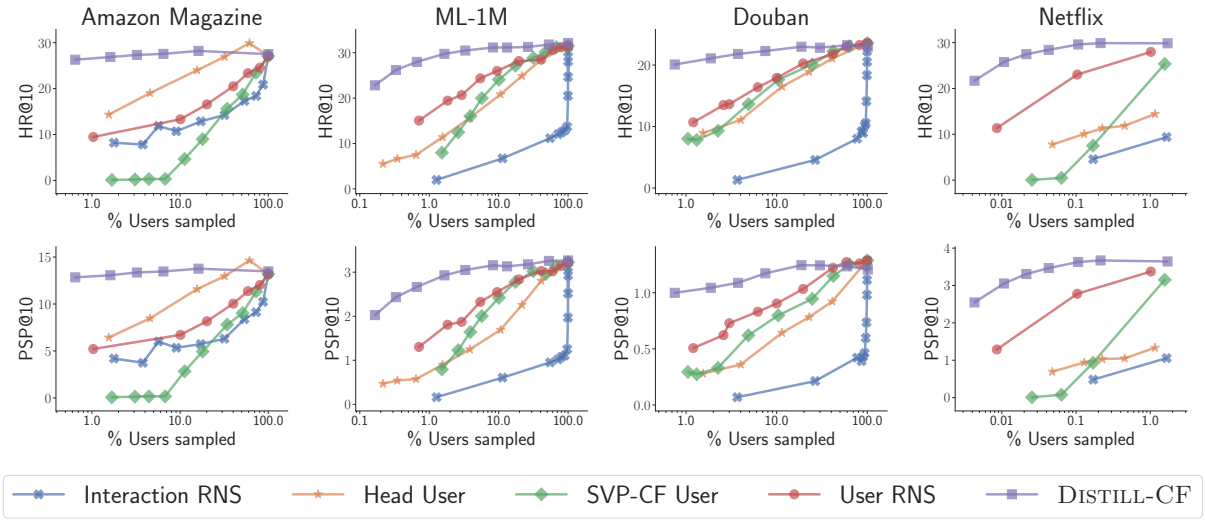

Figure 3.1. Performance of ∞-AE with the amount of users (log-scale) sampled according to different sampling strategies over the HR@10 and PSP@10 metrics. Results for the Netflix dataset have been clipped due to memory constraints.

of reference, for our largest dataset (Netflix), this equates to a mere 0.1% of the total users. There are a few prominent observations from the results in Table 3.3. First, ∞-AE significantly outperforms SoTA recommendation algorithms despite having only a single fully-connected layer, and also being much simpler to train and implement. Second, we note that ∞-AE trained on just 500 users generated by DISTILL-CF is able to attain $96-105\%$ of ∞-AE's performance on the full dataset while also outperforming all competitors trained on the *full* dataset.

3.6.2 How sample efficient is ∞-AE?

Having noted from Table 3.3 that ∞-AE is able to outperform all SoTA competitors with as little as 0.1% of the original users, we now aim to better understand ∞-AE's sample complexity, *i.e.*, the amount of training data ∞-AE needs in order to perform accurate recommendation. In addition to DISTILL-CF, we use the following popular heuristics for down-sampling (Section 2.4.3): interaction random negative sampling (RNS); user RNS; head user sampling; and a coreset construction technique, SVP-CF user (Chapter 1). We then train ∞-AE on sampled data for different sampling budgets, while evaluating on the original test-set. We plot the performance

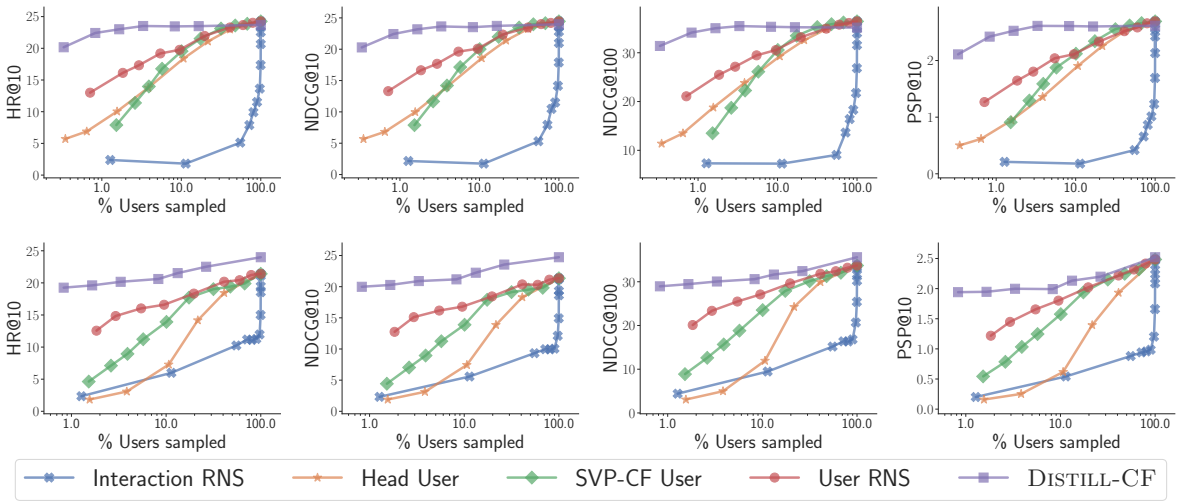

Figure 3.2. Performance of the (top) EASE model, and (bottom) MVAE model trained on different amounts of users (log-scale) sampled by different samplers on the ML-1M dataset.

for all datasets computed over the HR@10 and PSP@10 metrics in Figure 3.1. We observe that while all heuristic sampling strategies tend to be closely bound to the identity line with a slight preference to user RNS, ∞-AE when trained on data synthesized by DISTILL-CF tends to quickly saturate in terms of performance when the user budget is increased, even on the log-scale. This indicates DISTILL-CF's superiority in generating terse data summaries for ∞-AE, thereby allowing it to get SoTA performance on the largest datasets with as little as 500 users.

3.6.3 How transferable are the data summaries synthesized by DISTILL-CF?

In order to best evaluate the quality and universality of data summaries synthesized by DISTILL-CF, we train and evaluate EASE (Steck, 2019) on data synthesized by DISTILL-CF. Note that the inner loop of DISTILL-CF still consists of ∞-AE, but we nevertheless train and evaluate both EASE and MVAE to test the synthesized data's universality. We re-use the heuristic sampling strategies from the previous experiment for comparison with DISTILL-CF. From the results in Figure 3.2, we observe similar scaling laws as ∞-AE's for the heuristic samplers as well as DISTILL-CF. This behaviour validates the re-usability of data summaries generated by

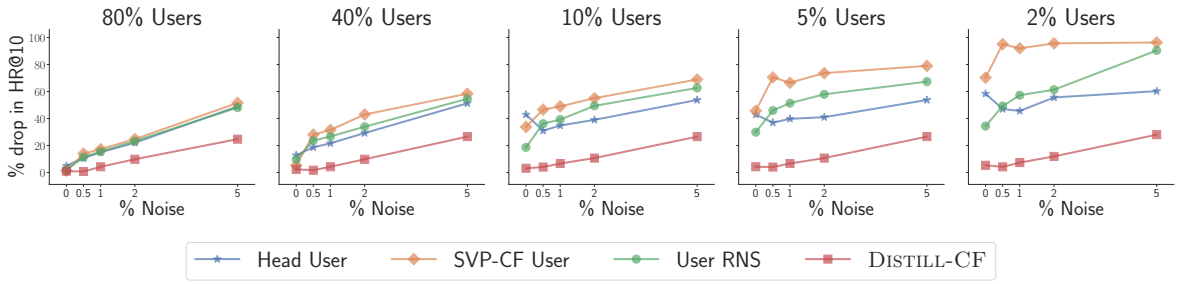

Figure 3.3. Performance drop of the EASE model trained on data sampled by different sampling strategies when there is varying levels of noise in the data. Performance drop is relative to training on the full, noise-free ML-1M dataset.

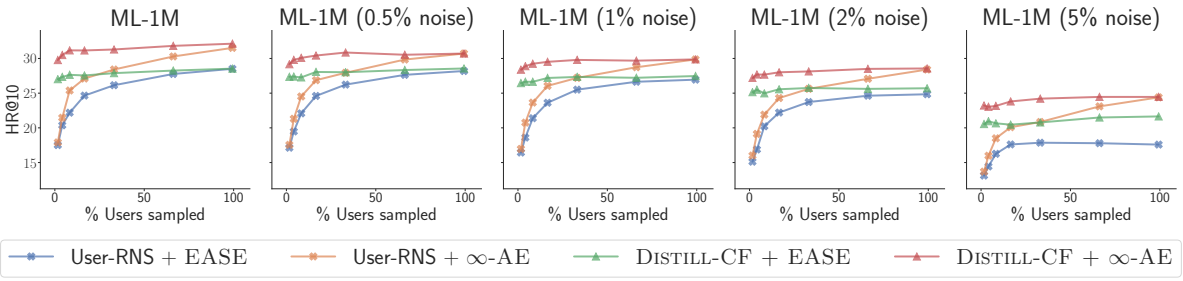

Figure 3.4. Performance of ∞-AE on data sampled by DISTILL-CF and User-RNS when there is noise in the data. Results for EASE have been added for reference. All results are on the ML-1M dataset.

DISTILL-CF, because they transfer well to SoTA finite-width models, which were not involved in DISTILL-CF's user synthesis optimization.

3.6.4 How robust are DISTILL-CF and ∞-AE to noise?

User feedback data is often noisy due to various biases (see (Chen et al., 2020) for a detailed review). Furthermore, due to the significant number of logged interactions in these datasets, recommender systems are often trained on down-sampled data in practice. Despite this, to the best of our knowledge, there is no prior work that explicitly studies the interplay between noise in the data and how sampling it affects downstream model performance. Consequently, we simulate a simple experiment: we add $x\%$ noise in the original train-set $\rightarrow$ sample the noisy training data $\rightarrow$ train recommendation models on the sampled data $\rightarrow$ evaluate their performance

on the original, noise-free test-set. For the noise model, we randomly flip $x\%$ of the total number of items in the corpus for each user. In Figure 3.3, we compare the drop in HR@10 the EASE model suffers for different sampling strategies when different levels of noise are added to the MovieLens-1M dataset (Harper and Konstan, 2015). We make a few main observations: (1) unsurprisingly, sampling noisy data compounds the performance losses of learning algorithms; (2) DISTILL-CF has the best noise:sampling:performance trade-off compared to other sampling strategies, with an increasing performance gap relative to other samplers as we inject more noise into the original data; and (3) as we down-sample noisy data more aggressively, head user sampling improves relative to other samplers, simply because these head users are the least affected by our noise injection procedure.

Furthermore, to better understand ∞-AE's denoising capabilities, we repeat the aforementioned noise-injection experiment but now train ∞-AE on down-sampled, noisy data. In Figure 3.4, we track the change in ∞-AE's performance as a function of the number of users sampled, and the amount of noise injected before sampling. We also add the semantically equivalent results for the EASE model for reference. Firstly, we note that the full-data performance-gap between ∞-AE and EASE significantly increases when there is more noise in the data, demonstrating ∞-AE's robustness to noise, even when its not specifically optimized for it. Furthermore, looking at the 5% noise injection scenario, we notice two counter-intuitive observations: (1) training EASE on tiny data summaries synthesized by DISTILL-CF is better than training it on the full data; and (2) solely looking at data synthesized by DISTILL-CF for EASE, we notice the best performance when we have a lower user sampling budget. Both of these observations call for more investigation of a data-centric viewpoint to recommendation, *i.e.*, focusing more on the quality of collected data rather than its quantity.

3.6.5 Applications to continual learning

Continual learning (see (Parisi et al., 2019) for a detailed review) is an important area for recommender systems, because these systems are typically updated at regular intervals. A

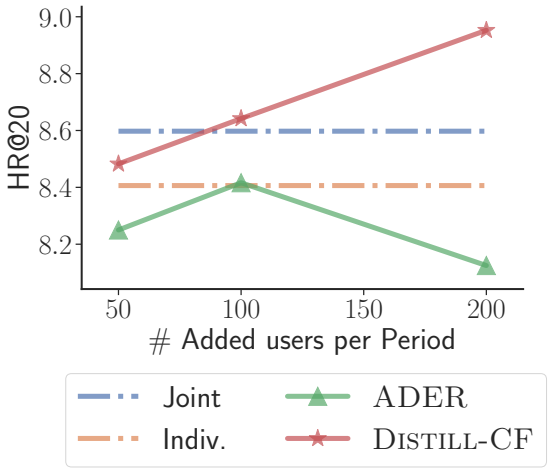

Figure 3.5. DISTILL-CF for continual learning.

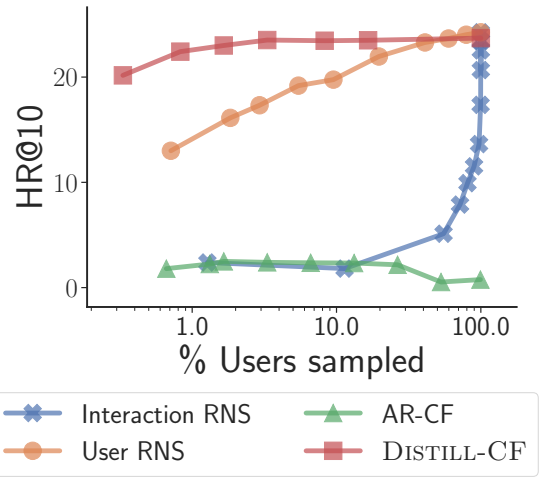

Figure 3.6. Performance of EASE on varying amounts of data sampled/synthesized using various strategies for the MovieLens-1M dataset.

continual learning scenario involves data that is split into multiple periods, with the predictive task being: given data until the i^{th} period, maximize algorithm performance for prediction on the $(i+1)^{\text{th}}$ period. ADER (Mi et al., 2020) is a SoTA continual learning model for recommender systems, that injects the most informative user sequences from the last period to combat the catastrophic forgetting problem (Rolnick et al., 2019). An intuitive application for DISTILL-CF is to synthesize succinct data summaries of the last period and inject these instead. To compare these approaches, we simulate a continual learning scenario by splitting the MovieLens-1M dataset into 17 equal sized epochs, and perform experiments with MVAE (Liang et al., 2018) for each period. Note that in DISTILL-CF, we still use ∞-AE to synthesize data summaries (inner loop). We also compare with two baselines: (1) Joint: concatenate the data from all periods before the current; and (2) Individual: use the data only from the current period. As we can see from Figure 3.5, DISTILL-CF consistently outperforms ADER and the baselines, again demonstrating its ability to generate high-fidelity data summaries.

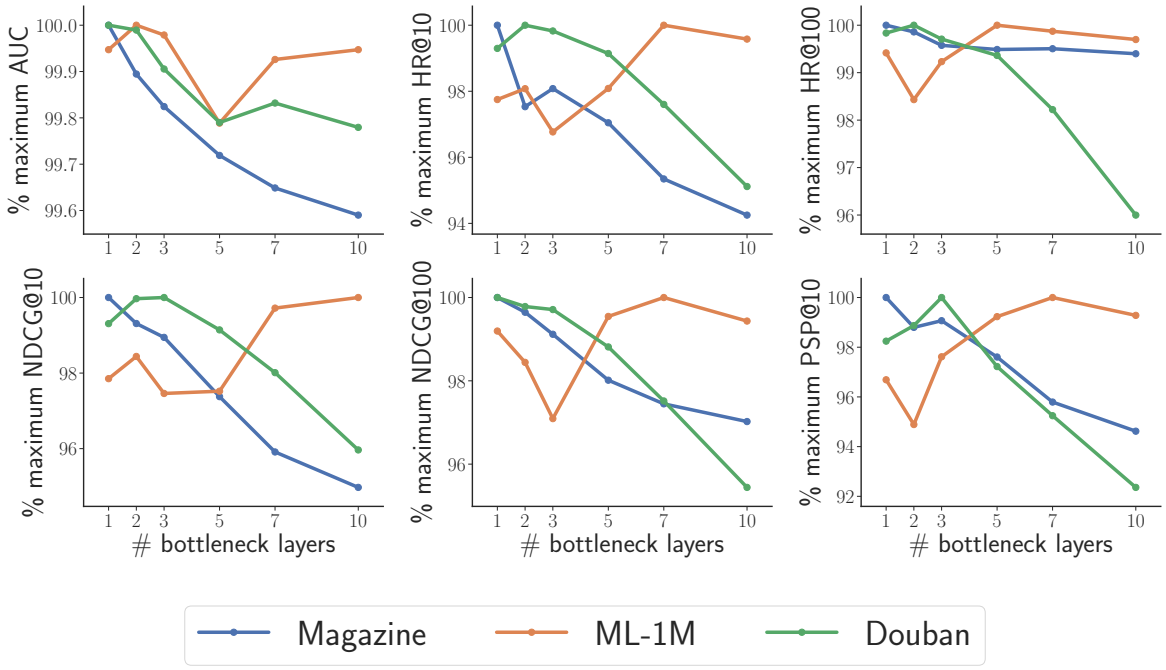

Figure 3.7. Performance of ∞-AE with varying depths. The y-axis represents the normalized metric *i.e.* performance relative to the best depth for a given metric.

3.6.6 How does DISTILL-CF compare to data augmentation approaches?

We compare the quality of data synthesized by DISTILL-CF with generative models proposed for data augmentation. One such SoTA method is AR-CF (Chae et al., 2020), which leverages two conditional GANs (Mirza and Osindero, 2014) to generate fake users and fake items. For our experiment, we focus only on AR-CF's user generation sub-network and consequently train the EASE (Steck, 2019) model *only* on these synthesized users, while testing on the original test-set for the MovieLens-1M dataset. We plot the results in Figure 3.6, comparing the amount of users synthesized according to different strategies and plot the HR@10 of the correspondingly trained model. The results signify that training models only on data synthesized by data augmentation models is impractical, as these users have only been optimized for being *realistic*, whereas the users synthesized by DISTILL-CF are optimized to be *informative for model training*. The same observation tends to hold true for the case of images as well (Zhao

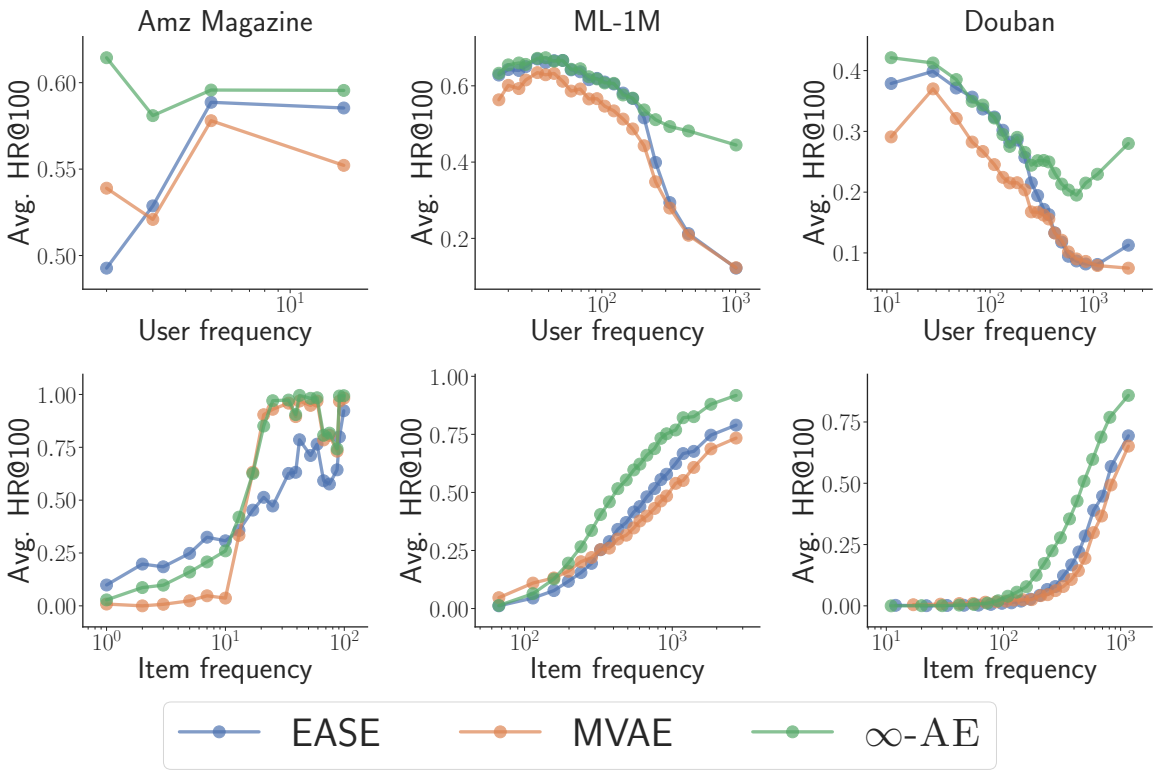

Figure 3.8. Performance comparison of ∞-AE with SoTA finite-width models stratified over the coldness of users and items. The y-axis represents the average HR@100 for users/items in a particular quanta. All user/item bins are equisized.

et al., 2021).

3.6.7 How does depth affect ∞-AE?

To better understand the effect of depth on an infinitely-wide auto-encoder's performance for recommendation, we extend ∞-AE to multiple layers and note its downstream performance change in Figure 3.7. The prominent observation is that models tend to get *worse* as they get deeper, with generally good performance in the range of $1-2$ layers, which also has been common practical knowledge even for finite-width recommender systems.

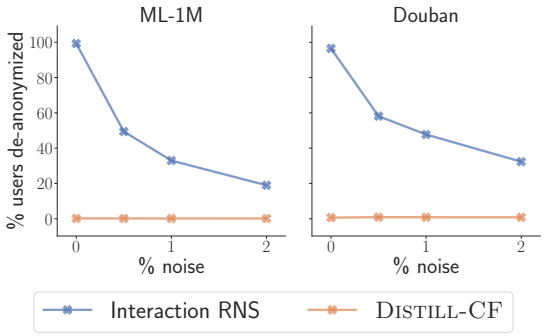

Figure 3.9. Amount of noise added in $\mathcal{D}'$ *vs.* % of users de-anonymized.

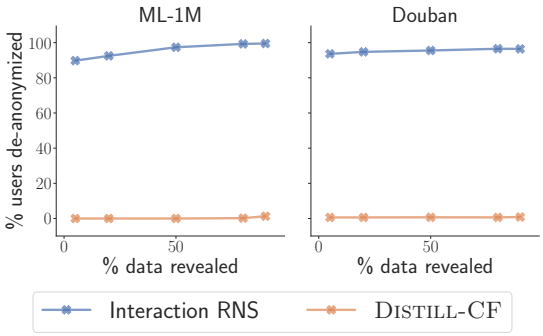

Figure 3.10. Amount of data revealed in $\mathcal{D}'$ *vs.* % of users de-anonymized.

3.6.8 How does ∞-AE perform on cold users & cold items?

Cold-start has been one of the hardest problems in recommender systems — how to best model users or items that have very little training data available? Even though ∞-AE doesn't have any extra modeling for these scenarios, we try to better understand the performance of ∞-AE over users' and items' coldness spectrum. In Figure 3.8, we quantize different users and items based on their coldness (computed by their empirical occurrence in the train-set) into equisized buckets and measure different models' performance only on the binned users or items. We note ∞-AE's dominance over other competitors especially over the tail, head-users; and torso, head-items.

3.6.9 How anonymized is the data synthesized by DISTILL-CF?

Having evaluated the fidelity of distills generated using DISTILL-CF, we now focus on understanding its anonymity and syntheticity. For the generic data down-sampling case, the algorithm presented in (Narayanan and Shmatikov, 2008) works well to de-anonymize the Netflix prize dataset. The algorithm assumes a complete, non-PII dataset $\mathcal{D}$ along with an incomplete, noisy version of the same dataset $\mathcal{D}'$, but also has the sensitive PII available. We simulate a similar setup but extend to datasets other than Netflix, by following a simple down-sampling and noise addition procedure: given a sampling strategy s, down-sample $\mathcal{D}$ and add $x\%$ noise

by randomly flipping $x\%$ of the total items for each user to generate $\mathcal{D}'$. We then use our implementation of the algorithm proposed in (Narayanan and Shmatikov, 2008) to *match* the corresponding users in the original, noise-free dataset $\mathcal{D}$.

However, if instead of a down-sampled dataset $\mathcal{D}'$, a data distill of $\mathcal{D}$ (using DISTILL-CF), let's say $\tilde{\mathcal{D}}$, is made publicly available. The task of de-anonymization can no longer be carried out by simply *matching* user histories from $\mathcal{D}'$ to $\mathcal{D}$, since $\mathcal{D}$ is no longer available. The only solution now is to *predict* the missing items in $\mathcal{D}'$. Note that this this task is easier than the usual recommendation problem, as the user histories to complete in $\mathcal{D}'$ do exist in some incoherent way in the data distill $\tilde{\mathcal{D}}$, and is more similar to train-set prediction. To test this out, we formulate a simple experiment: given a data distill $\tilde{\mathcal{D}}$, an incomplete, noisy subset $\mathcal{D}'$ with PII information, and also hypothetically the number of missing items for each user in $\mathcal{D}'$ — how accurately can we predict the *exact* set of missing items in $\mathcal{D}'$ using an ∞-AE model trained on $\tilde{\mathcal{D}}$.

We perform experiments for both the cases of data-sampling and data-distillation. In Figure 3.9, we measure the % of users de-anonymized using the aforementioned procedures. We interestingly note no level of de-anonymization with the data-distill, even if there's no noise in $\mathcal{D}'$. We also note the expected observation for the data-sampling case: less users are de-anonymized when there's more noise in $\mathcal{D}'$. In Figure 3.10, we now control the amount of data revealed in $\mathcal{D}'$. We again note the same observation: even with 90% of the correct data from $\mathcal{D}$ revealed in $\mathcal{D}'$ with 0% of noise, we still note a very tiny 0.86% of user de-anonymization with data-distillation, whereas 96.43% with data-sampling for the Douban dataset.

3.7 Conclusion & Future Work

In this work, we proposed two complementary ideas: ∞-AE, an infinite-width autoencoder for modeling recommendation data, and DISTILL-CF for creating tiny, high-fidelity data summaries of massive datasets for subsequent model training. To our knowledge, our work is the

first to employ and demonstrate that infinite-width neural networks can beat complicated SoTA models on recommendation tasks. Further, the data summaries synthesized through DISTILL-CF outperform generic samplers and demonstrate further performance gains for ∞-AE as well as finite-width SoTA models despite being trained on orders of magnitude less data.

Both our proposed methods are closely linked with one another: ∞-AE's closed-loop formulation is especially crucial in the practicality of DISTILL-CF, whereas DISTILL-CF's ability to distill the entire dataset's knowledge into small summaries helps ∞-AE to scale to large datasets. Moreover, the Gumbel sampling trick enables us to adapt data distillation techniques designed for continuous, real-valued, dense domains to heterogeneous, semi-structured, and sparse domains like recommender systems and graphs. We additionally explore the strong denoising effect observed with DISTILL-CF, noting that in the case of noisy data, models trained on considerably less data synthesized by DISTILL-CF perform better than the same model trained on the entire original dataset. These observations lead us to contemplate a much larger, looming question: *Is more data what you need for recommendation?* Our results call for further investigation on the data-centric viewpoint of recommendation.

The findings of this chapter open up numerous promising research directions. First, building such closed-form, easy-to-implement infinite networks is beneficial for various downstream practical applications like search, sequential recommendation, or CTR prediction. Further, the anonymization achieved by synthesizing fake data summaries is crucial for mitigating the privacy risks associated with confidential or PII datasets. Another direction is analyzing the environmental impact and reduction in carbon footprint as our experiments show that models can achieve similar performance gains when trained on much less data.

Chapter 3, in part, is a reprint of the material as it appears in "Infinite Recommendation Networks: A Data-centric Approach." by Noveen Sachdeva, Mehak Preet Dhaliwal, Carole-Jean Wu, and Julian McAuley, in *Advances in Neural Information Processing Systems. 2022.* The dissertation author was the primary investigator and author of this paper.

Chapter 4

Data Distillation for Autoregressive Data

We study data distillation for auto-regressive machine learning tasks, where the input and output have a strict left-to-right causal structure. More specifically, we propose FARZI, which summarizes an event sequence dataset into a small number of *synthetic* sequences — FARZI DATA — which are optimized to maintain (if not improve) model performance compared to training on the full dataset. Under the hood, FARZI conducts memory-efficient data distillation by (i) deriving efficient reverse-mode differentiation of the Adam optimizer by leveraging Hessian-Vector Products; and (ii) factorizing the high-dimensional discrete event-space into a latent-space which provably promotes implicit regularization. Empirically, for sequential recommendation and language modeling tasks, we are able to achieve $98-120\%$ of downstream full-data performance when training state-of-the-art models on FARZI DATA of size as little as 0.1% of the original dataset, translating to roughly $20\times$ faster training. Notably, being able to train *better models with significantly less data* sheds light on the design of future large auto-regressive models, and opens up new opportunities to further scale up model and data sizes.

4.1 Introduction

The effectiveness of machine learning models relies heavily on the *quantity* and *quality* of training data. While the quantity of training data is always well-regarded in the scaling-laws of training highly-parameterized neural networks (Hoffmann et al., 2022; Kaplan et al., 2020;

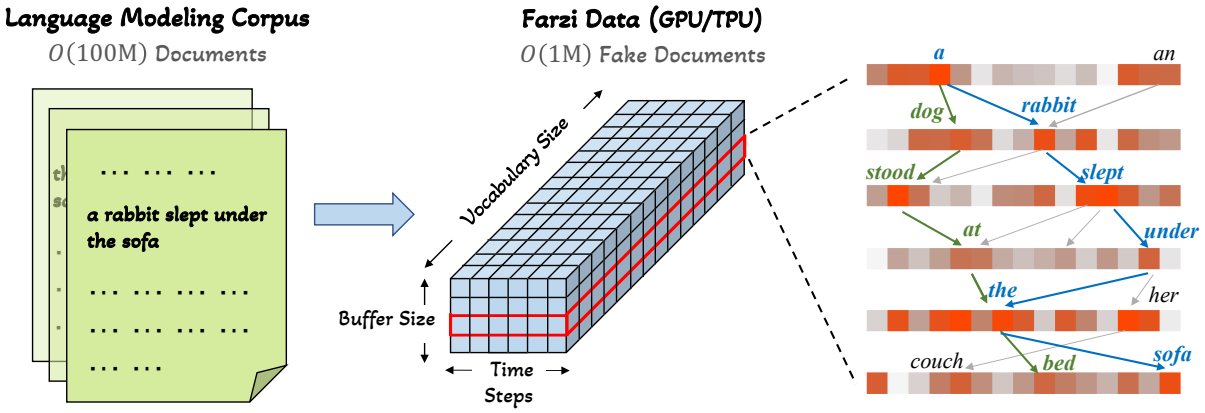

Figure 4.1. Visualization of FARZI DATA in the context of language modeling. FARZI DATA can be seen as a 3-D tensor comprising of *sequences of distributions over tokens*, where a single *distribution sequence* fuses the information content of multiple discrete sequences. E.g, a single distribution sentence can unfold into an entire tree of similar sentences like "the rabbit slept under the sofa" or "a dog stood at the bed" as depicted in the figure. Such a parameterization: (i) makes the dataset GPU/TPU friendly; (ii) reduces the cardinality of the dataset leading to efficient training; and (iii) enables models to be trained on *fuzzy sequences*, hopefully leading to robust learning.

Borgeaud et al., 2022; Zhai et al., 2022; Du et al., 2022), the quality of underlying data is often overlooked. Despite being an intuitive covariate in downstream model performance, there does not exist an efficient out-of-the-box solution for measuring the quality of a data point. Some popular heuristics (*e.g.*, data valuation (Ghorbani and Zou, 2019), coresets (Borsos et al., 2020a)) fall short from a variety of angles (Basu et al., 2021; Kumar et al., 2020; Toneva et al., 2019; Sener and Savarese, 2018).

Data distillation (DD) (see Sachdeva and McAuley (2023) for a comprehensive survey) offers a promising alternative to explicitly tagging the quality of each datapoint. Loosely, DD approaches aim to *synthesize* a terse data summary solely intended to train models to the same (if not better) quality as training them on the original dataset. In this chapter, we propose FARZI, a DD approach designed specifically for synthesizing high-fidelity auto-regressive data summaries. We call the data synthesized by FARZI as FARZI DATA.

FARZI DATA takes a step towards addressing the massive costs (*e.g.*, financial, environmental, *etc.*) associated with training large auto-regressive models (OpenAI, 2023; Anil et al.,

2023; Radford et al., 2022) on massive amounts of pretraining data by (i) implicitly *filtering* out low-quality sources of information resulting in a terse data summary, and (ii) *re-organizing* the data in a format that is most pertinent for model training. Intuitively, a vast majority of underlying *information* in such auto-regressive datasets is redundant from the downstream task's perspective. For example, looking at recommender systems, a predictive model wouldn't necessarily need trillions of event-level data from billions of users to accurately model user-behaviour patterns.

Typical DD techniques (Zhao and Bilen, 2023; 2021; Nguyen et al., 2021b; Cazenavette et al., 2022; Zhou et al., 2022; Deng and Russakovsky, 2022) are geared toward low-resolution image datasets due to (i) computationally expensive data optimization, and (ii) generation-friendly continuous domain of images (pixels). On the other hand, auto-regressive data generally consists of sequences of discrete *tokens* (*e.g.*, sub-words) with a potentially large vocabulary. Further, many applications call for sequences with a long list of such tokens. FARZI addresses the aforementioned characteristics of auto-regressive data by performing *data distillation in a latent space* by organizing FARZI DATA into (i) a *latent* data summary that captures the downstream task patterns, and (ii) a decoder (*e.g.*, token-embeddings) that maps the latent-space back to the token-space. In addition to making FARZI optimization-friendly (both of the aforementioned data components are non-discrete/continuous), we demonstrate that such latent-parameterization provably promotes implicit regularization when training models on FARZI DATA (Theorem 4.4.1). To summarize, we highlight four main contributions of this chapter:

- We develop FARZI, a scalable DD technique for summarizing massive auto-regressive datasets, and demonstrate FARZI DATA's sample efficiency over 5 datasets spanning sequential recommendation and language modeling tasks. Training on FARZI DATA, we are able to achieve up to $98-120\%$ of full-data performance for state-of-the-art models using as little as 0.1% of the original dataset size, as well as noting a strong cross-architecture generalization, *i.e.*, being able to train various (student) models on FARZI DATA synthesized using a given (teacher) model.
- Building atop the meta-matching framework of DD, we propose two crucial modifications for

largely improved sample efficiency. First, conducting an investigative study on the role of inner-loop optimizer in DD, we conclude Adam (Kingma and Ba, 2015) to be much more adept than SGD (with or without momentum) for auto-regressive DD. This is in stark contrast with existing DD and meta-learning studies where SGD is the de-facto optimizer of choice. We further improve FARZI's sample quality by leveraging a set of pretrained training trajectories in the underlying bilevel optimization.

- In addition to generating high-fidelity data, FARZI is computationally highly scalable. Firstly, parameterizing FARZI DATA into a latent data summary and a token decoder saves large amount of time and memory during optimization, thereby making FARZI (roughly) independent of the vocabulary size. Further, we derive an efficient reverse-mode differentiation of Adam which has a memory complexity independent of the number of inner-loop steps, unlike autograd systems which store all intermediate variables, therefore leading to more than $100\times$ memory footprint reduction.
- We provide a formal analysis of FARZI from various standpoints. We firstly show that FARZI DATA's latent parameterization implicitly promotes regularization and provably improves generalization. Previous studies have observed such *data overfitting* effects in DD empirically (Zhou et al., 2022), but we are the first to study its theoretical underpinnings. We further demonstrate the correctness of our proposed reverse-mode differentiation of Adam.

4.2 Related Work

4.2.1 Data downsampling

The complexity and training time for state-of-the-art models from different domains has grown exponentially in the recent years (OpenAI, 2023; Sun et al., 2019; Mittal et al., 2021; Rombach et al., 2022). Sampling has been the classic approach to summarize large datasets, approaches for which can be grouped into the following categories: **(i) Coreset construction** techniques which sample a weighted subset of the given dataset to accelerate model training

(Kaushal et al., 2019; Borsos et al., 2020b; Krause et al., 2021; Kazemi et al., 2021). Being a combinatorial optimization, coreset construction techniques typically leverage submodularity assumptions (Bilmes, 2022) to optimize the coreset in a tractable manner. **(ii) Data valuation** approaches which typically leverage shapley values (Shapley, 1953) to tag the *value* of each data point for model training (Wang and Jia, 2023; Ghorbani and Zou, 2019; Kwon and Zou, 2023; Kwon et al., 2021). Notably, such data valuation methods turn out to be computationally intractable even for moderate sized datasets. **(iii) Heuristic samplers** that build upon designing ad-hoc notions of data quality. Two prominent schools-of-thought in designing such heuristics has been to either preserve notions like diversity (Coleman et al., 2022; Abbas et al., 2023; Sorscher et al., 2022), discrepancy (Karnin and Liberty, 2019), *etc.* in some metric-space of the inputs, or use the loss-values from some proxy model to tag the difficulty (and thereby, quality) for each datapoint (Paul et al., 2021; Coleman et al., 2020; Sachdeva et al., 2021).

4.2.2 Data distillation

Contrary to sampling datapoints from a given dataset, data distillation approaches aim to *synthesize* high-quality data summaries for sample-efficient model training through bilevel optimization (see Sachdeva and McAuley (2023) for a comprehensive survey). Prominent existing approaches are designed for summarizing images (Wang et al., 2018b; Zhao et al., 2021; Zhao and Bilen, 2021; Cazenavette et al., 2022; Zhou et al., 2022; Deng and Russakovsky, 2022; Nguyen et al., 2021b), graphs (Jin et al., 2022a;c), and recommender systems (Sachdeva et al., 2022a). Such approaches can essentially be viewed as meta-learning approaches (see Hospedales et al. (2021) for a comprehensive survey) with the meta-optimization happening over the data summary instead of common applications like model initialization (Finn et al., 2017) or task hyper-parameters (Maclaurin et al., 2015b; Lorraine et al., 2020).

4.2.3 Autoregressive tasks

A variety of machine learning tasks are auto-regressive, *e.g.*, language modeling (OpenAI, 2023; Raffel et al., 2020), sequential recommendation (Sachdeva et al., 2019; Kang and McAuley, 2018), self-driving (Sachdeva et al., 2022b; Sun et al., 2020b), *etc.* Such tasks have a clear left-to-right causal structure with one event preceding the other, typically in time. Further, since a majority of such tasks are semi-supervised and are associated with large-amounts of data; training large foundation models on such datasets can become daunting despite its practicality, thereby limiting overall research progress. Concerningly, to the best of our knowledge, only simple *data sampling heuristics* scale to such large auto-regressive datasets (Toneva et al., 2019; Sener and Savarese, 2018).

4.3 FARZI: Autoregressive Data Distillation

4.3.1 Task & Notation

Given an autoregressive dataset $\mathcal{D} \triangleq \{\mathbf{x}_i\}_{i=1}^{|\mathcal{D}|}$ where $\mathbf{x}_i \triangleq [x_{ij} \in \mathcal{V}]_{j=1}^{|\mathbf{x}_i|}$ is an ordered sequence of tokens, each belonging to the vocabulary of all possible tokens $\mathcal{V}$. We aim to synthesize a data summary $\mathcal{D}_{\mathsf{syn}} \in \mathbb{R}^{\mu \times \xi \times \dim(\mathcal{V})}$ consisting of μ fake sequences of maximum length ξ, *s.t.*, $\mu \ll |\mathcal{D}|$. More specifically, we seek to construct $\mathcal{D}_{\mathsf{syn}}$ in such a way that a representative learning algorithm $\Phi_\theta : \mathcal{V}^n \mapsto \mathcal{V}$ trained on $\mathcal{D}_{\mathsf{syn}}$ using an autoregressive task (*e.g.*, next-token-prediction (Radford et al., 2018)) specified by a cost function $l : \mathcal{V} \times \mathcal{V} \mapsto \mathbb{R}$ can achieve performance equivalent to that of training Φ_θ on the original dataset $\mathcal{D}$. Taking next-token-prediction (Radford et al., 2018) as a representative predictive task, we denote the empirical risk as $\mathcal{L}_{\mathcal{D}}(\theta) \triangleq \mathbb{E}_{\mathbf{x} \sim \mathcal{D},\, x_i \sim \mathbf{x}}[l(\Phi_\theta(\mathbf{x}_{1:i}), x_{i+1})]$ for notational convenience, where $\mathbf{x}_{1:i}$ represents the sequence of first i tokens in $\mathbf{x}$.

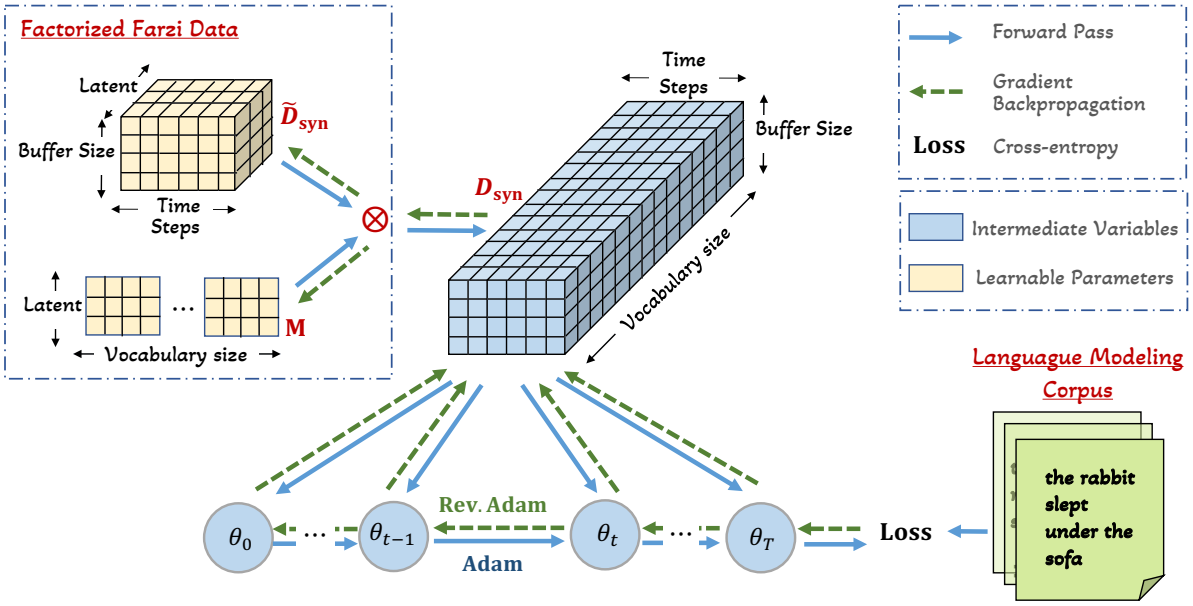

Figure 4.2. Visualization of a single outer-loop step in FARZI using the language modeling task. In this framework, each outer-loop step first materializes FARZI DATA using (a batch of) its respective low-rank counterparts, followed by training a learning algorithm on FARZI DATA for T-steps using Adam. The meta-gradient to update the factorized FARZI DATA is obtained using efficient reverse-mode Adam outlined in Algorithm 6. This process (outer-loop step) is repeated till convergence, or for a fixed number of iterations.

4.3.2 Methodology

We cast the problem of autoregressive DD as a meta-learning problem, wherein the *inner-loop* trains a learning algorithm on the data summary, and the *outer-loop* evaluates its *quality* via $l(\cdot,\cdot)$ on the original dataset to directly update the data summary via gradient descent. More formally, a naïve bilevel optimization problem can be framed as follows:

$$\arg\min_{\mathcal{D}_{\text{syn}}} \mathop{\mathbb{E}}_{\theta_0\sim\Theta}[\mathcal{L}_{\mathcal{D}}(\theta^*)] \quad \text{s.t.} \quad \theta^* \triangleq \arg\min_{\theta} \mathcal{L}_{\mathcal{D}_{\text{syn}}}(\theta \mid \theta_0) \ , \tag{4.1}$$

where Θ is a distribution to initialize model parameters (*e.g.*, uniform, Kaiming (He et al., 2015), *etc.*). Such a formulation is commonly termed as *meta-model matching based DD* (see Sachdeva and McAuley (2023) for a taxonomy of existing approaches), and is associated with significant computational complexity in terms of both time and memory. Typical approaches resort to

local optimization (*e.g.*, SGD) in the inner-loop, and Truncated Backpropagation Through Time (T-BPTT) by unrolling a finite number of inner optimization steps to obtain the meta-gradient. Notably, DD becomes infeasible — even after making such assumptions — when the data is autoregressive as each data-point (sequence) in the data summary ($\mathcal{D}_{\mathsf{syn}}$) is associated with (i) a large discrete token vocabulary, *i.e.*, $\dim(\mathcal{V})$; and (ii) a third sequential dimension, *i.e.*, ξ. Hence, the space complexity of existing DD techniques directly grows by a factor of $\approx \xi \cdot \dim(\mathcal{V})$.

To alleviate the computational challenges, FARZI performs *data distillation in a latent space*. More specifically, FARZI factorizes $\mathcal{D}_{\mathsf{syn}}$ into: (i) a latent data summary $\tilde{\mathcal{D}}_{\mathsf{syn}} \in \mathbb{R}^{\mu \times \xi \times d}$ where $d \ll \dim(\mathcal{V})$; and (ii) a token-decoder matrix $\mathbf{M} \in \mathbb{R}^{d \times \dim(\mathcal{V})}$. Finally, we can compose the latent data summary and the token-decoder to obtain the final data summary: $\mathcal{D}_{\mathsf{syn}} \equiv \text{softmax}(\tilde{\mathcal{D}}_{\mathsf{syn}} \cdot \mathbf{M} \;/\; T)$, where $T \in \mathbb{R}^{+}$ represents the temperature in $\text{softmax}(\cdot)$ and controls the entropy in $\mathcal{D}_{\mathsf{syn}}$. Such a factorization makes FARZI scalable to both extremely large datasets, *i.e.*, large $|\mathcal{D}|$ as well as datasets with large token vocabularies, *i.e.*, large $\dim(\mathcal{V})$.

In addition to promoting scalability, we prove that FARZI DATA's latent parameterization implicitly promotes regularization while training downstream models (Section 4.4). More specifically, we leverage the concept of *Rademacher complexity* (Bartlett and Mendelson, 2002) to show that explicit rank regularization while synthesizing data summaries (*e.g.*, latent factorization) promotes generalization. Notably such *data overfitting* has been previously (empirically) noted to notoriously affect DD (Zhou et al., 2022), but we are the first to explore the theoretical underpinnings.

Further, while typical bilevel optimization approaches almost all use SGD in the inner loop (Deng and Russakovsky, 2022) due to efficient reversible dynamics of SGD (see Maclaurin et al. (2015b) for efficient reverse-mode SGD), we empirically observe that in our setting of autoregressive DD, Adam optimization (Kingma and Ba, 2015) in the inner-loop is crucial for downstream DD performance (see Figure 4.5). Further, we also note that a significant number of inner-loop optimization steps — in the order of 100s — are needed for good generalization for both Adam and SGD based DD, as is concurrently reported by other work (Deng and Russakovsky,

Algorithm 6. Recursive reverse-mode differentiation of Adam.

Input: $\mathbf{w}_T$, $\mathbf{m}_T$, $\mathbf{v}_T$, α, ε, $L(w,x)$, meta-objective $f(w)$
Initialize: $d\mathbf{m} \triangleq df(\mathbf{w}_T)/d\mathbf{m}_0 \leftarrow 0,\ d\mathbf{x} \triangleq df(\mathbf{w}_T)/d\mathbf{x} \leftarrow 0,\ d\mathbf{w} \triangleq df(\mathbf{w}_T)/d\mathbf{w}_0 \leftarrow \nabla_{\mathbf{w}} f(\mathbf{w}_T)$
for $t = T$ **to** 1 **do**
  $\hat{\mathbf{m}}_t \triangleq \mathbf{m}_t/(1-\beta_1^t)$ ▷ exactly reverse Adam
  $\hat{\mathbf{v}}_t \triangleq \mathbf{v}_t/(1-\beta_2^t)$ ▷ exactly reverse Adam
  $\mathbf{w}_{t-1} = \mathbf{w}_t + \alpha \cdot \hat{\mathbf{m}}_t/(\hat{\mathbf{v}}_t + \varepsilon)$ ▷ exactly reverse Adam
  $\mathbf{g}_t \triangleq \nabla_{\mathbf{w}} L(\mathbf{w}_{t-1}, \mathbf{x})$ ▷ exactly reverse Adam
  $\mathbf{m}_{t-1} = [\mathbf{m}_t - (1-\beta_1)\cdot \mathbf{g}_t]/\beta_1$ ▷ exactly reverse Adam
  $\mathbf{v}_{t-1} = [\mathbf{v}_t - (1-\beta_2)\cdot \mathbf{g}_t^2]/\beta_2$ ▷ exactly reverse Adam
  $\varepsilon' \triangleq \varepsilon \cdot \sqrt{1-\beta_2^t}$
  $\alpha' \triangleq \alpha \cdot \sqrt{1-\beta_2^t} \ / \ (1-\beta_1^t)$
  $\beta' \triangleq (1-\beta_2) \ / \ (1-\beta_1)$
  $d\mathbf{m} = d\mathbf{m} + \alpha' \cdot \left(\frac{\beta' \cdot \mathbf{m}_t \cdot \mathbf{g}_t}{\sqrt{\mathbf{v}_t}\cdot(\sqrt{\mathbf{v}_t}+\varepsilon')^2} - \frac{1}{\sqrt{\mathbf{v}_t}+\varepsilon'} \right) \cdot d\mathbf{w}$ ▷ Proposition 4.4.7
  $d\mathbf{w} = d\mathbf{w} - (1-\beta_1)\cdot d\mathbf{m} \cdot \nabla_{\mathbf{w}} \nabla_{\mathbf{w}} L(\mathbf{w}_{t-1}, \mathbf{x})$ ▷ Hessian-vector product
  $d\mathbf{x} = d\mathbf{x} - (1-\beta_1)\cdot d\mathbf{m} \cdot \nabla_{\mathbf{x}} \nabla_{\mathbf{w}} L(\mathbf{w}_{t-1}, \mathbf{x})$ ▷ Hessian-vector product
  $d\mathbf{m} = \beta_1 \cdot d\mathbf{m}$
Output: gradient of $f(\mathbf{w}_T)$ *w.r.t.* $\mathbf{w}_0$, $\mathbf{m}_0$, and $\mathbf{x}$ (*i.e.*, $d\mathbf{w}$, $d\mathbf{m}$, $d\mathbf{x}$)

2022). To this end, we derive an efficient approximation of reverse-mode differentiation of the Adam optimization in Algorithm 6.

Algorithm 6 allows the memory footprint of the meta-gradient computation to be constant *w.r.t.* the number of inner-loop steps. Notably, meta-gradient computation is the biggest contributor in a meta-learning algorithm's overall scalability. This is in stark contrast with typical autograd libraries like PyTorch (Paszke et al., 2019), `JAX` (Bradbury et al., 2018), *etc.* which require storing all intermediate variables across the inner-optimization to compute the meta-gradient, resulting in a linearly growing memory footprint *w.r.t.* the number of inner-loop steps.

FARZI also improves the sample-efficiency of the underlying meta-matching framework (Equation (4.1)) by leveraging access to a limited number of training trajectories on the target dataset. Formally, let $\Omega \triangleq \{[\theta_i]_{i=1}^{T} \mid \theta_0 \sim \Theta\}$ be the *flat* set of all episodic checkpoints of training Φ_θ on $\mathcal{D}$ for a small number of random initializations. FARZI leverages Ω in its final optimization

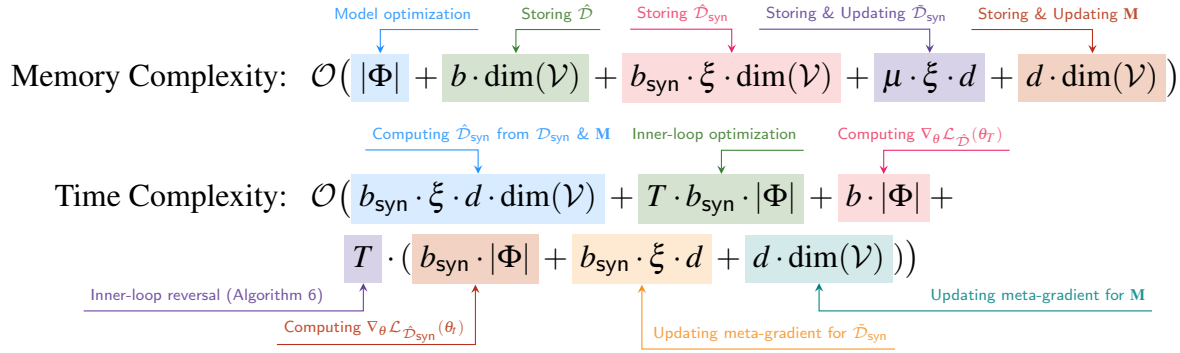

Figure 4.3. Computational complexity of a single outer-loop step in FARZI. $\hat{\mathcal{D}} \sim \mathcal{D}$ and $\hat{\mathcal{D}}_{\text{syn}} \sim \mathcal{D}_{\text{syn}}$ are batches of real data and FARZI DATA such that $b \triangleq |\hat{\mathcal{D}}|$ and $b_{\text{syn}} \triangleq |\hat{\mathcal{D}}_{\text{syn}}|$; and $|\Phi|$ represents the total number of parameters in Φ.

as follows:

$$\begin{aligned} &\underset{\mathbf{M}, \tilde{\mathcal{D}}_{\text{syn}}}{\arg\min} \; \mathop{\mathbb{E}}_{\theta_i \sim \Omega} \left[\mathcal{L}_{\mathcal{D}}(\theta_{i+T}) \right] \\ \text{s.t.} \quad &\theta_{t+1} \leftarrow \text{Adam}\left(\theta_t, \nabla_\theta \mathcal{L}_{\mathcal{D}_{\text{syn}}}(\theta_t) \right) \\ &\mathcal{D}_{\text{syn}} \leftarrow \text{softmax}\left(\tilde{\mathcal{D}}_{\text{syn}} \cdot \mathbf{M} \;/\; \tau \right) \quad , \end{aligned} \tag{4.2}$$

where $\text{Adam}(\cdot,\cdot)$ represents the set of Adam update equations (Kingma and Ba, 2015), T represents the number of inner-loop optimization steps for each outer-loop step, and $\theta_i \sim \Omega$ indicates that we start the inner-loop from a random checkpoint in Ω (could be of varying quality) rather than initializing randomly. Notably, curating Ω is independent of the DD optimization and can be precomputed beforehand, contributing nothing to FARZI's computational complexity.

In addition to the inner-loop, note that the outer-loop is also optimized via simple gradient descent (Adam) on $\tilde{\mathcal{D}}_{\text{syn}}$ and $\mathbf{M}$. To reiterate, we efficiently compute the meta-gradients using Algorithm 6, *i.e.*, $d\mathcal{L}_{\mathcal{D}}(\theta_{i+T})/d\tilde{\mathcal{D}}_{\text{syn}}$ and $d\mathcal{L}_{\mathcal{D}}(\theta_{i+T})/d\mathbf{M}$. In terms of the notation used in Algorithm 6, we set $\mathbf{w}_T \leftarrow \theta_{i+T}$, $\mathbf{m}_T$ and $\mathbf{v}_T$ to be the first- and second-order Adam momentums at the end of the inner-loop, the meta-objective $f(\cdot) \leftarrow \mathcal{L}_{\mathcal{D}}(\cdot)$, and compute the meta-gradient once for $\mathbf{x} \leftarrow \tilde{\mathcal{D}}_{\text{syn}}$ and once for $\mathbf{x} \leftarrow \mathbf{M}$.

4.3.3 FARZI's computational complexity

We elucidate the complexity of optimizing Equation (4.2) in terms of a single outer-loop step's runtime and memory usage in Figure 4.3. Key things to note are (i) FARZI's memory complexity to be independent of the number of inner-loop steps T due to Algorithm 6's recursive update equations (keep updating $d\mathbf{m}, d\mathbf{x}, d\mathbf{w}$ while unrolling the inner-loop), and (ii) dominant portion of FARZI's time complexity (factor multiplied with T) is independent of large terms like $\xi \cdot \dim(\mathcal{V})$.

4.4 Formal Analysis

4.4.1 Radamacher Complexity Bounds

Theorem 4.4.1. *Let $\mathcal{D}_{\mathsf{syn}} \in \mathbb{R}^{\mu \times \xi \times \dim(\mathcal{V})}$ be parameterized using $\tilde{\mathcal{D}}_{\mathsf{syn}} \in \mathbb{R}^{\mu \times \xi \times d}$ and $\mathbf{M} \in \mathbb{R}^{d \times \dim(\mathcal{V})}$, and $\mathcal{D}_{\mathsf{naive}} \in \mathbb{R}^{\mu \times \xi \times \dim(\mathcal{V})}$ denote the non-parameterized data. Let $\mathcal{F}$ be the function-class of linear classifiers, and $R_n(\mathcal{F}; \mathcal{D})$ denote the Rademacher complexity of a training set $\mathcal{D}$ (lower is less prone to overfitting). Then $R_n(\mathcal{F}; \mathcal{D}_{\mathsf{syn}} \cdot \mathbf{M}) < R_n(\mathcal{F}; \mathcal{D}_{\mathsf{naive}})$ whenever $\|\mathcal{D}_{\mathsf{syn}}\|_F < \|\mathcal{D}_{\mathsf{naive}}\|_F / \sqrt{2}$.*

Preliminaries. The Rademacher complexity is a classical way to express the "capacity" of a class of functions (Bartlett and Mendelson, 2002). Rademacher complexity is motivated by the intuition that a highly-expressive (high capacity) model is capable of fitting random labels just as well as structured ones. This complexity measure is useful for understanding various phenomena in machine learning. For example, the Rademacher complexity (plus a probabilistic failure-rate term that decays with n) is an upper bound on the generalization error in the PAC learning framework. It can be shown that *regularizers* reduce the Rademacher complexity – and thus the generalization error – of a class of models, e.g., for ℓ_p-regularized linear models (Kakade et al., 2008). There is a sizable literature focused on bounding the Rademacher complexity for various function classes and types of implicit / explicit regularization.

In the following analysis, we will show that the low-rank constraint in FARZI DATA implicitly regularizes the models trained on that data (i.e., reduces their Rademacher complexity).

The Rademacher complexity of a hypothesis class $\mathcal{F}$ is defined in terms the distribution $\mathcal{P}$ from which the dataset $\mathcal{D}$ is drawn.

Definition 4.4.2. (Empirical and Expected Rademacher complexity) For a given given function-class $\mathcal{F}$ and train-set $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$, the empirical Rademacher complexity $\hat{R}_n(\mathcal{D};\mathcal{F})$ is

$$\hat{R}_n(\mathcal{F};\mathcal{D}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(x_i)\right]$$

where $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_n]$ is a vector of i.i.d. draws from the Rademacher distribution. The expected Rademacher complexity $R_n(\mathcal{F};\mathcal{D})$ is simply the expectation over the data distribution, i.e., $R_n = \mathbb{E}_{\mathcal{P}}[\hat{R}_n]$.

Proof. In the case of Theorem 4.4.1, we are interested in the Rademacher complexities of linear functions trained on $\mathcal{D}_{\mathsf{naive}}$ and $\mathcal{D}_{\mathsf{syn}}$. For the purposes of this analysis, we will "flatten" the sequences so that we have $\mathcal{D}_{\mathsf{naive}} \in \mathbb{R}^{\mu\times(\xi \dim(\mathcal{V})}$.

Similarly, we will treat $\tilde{\mathcal{D}}_{\mathsf{syn}}$ and $\mathbf{M}$ as being "flattened" so that $\tilde{\mathcal{D}}_{\mathsf{syn}} \in \mathbb{R}^{\mu\times(\xi d)}$, $M \in \mathbb{R}^{d\times\dim(\mathcal{V})}$, and the distilled data matrix is:

$$\mathcal{D}_{\mathsf{syn}} = \mathrm{softmax}(\tilde{\mathcal{D}}_{\mathsf{syn}} \cdot \mathbf{M}/T)$$

where T is the softmax temperature. We will use a recent result from Awasthi et al. (2020), who study upper and lower bounds on $\hat{R}_n$ for linear functions. We obtain the following lemma by setting $p = 2$ in Theorems 1 and 2 of their paper.

Lemma 4.4.3. (Theorems 1 and 2 in Awasthi et al. (2020), with $p = 2$) *Let $\mathcal{F}_\lambda$ be the set of ℓ_2-norm bounded linear models $\mathcal{F}_\lambda = \{\mathbf{x}^\top \mathbf{w} : \|w\|_2 \leq \lambda\}$. Then the empirical Rademacher*

complexity obeys the following inequality, where $\|\mathcal{D}\|_F$ is the Frobenius norm of the data matrix.

$$\frac{\lambda}{\sqrt{2n}}\|\mathcal{D}\|_F \leq \hat{R}_n(\mathcal{F};\mathcal{D}) \leq \frac{\lambda}{\sqrt{n}}\|\mathcal{D}\|_F$$

Our proof strategy is to show an upper bound for $\hat{R}_n(\mathcal{F};\mathcal{D}_{\mathsf{syn}})$ and a lower bound for $\hat{R}_n(\mathcal{F};\mathcal{D}_{\mathsf{naive}})$. By analyzing the case where the lower bound exceeds the upper bound, we can understand when the low-rank constraint implicitly regularizes.

The following upper and lower bounds are a direct consequence of Lemma 4.4.3. The second equality comes from the equivalence between the Frobenius norm and the sum of eigenvalues σ_i.

$$\hat{R}_n(\mathcal{F};\mathcal{D}_{\mathsf{naive}}) \geq \frac{\lambda}{\sqrt{2n}}\|\mathcal{D}_{\mathsf{naive}}\|_F = \frac{\lambda}{\sqrt{2n}} \sum_{i=1}^{\mathrm{Rank}(\mathcal{D}_{\mathsf{naive}})} \sigma_i^2(\mathcal{D}_{\mathsf{naive}}) \tag{4.3}$$

$$\hat{R}_n(\mathcal{F};\mathcal{D}_{\mathsf{syn}}) \leq \frac{\lambda}{\sqrt{n}}\|\mathcal{D}_{\mathsf{syn}}\|_F = \frac{\lambda}{\sqrt{n}} \sum_{i=1}^{\mathrm{Rank}(\mathcal{D}_{\mathsf{syn}})} \sigma_i^2(\mathcal{D}_{\mathsf{syn}}) \tag{4.4}$$

Taken together, these two bounds imply that $\hat{R}_n(\mathcal{F};\mathcal{D}_{\mathsf{naive}})$ is larger than $\hat{R}_n(\mathcal{F};\mathcal{D}_{\mathsf{syn}})$ whenever

$$\|\mathcal{D}_{\mathsf{syn}}\|_F \leq \frac{1}{\sqrt{2}}\|\mathcal{D}_{\mathsf{naive}}\|_F \tag{4.5}$$

□

4.4.2 Realistic Conditions

We theoretically characterize a setting where the conditions of Theorem 4.4.1 hold. We consider a task where each element of $\mathcal{D}_{\mathsf{naive}} \in \{0,1\}$, $\mathcal{D}_{\mathsf{syn}} \in [0,1]$, and the vocabulary dimensions of both datasets sum to 1. This setting corresponds to the practical situation where $\mathcal{D}_{\mathsf{naive}}$ is a dataset of 1-hot coded sequences and $\mathcal{D}_{\mathsf{syn}}$ is a dataset of soft sequences, where the 1-hot is replaced by a vocabulary distribution.

Lemma 4.4.4. *Let $\mathcal{D}_{\mathsf{naive}}$ be a 1-hot sequence dataset and $\mathcal{D}_{\mathsf{syn}}$ be a soft sequence dataset. The conditions of Theorem 4.4.1 hold if* $\max(\mathcal{D}_{\mathsf{syn}}) < \left(\sqrt{3}-1\right)/2 \approx 0.366$.

We are interested in practical settings where the conditions of Theorem 4.4.1 can be expected to hold. As an easy warmup, consider the scenario where $\mathcal{D}_{\mathsf{naive}}$ and $\mathcal{D}_{\mathsf{syn}}$ are full-rank matrices with all singular values equal to 1. In this case, $\text{Rank}(\mathcal{D}_{\mathsf{syn}}) < \xi d$ and $\text{Rank}(\mathcal{D}_{\mathsf{naive}}) = \xi \mathcal{V}$, making it easy to see that the Rademacher complexity decreases when the low-rank dimension $d < \mathcal{V}/\sqrt{2}d$.

To analyze the more practical scenario described in the main text, we require the following technical lemma.

Lemma 4.4.5. *Given a vector $\mathbf{x} \in \mathbb{R}^d$ and $\alpha \in [0,1]$, if*

$$\|\mathbf{x}\|_\infty < \frac{1}{2}\left(\alpha c \pm \sqrt{c}\sqrt{4\alpha + c}\right)$$

then $\|\mathbf{x}\|_2^2 \leq \alpha \|\mathbf{x}\|_1^2$.

Proof. To simplify the notation, let

$$\beta = \|\mathbf{x}\|_\infty \qquad c = \|\mathbf{x}\|_1$$

Because β is the largest element of $\mathbf{x}$, $|x_i| \leq \beta$ for all $i > 1$.

We wish to find (and bound) the value of $\sum_{i=1}^d |x_i|^2$ subject to the constraint that $|x_i| \leq \beta$ and $\sum_{i=1}^d |x_i| = c$. The solution to this optimization problem is to set as many $x_i = \beta$ as possible, with one remaining component containing the remainder.

There are $\lfloor c/\beta \rfloor$ components that contribute β^2 to $\|\mathbf{x}\|_2^2$ and one component that contributes the remainder (which is smaller than β^2). This gives the following bound:

$$\|\mathbf{x}\|_2^2 \leq \beta^2 \left(\left\lfloor \frac{c}{\beta} \right\rfloor + 1\right) \leq \beta^2 \left(\frac{c}{\beta} + 1\right) = \beta c + \beta^2$$

Therefore, it is sufficient to find a value of β which satisfies the following inequality.

$$\beta^2 + \beta c \leq \alpha c$$

$$\beta^2 + \beta c - \alpha c \leq 0$$

The roots of this quadratic are $\beta = \frac{1}{2}\left(-c \pm \sqrt{c}\sqrt{4\alpha + c}\right)$. Because the quadratic is convex, the expression is negative between the roots. Furthermore, the smaller root is negative for all $\alpha > 0$ and $\beta = \|\mathbf{x}\|_\infty > 0$. Therefore, to satisfy the inequality, we only require

$$\|\mathbf{x}\|_\infty < \frac{1}{2}\left(-c \pm \sqrt{c}\sqrt{4\alpha + c}\right)$$

□

We are now ready to prove Lemma 4.4.4. Recall that $\mathcal{D}_{\mathsf{syn}}$ is the post-softmax data summary, i.e.

$$\mathcal{D}_{\mathsf{syn}} = \text{softmax}(\tilde{\mathcal{D}}_{\mathsf{syn}} \cdot \mathbf{M}/T)$$

For this reason, the sum of the elements of $\mathcal{D}_{\mathsf{syn}}$ over the last dimension (the vocabulary dimension) is one. As stated in Lemma 4.4.4, we assume that $\mathcal{D}_{\mathsf{naive}}$ is a binary 1-hot matrix, i.e.

$$\mathcal{D}_{\mathsf{naive}} \in \{0,1\}^{\mu \times \xi \times \dim(\mathcal{V})}$$

For Theorem 4.4.1 to hold, we require that

$$\|\mathcal{D}_{\mathsf{syn}}\|_F \leq \frac{1}{\sqrt{2}}\|\mathcal{D}_{\mathsf{naive}}\|_F$$

or, equivalently:

$$\|\mathcal{D}_{\mathsf{syn}}\|_F^2 \leq \frac{1}{2}\|\mathcal{D}_{\mathsf{naive}}\|_F^2$$

Because $\mathcal{D}_{\mathsf{naive}}$ is 1-hot,

$$\|\mathcal{D}_{\mathsf{naive}}\|_F^2 = \mu\xi$$

Using the technical lemma with $\alpha = 1/2$ and $\|\mathbf{x}\|_1 = 1$, if the maximal entry in each vocabulary distribution of $\mathcal{D}_{\mathsf{syn}}$ is smaller than $\frac{\sqrt{3}-1}{2}$, then each vocabulary row of $\tilde{\mathcal{D}}_{\mathsf{syn}}$ contributes at most $\frac{1}{2}$ to the Frobenius norm. This results in the following expression, where the second inequality comes from the application of the technical lemma.

$$\begin{aligned}
\|\mathcal{D}_{\mathsf{syn}}\|_F^2 &= \sum_{i=1}^{\mu}\sum_{j=1}^{\xi} \|\mathcal{D}_{\mathsf{syn}}(i,j,:)\|_2^2 \\
&\leq \sum_{i=1}^{\mu}\sum_{j=1}^{\xi} \|\mathcal{D}_{\mathsf{syn}}(i,j,:)\|_2^2 \\
&\leq \frac{1}{2}\|\mathcal{D}_{\mathsf{syn}}(i,j,:)\|_1^2 = \frac{1}{2}\mu\xi = \frac{1}{2}\|\mathcal{D}_{\mathsf{naive}}\|_F^2
\end{aligned}$$

which is true whenever $\max \mathcal{D}_{\mathsf{syn}} < \left(\sqrt{3}-1\right)/2 \approx 0.366$.

Using Lemma 4.4.4, we can demonstrate that both temperature and latent dimension induce regularization (under a mild boundedness condition for $\mathbf{M}$ and $\tilde{\mathcal{D}}_{\mathsf{syn}}$) by "smoothing out" the probabilities.

Theorem 4.4.6. *Let $\mathcal{D}_{\mathsf{naive}}$ be a 1-hot dataset, $\tilde{\mathcal{D}}_{\mathsf{syn}} \in [-B,B]^{\mu\times\xi\times d}$, $\mathbf{M} \in [-1,1]^{d\times\dim(\mathcal{V})}$, and T be the softmax temperature. Theorem 4.4.1 holds if $\dim(\mathcal{V}) \geq 3$ and $d/T \leq \frac{1}{2B}\log\dim(\mathcal{V})$.*

Proof. Recall that $\mathcal{D}_{\mathsf{syn}} = \mathrm{softmax}(\tilde{\mathcal{D}}_{\mathsf{syn}} \cdot \mathbf{M}/T)$, where

$$\tilde{\mathcal{D}}_{\mathsf{syn}} = [-R,R]^{\mu\times\xi\times d}$$

$$M = [-1,1]^{d\times\dim(\mathcal{V})}$$

We wish to apply Lemma 4.4.4 to understand what choices of the temperature T and latent dimension d will lead to regularization. Let $\mathbf{p}^{(n,m)} = \mathcal{D}_{\mathsf{syn}}[n,m,:]$ (post-softmax probabilities)

and $\mathbf{z}^{(n,m)} = \left(\tilde{\mathcal{D}}_{\mathsf{syn}} \cdot \mathbf{M}\right)[n,m,:]/T$ (pre-softmax logits) to simplify the notation. We will drop the superscripts when the context is obvious. For Lemma 4.4.4 to hold, the following inequality must hold for all (n,m).

$$\mathbf{p}^{(n,m)} \leq \left(\sqrt{3}-1\right)/2$$

By definition,

$$\mathbf{p}_i = \frac{e^{\mathbf{z}_i/T}}{\sum_{j=1}^{\dim(\mathcal{V})} e^{\mathbf{z}_j/T}} = \frac{1}{1+\sum_{\substack{j=1^{\dim(\mathcal{V})} \\ j\neq i}} e^{(z_j-z_i)/T}}$$

Because each entry of $\tilde{\mathcal{D}}_{\mathsf{syn}} \in [-B,B]$ and each entry of $M \in [-1,1]$, $-dB \leq z_i \leq dB$. Therefore, $z_j - z_i \geq -2dB$ and the following inequality holds.

$$\mathbf{p}_i = \frac{1}{1+\sum_{\substack{j=1^{\dim(\mathcal{V})} \\ j\neq i}} e^{(z_j-z_i)/T}} \leq \frac{1}{1+\sum_{\substack{j=1^{\dim(\mathcal{V})} \\ j\neq i}} e^{-2dB/T}} = \frac{1}{1+(\dim(\mathcal{V})-1)\,e^{-2dB/T}}$$

Therefore, it is sufficient that

$$\frac{1}{1+\sum_{\substack{j=1^{\dim(\mathcal{V})} \\ j\neq i}} e^{-2dB/T}} = \frac{1}{1+(\dim(\mathcal{V})-1)\,e^{-2dB/T}} \leq \frac{\sqrt{3}-1}{2}$$

Using the notation τ to represent the threshold $\tau = \left(\sqrt{3}-1\right)/2 \approx 0.336$, we have:

$$\frac{1}{1+(\dim(\mathcal{V})-1)\,e^{-2dB/T}} \leq \tau$$

which holds if the following holds.

$$\frac{1-\tau}{(\dim(\mathcal{V})-1)\,\tau} \leq e^{-2dB/T}$$

$$\log\left(\frac{1-\tau}{(\dim(\mathcal{V})-1)\,\tau}\right) \leq -2dB/T$$

Provided that $\dim(\mathcal{V}) > 1+\frac{1-\tau}{\tau} = 1+\frac{3-\sqrt{3}}{\sqrt{3}-1} \approx 2.74$ (which is satisfied because $\dim(\mathcal{V}) \geq 3$), the argument to the logarithm is < 1 and thus the logarithm is negative. As an aside, this condition makes sense because we need more than 3 target vocabulary tokens over which to spread the softmax probability if we are to have each token's probably be < 0.336.

The following condition is sufficient for the guarantees to hold. All we have done is to multiply both sides of the inequality by -1 (and flip the direction).

$$\log\left(\frac{(\dim(\mathcal{V})-1)\,\tau}{1-\tau}\right) \geq 2dB/T$$

This results in the following condition, which is sufficient for Theorem 4.4.1 to hold.

$$\begin{aligned}
\frac{d}{T} &\leq \frac{1}{2B}\log\left(\frac{(\dim(\mathcal{V})-1)\,\tau}{1-\tau}\right) \\
&\leq \frac{1}{2B}\log\left((\dim(\mathcal{V})-1)\frac{\tau}{1-\tau}\right) \\
&= \frac{1}{2B}\log\left((\dim(\mathcal{V})-1)\frac{3-\sqrt{3}}{\sqrt{3}-1}\right)
\end{aligned}$$

To get a slightly looser (but simpler) version of this inequality, first we observe that

$$\dim(\mathcal{V}) \leq (\dim(\mathcal{V})-1)\frac{3-\sqrt{3}}{\sqrt{3}-1}$$

whenever $\dim(\mathcal{V}) \geq (3-\sqrt{3})/(4-2\sqrt{3}) \approx 2.366$, which is true because $\dim(\mathcal{V}) \geq 3$. This results in the following sufficient condition.

$$\frac{d}{T} \leq \frac{1}{2B}\log\dim(\mathcal{V})$$

□

Table 4.1. Statistics of the meta-optimized `[10 x 150]` FARZI DATA ($\mathcal{D}_{\text{syn}}$) synthesized via factorized *vs.* non-factorized parameterizations for the ML-100k dataset. We also list the performance of models trained on each FARZI DATA parameterization. See Theorem 4.4.1 for a formal interpretation of these results. Most significantly, a lower $||\mathcal{D}_{\text{syn}}||_F^2$ signifies higher implicit regularization.

Parameterization	**Data Summary Statistics**			**Performance of Models trained on Data Summary ($\mathcal{D}_{\text{syn}}$)**				
	$\max(\mathcal{D}_{\text{syn}})$	$\|\|\mathcal{D}_{\text{syn}}\|\|_F^2$	$\max(\sigma(\mathcal{D}_{\text{syn}}))$	HR@10	HR@100	nDCG@10	nDCG@100	AUC
One-hot Initialization	1	511.99	1			—		
Factorized $\mathcal{D}_{\text{syn}} \triangleq \text{softmax}(\tilde{\mathcal{D}}_{\text{syn}} \cdot \mathbf{M})$	0.731	18.92	0.612	18.45	61.93	9.16	17.54	0.9041
Non-Factorized $\mathcal{D}_{\text{syn}} \triangleq \text{softmax}(\tilde{\mathcal{D}}_{\text{syn}})$	0.836	51.72	0.693	17.81	60.65	9.03	17.04	0.8960

4.4.3 Empirical Validation of Theorem 4.4.1

To better understand the practical significance of the result in Theorem 4.4.1, we devise a simple empirical experiment. We compare the Frobenius norm (and some other statistics) of the optimized data summaries obtained via FARZI, under the following circumstances:

1. The synthesized data is factorized (the standard setup in FARZI) as in Equation (4.2), *i.e.*, $\mathcal{D}_{\text{syn}} \triangleq \text{softmax}(\tilde{\mathcal{D}}_{\text{syn}} \cdot \mathbf{M})$.

2. The synthesized data is not factorized, and we optimize the full-rank $\tilde{\mathcal{D}}_{\text{syn}} \in \mathbb{R}^{\mu \times \xi \times |\mathcal{V}|}$ (computationally intractable for large datasets). To make the data amenable for model training, we take a softmax on $\tilde{\mathcal{D}}_{\text{syn}}$, *i.e.*, $\mathcal{D}_{\text{syn}} \triangleq \text{softmax}(\tilde{\mathcal{D}}_{\text{syn}})$. This emprically vastly improved the optimization stability and final performance.

Note that we used temperature $T = 1$ in the softmax$(\cdot)$ for both factorized and non-factorized FARZI DATA setups for convenience. From the results in Table 4.1, we do confirm that factorized parameterization of FARZI DATA does lead to data summaries with a significantly lower Frobenius norm, compared to both the original one-hot data representation as well as if we conducted FARZI using a non-factorized parameterization.

This has two implications. First for Theorem 4.4.1 to hold, we need for the value of $\|\mathcal{D}_{\mathsf{syn}}\|_F^2$ for the factorized FARZI DATA to be no more than $1/2\|\mathcal{D}_{\mathsf{syn}}\|_F^2$ of the non-factorized FARZI DATA. This is indeed the case for the results in Table 4.1.

Second, we confirm that our theoretical intuitions carry over into practice. The second column ("Performance") contains the results of training the same SASRec models as our main evaluation on the factorized and non-factorized distillation data. The factorized representation of FARZI DATA leads to better quality models than the non-factorized FARZI DATA (Table 4.1), likely because of the stronger implicit regularization with a low latent dimension.

4.4.4 Correctness of Reverse-mode Adam

Proposition 4.4.7. *Correctness of Algorithm 6, line 13.*

Proof. Using the chain rule of derivatives:

$$\begin{aligned}
\frac{df(\mathbf{w}_T)}{d\mathbf{m}_0} \triangleq d\mathbf{m} &= d\mathbf{m} + \frac{\partial w_t}{\partial m_t} \cdot d\mathbf{w} \\
&= d\mathbf{m} - \frac{\alpha}{1-\beta_1^t} \left(\frac{\left[\sqrt{\hat{\mathbf{v}}_t} + \varepsilon\right] - \left[\mathbf{m}_t \cdot \left(\frac{\partial \mathbf{v}_t / \partial \mathbf{m}_t}{2 \cdot \sqrt{\hat{\mathbf{v}}_t} \cdot (1-\beta_2^t)}\right)\right]}{(\sqrt{\hat{\mathbf{v}}_t} + \varepsilon)^2} \right) \cdot d\mathbf{w} \\
&= d\mathbf{m} + \alpha' \cdot \left(\frac{\left[\mathbf{m}_t \cdot \left(\frac{\partial \mathbf{v}_t / \partial \mathbf{m}_t}{2 \cdot \sqrt{\mathbf{v}_t}}\right)\right]}{(\sqrt{\mathbf{v}_t} + \varepsilon')^2} - \frac{1}{\sqrt{\mathbf{v}_t} + \varepsilon'} \right) \cdot d\mathbf{w} \quad ,
\end{aligned}$$

where, $\varepsilon' \triangleq \varepsilon \cdot \sqrt{1-\beta_2^t}$ and $\alpha' \triangleq \frac{\alpha \cdot \sqrt{1-\beta_2^t}}{1-\beta_1^t}$.

Using the chain rule again:

$$\frac{\partial \mathbf{v}_t}{\partial \mathbf{m}_t} = \frac{\partial \mathbf{v}_t / \partial \mathbf{g}_t}{\partial \mathbf{m}_t / \partial \mathbf{g}_t} = \frac{2 \cdot (1-\beta_2) \cdot \mathbf{g}_t}{(1-\beta_1)} = 2 \cdot \beta' \cdot \mathbf{g}_t \quad ,$$

where, $\beta' \triangleq \frac{1-\beta_2}{1-\beta_1}$, leading to finally:

$$\frac{df(\mathbf{w}_T)}{d\mathbf{m}_0} \triangleq d\mathbf{m} = d\mathbf{m} + \alpha' \cdot \left(\frac{\left[\mathbf{m}_t \cdot \left(\frac{\partial \mathbf{v}_t / \partial \mathbf{m}_t}{2 \cdot \sqrt{\mathbf{v}_t}} \right) \right]}{(\sqrt{\mathbf{v}_t} + \varepsilon')^2} - \frac{1}{\sqrt{\mathbf{v}_t} + \varepsilon'} \right) \cdot d\mathbf{w}$$
$$= d\mathbf{m} + \alpha' \cdot \left(\frac{\beta' \cdot \mathbf{m}_t \cdot \mathbf{g}_t}{\sqrt{\mathbf{v}_t} \cdot (\sqrt{\mathbf{v}_t} + \varepsilon')^2} - \frac{1}{\sqrt{\mathbf{v}_t} + \varepsilon'} \right) \cdot d\mathbf{w}$$

□

4.5 Empirical Setup

4.5.1 Predictive Tasks

We empirically evaluate FARZI's practicality over two well-studied autoregressive predictive tasks:

- *Sequential Recommendation:* Predict the *item* that a user is most likely to consume next, given their historic item consumption history. We use four datasets: Movielens-100k, Movielens-1M (Harper and Konstan, 2015), Amazon Magazine (Ni et al., 2019a), Netflix (Bennett and Lanning, 2007); and use AUC, HitRate, and nDCG to evaluate model quality.
- *Language Modeling (LM):* Predict the most probable following word given a sequence of words. We conduct our experiments on the official-released train/validation/test split of the English Penn Treebank (PTB) corpus (Marcus et al., 1993): an open-sourced benchmark widely used for LM. We evaluate our models using word-level perplexity, as well as the token prediction accuracy after greedy decoding on the test set.

4.5.2 Models

We use SASRec (Kang and McAuley, 2018) and a small Transformer model (Vaswani et al., 2017) as the representative learning algorithms (Φ) in FARZI's inner-loop for sequential recommendation and LM tasks respectively, and use cross-entropy as the underlying objective

Table 4.2. List of all hyper-parameters combinations tried for FARZI and other baselines for sequential recommendation.

Hyper-Parameter		Model	Magazine	ML-100k	ML-1M	Netflix
Latent size		SASRec GRU4Rec FMLP	{8, 16, 32, 50, 64, 128}			
# Layers		SASRec GRU4Rec FMLP	{1, 2}			
Attention Heads		SASRec FMLP	{1, 2}			
Learning rate		SASRec GRU4Rec FMLP	{0.01, 0.02, 0.05} {0.01, 0.02, 0.05} {0.0001, 0.0002, 0.0005}			
Dropout		SASRec GRU4Rec FMLP FARZI	{0.0, 0.2, 0.4}			
ξ		FARZI	{10, 20}	{50, 100, 150}	{50, 100, 150}	200
d		FARZI	8	8	32	32
τ		FARZI	{0.5, 1, 2}			
$\lvert\Omega\rvert$		FARZI	100	100	100	50
Inner loop	Weight Decay Learning Rate # Steps β_1 β_2 SGD Momentum	FARZI	$\{0, 10^{-6}\}$ {0.01, 0.02} {100, 200, 300} 0.9 0.999 {0.5, 0.75, 0.9, 0.95, 0.99}			
Outer loop	Weight Decay Learning Rate # Steps	FARZI	$\{0, 10^{-6}, 10^{-4}\}$ 0.01 4000			
Batch size	$\mathcal{D}$	FARZI	512			
	$\mathcal{D}_{\text{syn}}$		—	—	50	25

Table 4.3. List of all hyper-parameters combinations tried for FARZI and other baselines for language modeling.

Hyper-Parameter		Model	Penn Treebank
Latent size		Transformer RNN	16
# Layers		Transformer RNN	1
Attention Heads		Transformer RNN	1
Learning rate		Transformer RNN	{0.01, 0.02, 0.05} {0.01, 0.02, 0.05}
Dropout		Transfomer RNN	{0.0, 0.2}
ξ		FARZI	{5, 15}
d		FARZI	8
τ		FARZI	{0.5, 1, 2}
$\lvert\Omega\rvert$		FARZI	400
Inner loop	Weight Decay Learning Rate # Steps β_1 SGD Momentum	FARZI	{0, 10^{-7}} {0.01, 0.02} {200, 300,400, 500, 600} 0.999 -
Outer loop	Weight Decay Learning Rate # Steps	FARZI	{0, 10^{-6}, 10^{-4}} 0.01 8000
Batch size	$\mathcal{D}$ $\mathcal{D}_{\text{syn}}$	FARZI	256 —

function for both. We implement FARZI using PyTorch (Paszke et al., 2019), and for the sake of better reproducibility, we list all hyper-parameter combinations tried for our experiments in Tables 4.2 and 4.3.

4.5.3 Metrics

We present a formal definition of all metrics used in this chapter for both sequential recommendation and language modeling tasks.

Sequential Recommendation. We use the AUC, HitRate (HR@k), and Normalized Discounted Cumulative Gain (nDCG@k) metrics, as earlier defined in Section 3.5.4.

Language Modeling. We first use Perplexity (PPL) to evaluate language modeling performance. Perplexity quantifies how uncertain the model is when trying to predict the next word in a sequence, given the previous words. Given a sentence $\vec{x_i}$, which is tokenized into a sequence of tokens $[w_1, w_2, \cdots, w_{|\vec{x_i}|}]$, the sentence PPL is defined as:

$$\log_2(\text{PPL}_i) \triangleq -\frac{1}{|\vec{x_i}|}\sum_{i}^{|\vec{x_i}|}\log_2 P(w_i|w_1, w_2, \cdots, w_{i-1})$$

where P is the probability assigned by the language model to w_i given the context of the previous words. Then, given a corpus $\mathcal{C}$ containing N sentences $\mathcal{C} = \{\vec{x_1}, \vec{x_2}, \cdots, \vec{x_N}\}$, the perplexity over $\mathcal{C}$ is defined as the average PPL over the sentence PPLs:

$$\text{PPL}_{\mathcal{C}} \triangleq \frac{1}{N} \cdot \sum_{i}^{N} \text{PPL}_i$$

To better evaluate the generation quality of a language model, we also evaluate the average top-1 predicted token accuracy after greedy decoding, similar to the HR@1 metric described earlier.

Table 4.4. Datasets used in this chapter as well as a brief set of statistics.

Dataset	# Users / # Sentences	# Items / # Unique tokens	# Interactions / # Total tokens	Seq. Length Mean / Median / Min
Amazon Magazine (Ni et al., 2019b)	3k	1.3k	12k	4.10 / 3 / 3
ML-100k (Harper and Konstan, 2015)	943	1.6k	100k	104.04 / 63 / 18
ML-1M (Harper and Konstan, 2015)	6k	3.7k	1M	165.22 / 95 / 20
Netflix (Bennett and Lanning, 2007)	476k	17k	100M	210.91 / 98 / 3
PTB (Marcus et al., 1993)	49k	10k	1M	22 / 21 / 2

4.5.4 Datasets

We list the datasets used in this chapter as well as brief data statistics in Table 4.4. We discuss other task-specific preprocessing and train/test splitting strategy below.

Sequential Recommendation. Owing to recent work (Sachdeva et al., 2022c), we follow the minimal amount of preprocessing by only removing the users with less than two total interactions. We simulate the train/test split from the strong-generalization school-of-thought (Liang et al., 2018), where we keep a completely disjoint set of $80/10/10\%$ train, validation, and test users split randomly. For each user in the validation/test-set, the chronologically last interacted item is used for computing ranking metrics, whereas all previous interactions are used as context for the model. Further, to simulate a realistic recommendation scenario, we compute all metrics on the full item-space without any down-sampling (Krichene and Rendle, 2020).

Language Modeling. We employ the Penn Treebank (PTB) dataset, an established and openly accessible benchmark extensively utilized in natural language processing and language modeling tasks, as introduced by (Marcus et al., 1993). We use the train/validation/test split of the official release. The original PTB corpus consists of more than 4.5 million words of American English, featuring a word vocabulary of 9,999 words, including the `<unk>` token. In our experimentation, we opt to maintain a vocabulary comprising 2,000 words with the highest frequencies, while any out-of-vocabulary words are represented as `<unk>`.

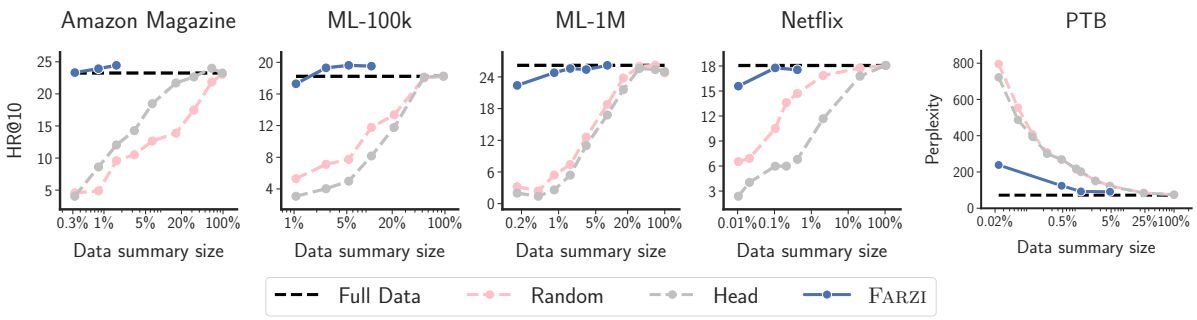

Figure 4.4. Performance change of SASRec (left four columns) and Transformer (rightmost column) with increasing data (log-scale) for recommendation and LM respectively. For a tabular version of these results see Table 4.5.

4.6 Experiments

4.6.1 How sample efficient is FARZI DATA?

We evaluate the fidelity of FARZI DATA by first optimizing for $\mathcal{D}_{\text{syn}}$ using Equation (4.2), followed by training Φ (from scratch) on $\mathcal{D}_{\text{syn}}$. We plot the performance of the trained Φ on the test-set for various amounts of data budgets (μ) in Figure 4.4 for sequential recommendation and LM tasks. A tabular version of the same results can be found in Table 4.5. We also plot semantically equivalent results for other commonly used data samplers, namely (i) random sampling: sample sequences uniformly at random, and (ii) head sampling: retain the sequences with the largest length. We first note that FARZI DATA is much more sample-efficient than other data sampling techniques, being able to achieve up to 1000$\times$ data compression with no loss in performance. Further, in Figure 4.4, we notice that on two out of the four recommendation datasets, FARZI's orders of magnitude smaller data is able to train models of higher quality than the original dataset itself. This observation acts as further evidence for the intuitive yet under-explored idea that less but high-quality data can be more important for model training than a very large but noisy dataset (Sachdeva et al., 2022a; Zhou et al., 2023).

Table 4.5. Performance change of SASRec & Transformer with various sizes of FARZI DATA for sequential recommendation & language modeling tasks respectively. The best result for each dataset & metric is colored **orange**.

Dataset & Model	FARZI DATA size	HR@10	HR@100	nDCG@10	nDCG@100	AUC	PPL	Acc.
Magazine & SASRec	[10 x 10] ≡ 0.3%	23.3	**52.3**	15.8	21.1	**0.8428**	-	-
	[25 x 20] ≡ 0.8%	23.9	**52.3**	16.5	21.6	0.8307	-	-
	[50 x 20] ≡ 1.6%	**24.5**	52.1	**17.1**	**22.1**	0.8291	-	-
	Full-data	23.2	52.0	16.9	21.7	0.8223	-	-
ML-100k & SASRec	[10 x 150] ≡ 1%	17.3	61.2	9.2	17.7	0.8957	-	-
	[25 x 150] ≡ 2.6%	19.3	61.6	9.9	17.7	0.902	-	-
	[50 x 50] ≡ 5.3%	**19.6**	**62.9**	9.9	**18.1**	**0.9016**	-	-
	[100 x 100] ≡ 10.6%	19.5	61.9	**10.1**	**18.1**	**0.9016**	-	-
	Full-data	18.2	60.6	9.3	17.6	0.9011	-	-
ML-1M & SASRec	[10 x 150] ≡ 0.1%	22.4	59.0	12.0	19.0	0.923	-	-
	[50 x 100] ≡ 0.8%	24.8	61.6	13.8	20.8	0.9301	-	-
	[100 x 100] ≡ 1.6%	25.6	**63.6**	14.1	21.3	**0.9317**	-	-
	[200 x 50] ≡ 3.3%	25.4	61.8	14.1	21.0	0.9315	-	-
	[500 x 50] ≡ 8.2%	**26.2**	61.0	13.8	20.7	0.9293	-	-
	Full-data	**26.2**	62.8	**14.4**	**21.8**	0.9291	-	-
Netflix & SASRec	[50 x 200] ≡ 0.01%	15.6	38.0	9.9	14.1	0.9235	-	-
	[500 x 200] ≡ 0.1%	17.8	40.7	11.6	16.1	0.9449	-	-
	[2000 x 200] ≡ 0.4%	17.5	40.3	11.3	15.8	0.9455	-	-
	Full-data	**18.1**	**41.9**	**11.8**	**16.4**	**0.947**	-	-
PTB & Transformer	[10 x 50] ≡ 0.02%	-	-	-	-	-	238.5	20.48
	[200 x 50] ≡ 0.47%	-	-	-	-	-	124.0	24.0
	[400 x 50] ≡ 1%	-	-	-	-	-	91.9	25.16
	[2000 x 50] ≡ 4.7%	-	-	-	-	-	91.0	25.4
	Full-data	-	-	-	-	-	**72.10**	**26.03**

Table 4.6. Cross-architecture generalization for FARZI DATA of size [50×150] of the ML-100k dataset.

Teacher	Student HR@10 / HR@100	
	SASRec	GRU4Rec
SASRec	19.61/61.50	19.93/64.47
GRU4Rec	18.23/61.08	22.16/66.70
Full-Data	18.23/60.65	21.20/64.05

4.6.2 How versatile is FARZI DATA?

Since FARZI DATA is inherently optimized for a specific learning algorithm, we ascertain its universality by training different kinds of `student networks` over data synthesized using a given `teacher network` in FARZI's inner-loop for the sequential recommendation task. Note that the `student network` is completely unrelated to the data synthesis procedure and underlying FARZI optimization. From the results in Table 4.6, we observe that irrespective of the `teacher network`, FARZI DATA is able to train varied `student network` architectures (Transformers $\leftrightarrow$ RNNs) better than training on the full dataset. On the other hand, however, the best performance for any given `student network` is obtained when the same network is used during FARZI optimization.

4.6.3 Does training on FARZI DATA converge faster?

Unlike the existing DD literature, we assess the training time improvement for using FARZI DATA. Comparing the training time for SASRec on the entire Netflix dataset *vs.* its corresponding FARZI DATA in Table 4.7, we observe that SASRec can train equally well (cf. Figure 4.4) while seeing 70× less training sequences leading to 20× less training time. Even for this large dataset, the total time for FARZI optimization (500 outer-loop steps ×15s / step = 7500s) is only twice as large compared to a single training run on the full dataset (3700s). As is the case with all popular data quality scoring, data distillation, *etc.* family of methods;

Table 4.7. Training time for SASRec on the Netflix dataset. Wall-clock times are calculated using the same hardware and codebase.

Data	Size	Convergence Duration	Wall-clock Training Time
FARZI DATA	[2000×200]	100 Epochs (≡ 0.2M Sequences)	172s
Full Data	[476k×200]	30 Epochs (≡ 14M Sequences)	3700s

the cost of curating high-quality data summaries is always justified via *amortization*, *i.e.*, the one-time cost of data-curation is recovered by reaping the efficiency gains over multiple rounds of using the same curated data. Notably, the case for amortization is even stronger for FARZI where we observe strong cross-architecture generalization (Table 4.6), making once-optimized FARZI DATA to be re-usable not only for the same model used during the FARZI optimization but also by various other kinds of model architectures.

4.6.4 How important is the inner-loop optimizer in FARZI?

We compare SGD (with or without momentum) and Adam (Kingma and Ba, 2015) as different optimization routines in FARZI's inner-loop (Equation (4.2)). Notably, we implement differentiable Adam optimization in three different ways: (i) `higher` package (Grefenstette et al., 2019); (ii) PyTorch's autograd; and (iii) our efficient reverse-mode implementation (Algorithm 6). We measure their effect on downstream performance as well as the time and memory associated with each outer-loop iteration in Figure 4.5. We first observe that Adam is much better suited for DD in our setting. This is a novel finding in the context of meta-learning and its applications, where Adam has been reported to be worse than SGD (Grefenstette et al., 2019). Further, we observe that while different reverse-mode implementations of Adam lead to data of similar sample quality, their computational properties vastly differ. We observe that PyTorch and `higher` have similar memory footprints, but the former has a lower runtime. Our efficient

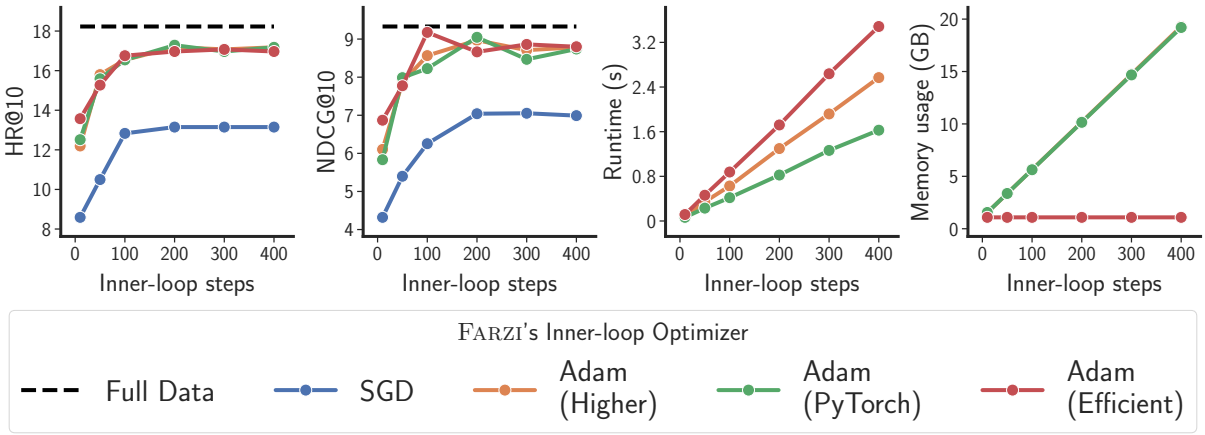

Figure 4.5. Changes in distillation performance and computational scalability of each outer-loop step for different inner-loop optimizers and increasing number of inner-loop steps. All results are for `[10×150]` sized FARZI DATA of the ML-100k dataset.

implementation elegantly trades-off memory with runtime, leading to constant memory footprint and a linear increase in runtime compared to PyTorch's autograd. This allows FARZI to scale to large autoregressive datasets without compromising on data fidelity.

4.6.5 How do different meta-objectives affect FARZI?

We further evaluate the importance of FARZI's optimization objective by comparing it with existing DD approaches. We adapt existing approaches to work with autoregressive data by reusing the latent distillation proposition of FARZI, and vary only the outer-loop *goodness function* to (i) gradient matching (DC (Zhao et al., 2021)); (ii) meta-matching (MM (Wang et al., 2018b; Deng and Russakovsky, 2022)); or (iii) trajectory matching (MTT (Cazenavette et al., 2022)), as explained below.

DC (Zhao et al., 2021) This data distillation objective performs one-step gradient matching using a distance function $\mathfrak{D} : \mathbb{R}^{|\Phi|} \times \mathbb{R}^{|\Phi|} \mapsto \mathbb{R}$:

$$\underset{\mathbf{M},\tilde{\mathcal{D}}_{\text{syn}}}{\arg\min} \; \mathop{\mathbb{E}}_{\theta_0 \sim \Theta} \left[\sum_{t=0}^{T} \mathfrak{D}\left(\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta_t), \nabla_\theta \mathcal{L}_{\mathcal{D}_{\text{syn}}}(\theta_t) \right) \right]$$

$$\text{s.t.} \quad \theta_{t+1} \leftarrow \text{Opt}\left(\theta_t, \nabla_\theta \mathcal{L}_{\mathcal{D}_{\text{syn}}}(\theta_t)\right) \quad ; \quad \mathcal{D}_{\text{syn}} \leftarrow \text{softmax}\left(\tilde{\mathcal{D}}_{\text{syn}} \cdot \mathbf{M} \;/\; \tau\right) .$$

Table 4.8. Comparison of FARZI with other existing DD techniques modified to distill autoregressive data. Results for both using SGD or Adam as the inner-loop optimizer are listed. Meta-matching is shortened as MM. The best distillation result for each metric is colored **orange**, and the best result other than FARZI is colored **blue** for Adam-based methods and **emboldened** for SGD-based methods.

Dataset	Metric	Random Sampling	Data Distillation Objectives							Full-Data
			DC		MM		MTT		FARZI	
			SGD	Adam	SGD	Adam	SGD	Adam		
ML-100k [50×150]	HR@10 ↑	7.74	7.95	11.77	9.65	**16.86**	**12.19**	14.52	**19.61**	18.23
	HR@100 ↑	39.13	41.88	49.52	42.31	**58.43**	**50.37**	56.94	**61.50**	60.65
	nDCG@10 ↑	3.83	3.44	5.6	4.72	**8.35**	**6.12**	6.73	**9.91**	9.33
	nDCG@100 ↑	9.61	9.84	12.51	10.85	**16.47**	**13.03**	14.66	**17.91**	17.69
PTB [400×50]	Perplexity ↓	218.66	203.23	131.07	**180.61**	**115.84**	202.98	129.72	**91.92**	72.10
	Accuracy ↑	20.42	20.64	22.35	**21.60**	**23.47**	21.00	23.00	**25.16**	26.03

MM (Wang et al., 2018b; Deng and Russakovsky, 2022) The meta-matching objective computes the meta-gradient by unrolling the inner-loop optimization starting from random networks:

$$\underset{\mathbf{M},\tilde{\mathcal{D}}_{\text{syn}}}{\arg\min} \; \mathop{\mathbb{E}}_{\theta_0\sim\Theta} [\mathcal{L}_{\mathcal{D}}(\theta_T)]$$

$$\text{s.t.} \quad \theta_{t+1} \leftarrow \text{Opt}\left(\theta_t, \nabla_\theta \mathcal{L}_{\mathcal{D}_{\text{syn}}}(\theta_t)\right) \quad ; \quad \mathcal{D}_{\text{syn}} \leftarrow \text{softmax}\left(\tilde{\mathcal{D}}_{\text{syn}} \cdot \mathbf{M} \;/\; \tau\right) .$$

MTT (Cazenavette et al., 2022) The trajectory matching objective computes the meta-gradient by matching the parameters of networks trained on the real data for M optimization steps *vs.* models trained on the data summary for $N \ll M$ steps. Let $\{\theta_t^{\mathcal{D}}\}_{t=0}^{T}$ represent the training

trajectory of training Φ_θ on $\mathcal{D}$, and $\mathfrak{D} : \mathbb{R}^{|\Phi|} \times \mathbb{R}^{|\Phi|} \mapsto \mathbb{R}$ be a pertinent distance function:

$$\underset{\mathbf{M}, \tilde{\mathcal{D}}_{\mathsf{syn}}}{\arg\min} \; \underset{\theta_0 \sim \Theta}{\mathbb{E}} \left[\sum_{t=0}^{T-M} \frac{\mathfrak{D}\left(\theta^{\mathcal{D}}_{t+M}, \theta^{\mathcal{D}_{\mathsf{syn}}}_{t+N}\right)}{\mathfrak{D}\left(\theta^{\mathcal{D}}_{t+M}, \theta^{\mathcal{D}}_{t}\right)} \right]$$

$$\text{s.t.} \quad \theta^{\mathcal{D}_{\mathsf{syn}}}_{t+i+1} \leftarrow \text{Opt}\left(\theta^{\mathcal{D}_{\mathsf{syn}}}_{t+i}, \nabla_\theta \mathcal{L}_{\mathcal{D}_{\mathsf{syn}}}(\theta^{\mathcal{D}_{\mathsf{syn}}}_{t+i})\right) \quad ; \quad \theta^{\mathcal{D}_{\mathsf{syn}}}_{t+1} \leftarrow \text{Opt}\left(\theta^{\mathcal{D}}_{t}, \nabla_\theta \mathcal{L}_{\mathcal{D}_{\mathsf{syn}}}(\theta^{\mathcal{D}}_{t})\right) \quad ;$$

$$\mathcal{D}_{\mathsf{syn}} \leftarrow \text{softmax}\left(\tilde{\mathcal{D}}_{\mathsf{syn}} \cdot \mathbf{M} \;/\; \tau\right) .$$

Even though all existing DD approaches use SGD in their inner-loop, we nonetheless experiment with both SGD and our efficient reverse-mode Adam (Algorithm 6), and list the results in Table 4.8. We observe that Adam is a consistently better inner-loop optimizer irrespective of the meta-objective used. This is in stark contrast with existing DD studies which use SGD in the inner-loop. Further, FARZI significantly outperforms all existing DD techniques despite improving them to use Adam in the inner-loop.

4.6.6 How important are pre-trained trajectories for data distillation?

To elicit the importance of the pre-trained trajectories, *i.e.*, $\Omega \triangleq \{[\theta_i]_{i=1}^{T} \mid \theta_0 \sim \Theta\}$ in FARZI's optimization (Equation (4.2)), we plot the change in downstream distillation performance with increasing $|\Omega|$ in Figure 4.6b. We indeed observe a massive improvement in downstream distillation performance with using as little as just 5 trajectories, compared to randomly initializing networks in FARZI's inner-loop. Notably, the improvement saturates as we keep adding more trajectories to Ω.

4.6.7 Does FARZI affect cold users or items more?

A longstanding problem in recommender systems is modeling the cold-start scenario, *i.e.*, users/items with less data. We study the effect training models on FARZI DATA from the cold-start perspective, by stratifying the users and items based on their popularity into equal-sized quantiles, and checking the trained model's performance on each individual quantile. In

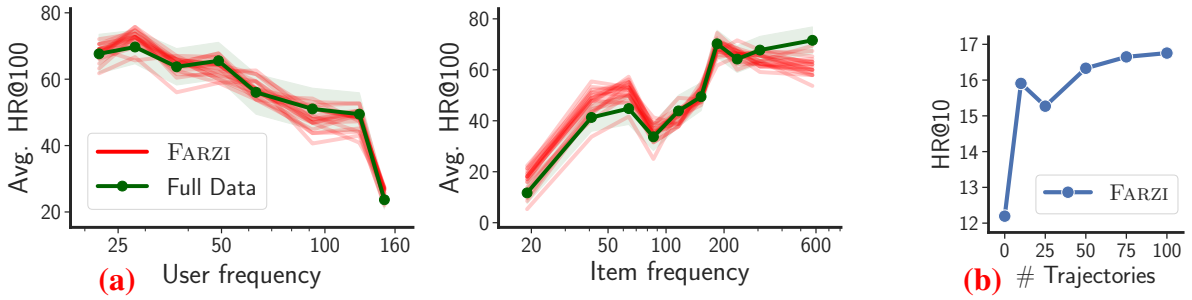

Figure 4.6. (a) Performance of SASRec trained on `[50×150]` sized FARZI DATA for ML-100k, and stratified over the popularity of users and items. The popularities are quantized into 10 equal sized bins and the average HR@100 is plotted. (b) Performance change of SASRec trained on `[10×150]` sized FARZI DATA for ML-100k with increasing number of pretrained trajectories.

Figure 4.6a, we do this for SASRec (Kang and McAuley, 2018) trained on (i) the full dataset; and (ii) FARZI DATA synthesized using different hyper-parameter combinations. We first observe that less popular items are harder to model, as is the typical case of recommender systems. Further, we observe that models trained on FARZI DATA are, in expectation, (i) better on the tail/torso region of users/items; but (ii) worse for the head users/items. Notably, this behaviour is not directly optimized-for by FARZI, and is a by-product of the overall data-quality optimization in Equation (4.2).

4.7 Conclusion & Discussion

In this chapter, we proposed FARZI—a scalable technique to summarize large autoregressive datasets into terse, high-fidelity data summaries. Through extensive experiments on next-item recommendation and language modeling, we demonstrated that data synthesized by FARZI (FARZI DATA) is able to train various kinds of SoTA models to the same quality (if not better) as training them on the full dataset, despite FARZI DATA being up to 1000× smaller.

Better performance with less FARZI DATA. Figure 4.4 hints towards the possibility for improving performance, while training on orders of magnitude smaller FARZI DATA *vs.* the original dataset, that also leads to training efficiency gains (Table 4.7). We hypothesize this to be an artifact of the *implicit denoising* in FARZI due to the data bottleneck, and are excited to

explore this direction further in future work.

FARZI for Large Language Models (LLMs). While training LLMs is out-of-scope for this chapter given the compute available to us, we see a clear path for enabling DD to scale to LLM-scale data via FARZI. First, our largest dataset (Netflix, 0.5M sequences) uses a tiny batch size of just 512 and 25 for the real and synthetic data respectively (Table 4.2). Further, FARZI uses efficient reverse-mode Adam for meta-gradient computation (Algorithm 6). Both of these allow DD to scale out-of-the-box to LLM-scale models and data using strictly no more hardware requirements *vs.* training the same LLM (cf. Figure 4.3 for FARZI's computational complexity).

Chapter 4, in part, is currently being prepared for submission for publication of the material "Farzi Data: Autoregressive Data Distillation." by Noveen Sachdeva, Zexue He, Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Derek Zhiyuan Cheng, and Julian McAuley. 2024. The dissertation author was the primary investigator and author of this paper.

Bibliography

Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

Rishabh Agarwal, Nino Vieillard, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. Gkd: Generalized knowledge distillation for auto-regressive sequence models. *arXiv preprint arXiv:2306.13649*, 2023.

Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard G Baraniuk. Self-consuming generative models go mad. *arXiv preprint arXiv:2307.01850*, 2023.

Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Dietmar Jannach, and Claudio Pomo. Top-n recommendation algorithms: A quest for the state-of-the-art. *arXiv preprint arXiv:2203.01155*, 2022.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, and Zhifeng Chen et al. Palm 2 technical report, 2023.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019.

Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkl8sJBYvH.

Pranjal Awasthi, Natalie Frank, and Mehryar Mohri. On the rademacher complexity of linear hypothesis sets. *arXiv preprint arXiv:2007.11045*, 2020.

Fadhel Ayed and Soufiane Hayou. Data pruning and neural scaling laws: fundamental limitations of score-based algorithms. *arXiv preprint arXiv:2302.06960*, 2023a.

Fadhel Ayed and Soufiane Hayou. Data pruning and neural scaling laws: fundamental limitations of score-based algorithms. *Transactions on Machine Learning Research*, 2023b. ISSN 2835-8856. URL https://openreview.net/forum?id=iRTL4pDavo.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Samyadeep Basu, Phil Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*, 2021.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning practice and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118*, 2018.

Francois Belletti, Karthik Lakshmanan, Walid Krichene, Nicolas Mayoraz, Yi-Fan Chen, John Anderson, Taylor Robie, Tayo Oguntebi, Dan Shirron, and Amit Bleiwess. Scaling up collaborative filtering data sets through randomized fractal expansions, 2019.

James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, 2007.

Jeff Bilmes. Submodularity in machine learning and artificial intelligence. *arXiv preprint arXiv:2202.00132*, 2022.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in Neural Information Processing Systems*, 33: 14879–14890, 2020a.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020b.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Martin Briesch, Dominik Sobania, and Franz Rothlauf. Large language models suffer from their own output: An analysis of the self-consuming training loop. *arXiv preprint arXiv:2311.16822*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

R. Cañamares and P. Castells. On target item sampling in offline recommender system evaluation. In *14th ACM Conference on Recommender Systems*, 2020.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4750–4759, 2022.

Dong-Kyu Chae, Jihoo Kim, Duen Horng Chau, and Sang-Wook Kim. Ar-cf: Augmenting virtual users and items in collaborative filtering for addressing cold-start problems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 1251–1260, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. URL https://doi.org/10.1145/3397271.3401038.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *CoRR*, abs/2010.03240, 2020. URL https://arxiv.org/abs/2010.03240.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *KDD '16*, 2016.

Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. On sampling strategies for neural network-based collaborative filtering. In *Proceedings of the 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, volume 33 of *KDD '17*, pages 1094–1105, 2017.

Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. *arXiv preprint arXiv:1203.3472*, 2012.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Kashyap Chitta, José M Álvarez, Elmar Haussmann, and Clément Farabet. Training data subset search with ensemble active learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14741–14752, 2021.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Benjamin Coleman and Anshumali Shrivastava. Sub-linear race sketches for approximate kernel density estimation on streaming data. In *Proceedings of The Web Conference 2020*, pages 1739–1749, 2020.

Benjamin Coleman, Benito Geordie, Li Chou, RA Leo Elworth, Todd Treangen, and Anshumali Shrivastava. One-pass diversified sampling with application to terabyte-scale genomic sequence streams. In *International Conference on Machine Learning*, pages 4202–4218. PMLR, 2022.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *ICLR*, 2020.

Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6, 1987.

M. Dacrema, P. Cremonesi, and D. Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, 2019.

Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. *Advances in neural information processing systems*, 29, 2016.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 2009.

Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks. In *Advances in Neural Information Processing Systems*, 2022.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Luc Devroye. The equivalence of weak, strong and complete convergence in l1 for kernel density estimates. *The Annals of Statistics*, pages 896–904, 1983.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels, 2024.

Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM Journal on Computing*, 49(3): 601–657, 2020.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33: 2881–2891, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Team Gemini, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*, pages 181–195. Springer, 2022a.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*, pages 181–195. Springer, 2022b.

Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. Wiki-40b: Multilingual language model dataset. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2440–2452, 2020.

U. Gupta, Y. Kim, S. Lee, J. Tse, H. S. Lee, G. Wei, D. Brooks, and C. Wu. Chasing carbon: The elusive environmental footprint of computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, mar 2021.

Kelvin Guu, Albert Webson, Ellie Pavlick, Lucas Dixon, Ian Tenney, and Tolga Bolukbasi. Simfluence: Modeling the influence of individual training examples by simulating training runs. *arXiv preprint arXiv:2303.08114*, 2023.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 2015.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, 2017.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35:30016–30030, 2022.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 100–108, 2014.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

H. Jain, Y. Prabhu, and M. Varma. Extreme multi-label loss functions for recommendation, tagging, ranking and other missing label applications. In *Proceedings of the SIGKDD Conference on Knowledge Discovery and Data Mining*, 2016.

H. Jain, V. Balasubramanian, B. Chunduri, and M. Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the 12th ACM Conference on Web Search and Data Mining*, 2019.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: The surprising power of small language models, 2023.

Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019.

Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 720–730, 2022a.

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=WLEx3Jo4QaB.

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022c.

Sham M Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. *Advances in neural information processing systems*, 21, 2008.

Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *Advances in Neural Information Processing Systems*, 33:21798–21809, 2020.

W. Kang and J. McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining*, 2018.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Zohar Karnin and Edo Liberty. Discrepancy, coresets, and sketches in machine learning. In *Conference on Learning Theory*, pages 1975–1993. PMLR, 2019.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

V. Kaushal, R. Iyer, S. Kothawade, R. Mahadev, K. Doctor, and G. Ramakrishnan. Learning from less data: A unified data subset selection and active learning framework for computer vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

Ehsan Kazemi, Shervin Minaee, Moran Feldman, and Amin Karbasi. Regularized submodular maximization at scale. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5356–5366. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/kazemi21a.html.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*, 2020.

Krishnateja Killamsetty, Durga S, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5464–5474. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/killamsetty21a.html.

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

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017a. URL https://openreview.net/forum?id=SJU4ayYgl.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017b.

Adam Kirsch and Michael Mitzenmacher. Distance-sensitive bloom filters. In *2006 Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 41–50. SIAM, 2006.

Taeyong Kong, Taeri Kim, Jinsung Jeon, Jeongwhan Choi, Yeon-Chang Lee, Noseong Park, and Sang-Wook Kim. Linear, or non-linear, that is the question! In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 517–525, 2022.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

Andreas Krause, Marco Tagliasacchi, and Zalán Borsos. Semi-supervised batch active learning via bilevel optimization. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021.

Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, 2020.

A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

I Elizabeth Kumar, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler. Problems with shapley-value-based explanations as feature importance measures. In *International Conference on Machine Learning*, pages 5491–5500. PMLR, 2020.

Yongchan Kwon and James Zou. Data-oob: Out-of-bag estimate as a simple and efficient data value. In *International conference on machine learning*. PMLR, 2023.

Yongchan Kwon, Manuel A Rivas, and James Zou. Efficient computation and analysis of distributional shapley values. In *International Conference on Artificial Intelligence and Statistics*, pages 793–801. PMLR, 2021.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Alycia Lee, Brando Miranda, and Sanmi Koyejo. Beyond scale: the diversity coefficient as a data quality metric demonstrates llms are pre-trained on formally diverse data. *arXiv preprint arXiv:2306.13840*, 2023.

Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak,

and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, 2022.

J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM SIGKDD Conference on Knowledge Discovery in Data Mining*, KDD '05, 2005.

Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *KDD '06*, 2006.

Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, 2018.

Zichang Liu, Zhaozhuo Xu, Benjamin Coleman, and Anshumali Shrivastava. One-pass distribution sketch for measuring data heterogeneity in federated learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023a.

Shayne Longpre, Gregory Yauney, Emily Reif, Katherine Lee, Adam Roberts, Barret Zoph, Denny Zhou, Jason Wei, Kevin Robinson, David Mimno, et al. A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, & toxicity. *arXiv preprint arXiv:2305.13169*, 2023b.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of Machine Learning Research*, 24(253):1–15, 2023.

Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. Learning disentangled representations for recommendation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/a2186aa7c086b46ad4e8bf81e2a3a19b-Paper.pdf.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015a.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015b.

Tung Mai, Cameron Musco, and Anup Rao. Coresets for classification–simplified and strengthened. *Advances in Neural Information Processing Systems*, 34:11643–11654, 2021.

Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling, 2024.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*, 2023.

P. Mattson, C. Cheng, G. Diamos, C. Coleman, P. Micikevicius, D. Patterson, H. Tang, G. Wei, P. Bailis, V. Bittorf, D. Brooks, D. Chen, D. Dutta, U. Gupta, K. Hazelwood, A. Hock, X. Huang, D. Kang, D. Kanter, N. Kumar, J. Liao, D. Narayanan, T. Oguntebi, G. Pekhimenko, L. Pentecost, Vijay Janapa, R., T. Robie, T. St John, C. Wu, L. Xu, C. Young, and M. Zaharia. Mlperf training benchmark. In *Proceedings of Machine Learning and Systems*, 2020.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *ICDM '12*, 2012.

Kristof Meding, Luca M Schulze Buschoff, Robert Geirhos, and Felix A Wichmann. Trivial or impossible–dichotomous data difficulty masks model differences (on imagenet and beyond). *arXiv preprint arXiv:2110.05922*, 2021.

Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Exploring data splitting strategies for the evaluation of recommendation models. In *Fourteenth ACM Conference on Recommender Systems*, RecSys '20, 2020.

Fei Mi, Xiaoyu Lin, and Boi Faltings. Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *Fourteenth ACM Conference on Recommender Systems*, page 408–413, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375832. URL https://doi.org/10.1145/3383313.3412218.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and

developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*, 2021.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pages 15630–15649. PMLR, 2022.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Shira Mitchell, Eric Potash, Solon Barocas, Alexander D'Amour, and Kristian Lum. Prediction-based decisions and fairness: A catalogue of choices, assumptions, and definitions. *arXiv preprint arXiv:1811.07867*, 2018.

A. Mittal, N. Sachdeva, S. Agrawal, S. Agarwal, P. Kar, and M. Varma. Eclare: Extreme classification with label graph correlations. In *Proceedings of The ACM International World Wide Web Conference*, 2021.

Fred Morstatter, Jürgen Pfeffer, Huan Liu, and Kathleen Carley. Is the sample good enough? comparing data from twitter's streaming api with twitter's firehose. *Proceedings of the International AAAI Conference on Web and Social Media*, 7(1), Jun. 2013. URL https://ojs.aaai.org/index.php/ICWSM/article/view/14401.

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

Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE, 2008.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems. *CoRR*, abs/1906.00091, 2019.

Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=l-PrrQrK0QR.

Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*,

34, 2021b.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197, 2019a.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019b.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1g30j0qF7.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL https://github.com/google/neural-tangents.

S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *NeurIPS*, 2016.

OpenAI. Gpt-4 technical report, 2023.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Christopher R. Palmer, Phillip B. Gibbons, and Christos Faloutsos. Anf: A fast and scalable tool for data mining in massive graphs. In *Proceedings of the 8th SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '02, 2002.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance

deep learning library. In *NeurIPS*. 2019.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training, 2021.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607, 2021.

Jeff M Phillips. Coresets and sketches. In *Handbook of discrete and computational geometry*, pages 1269–1288. Chapman and Hall/CRC, 2017.

Mary Phuong and Marcus Hutter. Formal algorithms for transformers. *arXiv preprint arXiv:2207.09238*, 2022.

Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the International World Wide Web Conference*, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *CoRR*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

Adityanarayanan Radhakrishnan. Lecture 5: Ntk origin and derivation. 2022.

Adityanarayanan Radhakrishnan, George Stefanakis, Mikhail Belkin, and Caroline Uhler. Simple, fast, and flexible framework for matrix completion with infinite width neural networks. *Proceedings of the National Academy of Sciences*, 119(16):e2115064119, 2022.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena,

Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, 2009.

Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. Neural collaborative filtering vs. matrix factorization revisited. In *Fourteenth ACM conference on recommender systems*, pages 240–248, 2020.

Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=CR1XOQ0UTh-.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The annals of mathematical statistics*, pages 832–837, 1956.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Durga S, Rishabh Iyer, Ganesh Ramakrishnan, and Abir De. Training data subset selection for regression with controlled generalization error. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9202–9212. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/s21a.html.

Noveen Sachdeva and Julian McAuley. How useful are reviews for recommendation? a critical review and potential improvements. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, 2020.

Noveen Sachdeva and Julian McAuley. Data distillation: A survey. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. Survey Certification.

Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, WSDM '19, 2019.

Noveen Sachdeva, Yi Su, and Thorsten Joachims. Off-policy bandits with deficient support. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, 2020.

Noveen Sachdeva, Carole-Jean Wu, and Julian McAuley. Svp-cf: Selection via proxy for collaborative filtering data. *arXiv preprint arXiv:2107.04984*, 2021.

Noveen Sachdeva, Mehak Preet Dhaliwal, Carole-Jean Wu, and Julian McAuley. Infinite recommendation networks: A data-centric approach. In *Advances in Neural Information Processing Systems*, 2022a.

Noveen Sachdeva, Ziran Wang, Kyungtae Han, Rohit Gupta, and Julian McAuley. Gapformer: Fast autoregressive transformers meet rnns for personalized adaptive cruise control. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2528–2535. IEEE, 2022b.

Noveen Sachdeva, Carole-Jean Wu, and Julian McAuley. On sampling collaborative filtering datasets. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, page 842–850, New York, NY, USA, 2022c. Association for Computing Machinery. ISBN 9781450391320. doi: 10.1145/3488560.3498439. URL https://doi.org/10.1145/3488560.3498439.

Noveen Sachdeva, Zexue He, Wang-Cheng Kang, Jianmo Ni, Derek Zhiyuan Cheng, and Julian McAuley. Farzi data: Autoregressive data distillation. *arXiv preprint arXiv:2310.09983*, 2023.

T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, and T. Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Generalized outlier detection with flexible kernel density estimates. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 542–550. SIAM, 2014.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai, 2019.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018.

Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, et al. Mixture-of-experts meets instruction tuning: A winning combination for large language models. *arXiv preprint arXiv:2305.14705*, 2023.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget.(5 2023). *URl: https://arxiv. org/abs/2305.17493*, 2023.

Paris Siminelakis, Kexin Rong, Peter Bailis, Moses Charikar, and Philip Levis. Rehashing kernel evaluation in high dimensions. In *International Conference on Machine Learning*, pages 5789–5798. PMLR, 2019.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Harald Steck. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*, pages 3251–3257, 2019.

Eliza Strickland. Andrew ng: Farewell, big data. IEEE Spectrum, Mar 2022. URL https://spectrum.ieee.org/andrew-ng-data-centric-ai.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, July 2019a.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019b.

Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2020a.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer.

In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020b.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari S Morcos. D4: Improving llm pretraining via document de-duplication and diversification. *arXiv preprint arXiv:2308.12284*, 2023.

M. Toneva, A. Sordoni, R. Combes, A. Trischler, Y. Bengio, and G. Gordon. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Murad Tukan, Cenk Baykal, Dan Feldman, and Daniela Rus. On coresets for support vector machines. *Theoretical Computer Science*, 890:171–191, 2021.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

M. Wan and J. McAuley. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018a.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Jiachen T Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 6388–6421. PMLR, 2023.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Lilian Weng. Large transformer model inference optimization. *Lil'Log*, Jan 2023. URL https://lilianweng.github.io/posts/2023-01-10-inference-optimization/.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359*, 2019.

Dominik Wied and Rafael Weißbach. Consistency of the kernel density estimator: a survey. *Statistical Papers*, 53(1):1–21, 2012.

Carole-Jean Wu, Robin Burke, Ed H. Chi, Joseph Konstan, Julian McAuley, Yves Raimond, and Hao Zhang. Developing a recommendation benchmark for mlperf training and inference. *CoRR*, abs/2003.07336, 2020.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4: 795–813, 2022.

Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth ACM international conference on web search and data mining*, pages 153–162, 2016.

Amatriain X, Jaimes A, Oliver N, and Pujol J.M. Data mining methods for recommender systems. In *Recommender Systems Handbook*. Springer, 2011.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *arXiv preprint arXiv:2305.10429*, 2023a.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. *arXiv preprint arXiv:2302.03169*, 2023b.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36, 2024.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Rethinking neural vs. matrix-factorization collaborative filtering: the theoretical perspectives. In *International Conference on Machine Learning*, pages 11514–11524. PMLR, 2021.

R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD '18*, 2018. ISBN 9781450355520.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=mSAKhLYLSsl.

Wenqing Zheng, Edward W Huang, Nikhil Rao, Sumeet Katariya, Zhangyang Wang, and Karthik Subbian. Cold brew: Distilling graph node representations with incomplete or missing neighborhoods. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1ugNpm7W6E.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*, 2023.

Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. In *Advances in Neural Information Processing Systems*, 2022.

Feng Zhu, Chaochao Chen, Yan Wang, Guanfeng Liu, and Xiaolin Zheng. Dtcdr: A framework for dual-target cross-domain recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1533–1542, 2019.

Ziwei Zhu, Yun He, Yin Zhang, and James Caverlee. Unbiased implicit recommendation and propensity estimation via combinational joint learning. In *Fourteenth ACM Conference on Recommender Systems*, RecSys '20, 2020.