공학석사 학위논문

Energy-based Contrastive Learning and Ensembling Models for Sequential Recommendation

순차 추천을 위한 에너지 기반 대조학습과 앙상블 모델

2023년 8월

서울대학교 대학원

협동과정 인공지능전공

류 정 현

Energy-based Contrastive Learning and Ensembling Models for Sequential Recommendation

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering
to the faculty of the Graduate School of
Seoul National University

by

Jung Hyun Ryu

Dissertation Director : Professor Myungjoo Kang

Interdisciplinary Program in Artificial Intelligence
Seoul National University

August 2023

Energy-based Contrastive Learning and Ensembling Models for Sequential Recommendation

순차 추천을 위한 에너지 기반 대조학습과 앙상블 모델

지도교수 강 명 주

이 논문을 공학석사 학위논문으로 제출함

2023년 6월

서울대학교 대학원

협동과정 인공지능

류 정 현

류 정 현의 공학석사 학위논문을 인준함

2023년 6월

위 원 장 국 웅 (인)

부 위 원 장 강 명 주 (인)

위 원 하 승 열 (인)

Abstract

Energy-based Contrastive Learning and Ensembling Models for Sequential Recommendation

Jung Hyun Ryu

Interdisciplinary Program in Artificial Intelligence

The Graduate School

Seoul National University

With the exponential growth of online platforms and services, recommendation systems have become essential for identifying relevant items based on user preferences. In the domain of sequential recommendation, which aims to capture evolving user preferences over time. To address this, contrastive learning methods have been proposed to target data sparsity, a challenge in recommendation systems due to the limited user-item interactions. However, they do have limitations such as the occurrence of incorrectly labeling similar instances as dissimilar, leading to missed opportunities for meaningful connections and embeddings.

In this thesis, we present two main contributions in the field of sequential learning. Firstly, we propose an advanced approach to contrastive learning specifically designed for sequential recommendation systems. By adaptively constructing negative samples, we improve the quality of item embeddings, leading to enhanced performance in recommendation tasks. Secondly, we introduce a novel ensemble

technique for sequential models by applying Fisher merging. This approach ensures robust fine-tuning by merging the parameters of multiple models, resulting in improved overall performance. Through extensive experiments, we demonstrate the effectiveness of our proposed methods, highlighting their potential to advance the state-of-the-art in sequential learning and recommendation systems.

Key words: contrastive learning, ensemble, sequential recommendation

Student Number: 2021-26380

Contents

Chapter 1

Introduction

In recent years, the rapid growth of online platforms and services has led to the accumulation of vast amounts of data daily. Among the various methods of harnessing this data, recommendation systems play a prominent role. Recommendation systems are employed to identify relevant items based on user preferences and interests. To capture the evolving user preferences over time, the field of sequential recommendation has emerged. Prominent examples in this domain include SASRec and BERT4Rec[28]. In this paper, we define the problem of sequential recommendation as follows:

Let $\mathcal{U}$ be the set of users $\mathcal{U} = \{u_1, u_2, \cdots, u_{|\mathcal{U}|}\}$, and $\mathcal{V}$ be the set of items as $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$. The sequence of user-item interaction for u_i is a list with chronological order, $S_i = [v_1^{u_i}, v_2^{u_i}, \cdots, v_t^{u_i}, \cdots, v_{n_{u_i}}^{u_i}]$. Here user $u_i \in \mathcal{U}$, $v_t^{u_i} \in \mathcal{V}$, and user u_i interact item $v_t^{u_i}$ in time step t. The length of sequence for user u_i is n_{u_i}, and our object is to build a model predicting the item with which user is

interact in the next time step, i.e,

$$p(v^{u_i}_{n_{u_i}+1} = v|S_i) \tag{1.1}$$

However, the field of recommendation systems inherently faces the challenge of data sparsity. This problem arises when the majority of elements in the user-item matrix are empty since users have not interacted with most items. Consequently, this poses limitations on embedding the items during the modeling process. To address this issue, previous works such as CL4SRec[32], DuoRec[25], and ECL-SL have proposed methods that incorporate contrastive learning. In this paper, we introduce an advanced approach to contrastive learning in sequential recommendation.

The aforementioned methodologies typically employ similar model structures but utilize various learning frameworks. Recognizing this, we propose a method to ensemble the parameters of models trained with different contrastive learning techniques in a sequential recommendation. Prior research has shown that ensemble methods yield significant benefits when multiple learning frameworks are employed.

Furthermore, by assuming the posterior distribution of parameters θ_i for each model model_i, we achieved more effective ensemble results. This approach allowed us to capture the uncertainty associated with each model's parameter estimates and leverage this information to enhance the ensemble process. By considering the posterior distributions, we were able to account for the variability in parameter values across different models and obtain a more robust and comprehensive ensemble outcome.

Chapter 2

Related Works

2.1 Contrastive Learning

Contrastive learning is a type of self-supervised learning technique, where the goal is to learn useful representations of data without any explicit labeling or supervision[20]. In contrastive learning, the algorithm is trained to compare and contrast different samples of data to learn the underlying patterns and relationships between them. The basic idea is to take two different data points and generate two different representations for each data point[24]. The algorithm aims to learn to differentiate between representations of the same data point, referred to as a positive pair, and representations of different data points, known as a negative pair, using a similarity metric. The training process involves optimizing the model to maximize the similarity between the representations of positive pairs while minimizing the similarity between the representations of negative pairs. This way, the model learns to capture the meaningful differences and similarities between the data points, which can then be used for downstream tasks such as classification,

clustering, or retrieval.

Contrastive learning can be mathematically formulated using the InfoNCE (Normalized Mutual Information Neural Estimation)[31] loss function, which is used to maximize the similarity between positive pairs and minimize the similarity between negative pairs. The InfoNCE loss is defined as:

$$\ell_{\mathsf{NCE}} \triangleq \mathop{\mathbb{E}}_{\substack{(x,x^+)\sim p_{\mathsf{pos}} \\ \{x_i^-\}_{i=1}^M \overset{i.i.d.}{\sim} p_{\text{data}}}} \left[-\log \frac{e^{f(x)^\top f(x^+)/\tau}}{e^{f(x)^\top f(x^+)/\tau} + \sum_i e^{f(x_i^-)^\top f(x)/\tau}} \right]. \tag{2.1}$$

2.1.1 Contrastive Learning in Sentence Embedding

The paper introduces SimCSE[10] a contrastive sentence embedding framework, which utilizes pre-trained embeddings to enhance the quality of sentence representations. It employs contrastive learning principles to bring similar samples closer and push dissimilar samples apart, resulting in improved uniformity and alignment within the embeddings. The paper presents two variations: unsupervised SimCSE, which introduces dropout-based noise[27] and batch-level negative sampling, and supervised SimCSE[10], which utilizes NLI(Natural Language Inference) datasets with entailment and contradiction pairs[5, 26]. Therefore, for each premise and its entailment hypothesis, there is an accompanying contradiction hypothesis7

Both variations demonstrate the effectiveness of contrastive objectives in refining sentence embeddings. SimCSE[10] offers a promising approach to enhance the quality of sentence representations, making it applicable to various natural language processing tasks.

DiffCSE[3] is an extension of SimCSE[10] that addresses limitations in the

unsupervised learning domain by considering additional augmentation methods and promoting equivariant contrastive learning[6] for sentence embeddings. While SimCSE[10] focuses on dropout insensitivity, DiffCSE[3] argues for sensitivity to augmentation techniques like masked language models. Leveraging theoretical foundations from computer vision, DiffCSE[3] emphasizes the superiority of equivariant contrastive learning, surpassing the limited capabilities of invariance-based approaches. It incorporates a difference prediction objective using a conditional ELECTRA[4] architecture to capture distinctions between original and modified sentences. By broadening augmentation methods and embracing equivariant contrastive learning, DiffCSE[3] enhances the robustness of sentence embeddings in unsupervised learning, providing valuable insights for more sophisticated representation learning frameworks in natural language processing.

PromCSE[17] incorporates an Energy-based Hinge loss, leveraging hard negatives for improved discrimination in supervised learning. The inclusion of this loss enhances the model's ability to distinguish between similar and dissimilar sentences, surpassing softmax-based losses. PromCSE's[17] integration of frozen PLMs, Soft Prompts, and the Energy-based Hinge loss provides a compelling approach to address overfitting and domain shift, advancing deep learning in natural language processing tasks.

2.1.2 Contrastive Learning in Sequential Recommendation

Contrastive Learning for Sequential Recommendation[32] is proposed as a solution to address the significant issue of data sparsity in recommendation systems by in-

corporating contrastive learning techniques. CL4SRec[32] focuses on constructing pairs with different viewpoints, where positive pairs carry semantic information. The paper introduces three data augmentation techniques at the sequence level to obtain more effective user representations. These augmentation methods include item cropping, which involves cropping a certain interval within the sequence, item masking, which masks several items within the sequence, and item reordering, which rearranges the order of items within a specific interval in the sequence. By applying these data augmentation techniques, CL4SRec[32] performs contrastive learning by creating positive pairs from the original sequence. These augmentation methods closely resemble the data augmentation approaches commonly used in the field of computer vision.[2] This integration of contrastive learning and sequence-based data augmentation in CL4SRec[32] contributes to advancements in addressing data sparsity and enhancing sequential recommendation systems.

DuoRec[25] model combines two forms of contrastive loss. Firstly, it utilizes unsupervised augmentation through dropout-based model-level augmentation to create positive pairs. It emphasizes the importance of pairing samples under the assumption that their meaning remains unchanged. The paper also highlights that semantic preservation alone in CL4SRec[32] may not be sufficient. In this regard, the proposed DuoRec[25] model heavily incorporates the feature-level augmentation through dropout, as suggested by SimCSE[10]. Additionally, DuoRec[25] employs supervised positive sampling, where pairs are created by considering sequences with the same target item as positive samples. This approach enhances the model's ability to capture the relationships between sequences with shared target items. By combining both unsupervised and supervised contrastive learning, DuoRec[25]

aims to improve the effectiveness of recommendation models in capturing semantic information and considering item-level relationships.

2.1.3 Alignment and Uniformity

In order to point out how well features are represented, previous work has suggested two key properties; alignment and uniformity[29]. Alignment measures how well paired positive instances are closely represented, typically computed as the average distance between them. On the other hand, uniformity assesses how well the feature distribution is uniformly spread, often quantified using a Gaussian kernel on a hypersphere. RGiven the data distribution p_{data} and a positive pair distribution p_{pos}, alignment and uniformity is defined as :

$$\ell_{\text{align}} \triangleq \mathbb{E}_{(x,x^+)\sim p_{\text{pos}}} \|f(x) - f(x^+)\|^2, \tag{2.2}$$

$$\ell_{\text{uniform}} \triangleq \mathbb{E}_{(x,y)\sim p_{\text{data}}} e^{-2\|f(x)-f(y)\|^2}. \tag{2.3}$$

Minimizing both metrics indicates an improved performance. Minimizing ℓ_{align} in equation 2.2 encourages the learned representations of samples x and x^+, sampled from p_{pos}, to be closer together. Minimizing ℓ_{uniform} in euation 2.3 encourages the samples from p_{data} to be uniformly represented. By considering these two objectives, the goal is to enhance the proximity of positive pairs while promoting uniform representation across the data distribution.

2.2 Model Ensemble

Researchers have extensively explored various methods for model ensemble, which involve combining the predictions or parameters of multiple models to attain enhanced performance compared to individual models[9]. One such example is bootstrapping, wherein models trained on different subsets of data are ensembled[1]. Boosting, another prominent ensemble method, leverages the collective knowledge of multiple weak models to create a stronger ensemble by iteratively adjusting the weights of misclassified instances[23]. Another notable ensemble technique is parameter merging, which aims to reduce the model's size by merging the parameters of multiple models into a single, more compact model[15]. This consolidation of parameters allows for improved efficiency in terms of storage and computational resources. However, it is important to note that many model ensemble approaches often necessitate additional training, which can be a drawback due to the increased computational requirements and time constraints involved.

2.2.1 Diverse Learning Framework

Models trained using diverse training methodologies exhibit different generalization capabilities, ultimately leading to uncorrelated errors. Research has demonstrated the ensemble effect across various training methodologies at the initialization, hyperparameter, architecture, framework, and dataset levels. These findings suggest that models tend to specialize in subdomains within the data, highlighting the crucial role of ensemble techniques in enhancing overall performance.

2.2.2 Merging Methods

Averaging Parameter Model Soup[30] presents an effective approach for combining parameters without additional training. It demonstrates research findings that improve the performance of trained models by constructing a "recipe" composed of diverse models and averaging their parameters. The study introduces three methods for creating the recipe: the uniform soup, which averages the parameter values of all models; the greedy soup, which sequentially adds models based on their performance ranking; and the learned soup, which identifies the optimal model interpolation through training. These approaches contribute to enhancing the overall performance of the model without the need for additional training.

Fisher Merging Within the scope of related works, parameter merging is interpreted as a process that maximizes the joint likelihood of model parameters' posteriors[22]. Previous studies consider averaging as a scenario where the posteriors of these models are assumed to follow an isotropic Gaussian distribution, and the joint likelihood is maximized accordingly. To refine this approach, efforts have been made to approximate the posterior of the model using Laplace Approximation. In this case, the distribution of each model is modeled by assuming the mean as the observed, which can be interpreted as trained parameter and the variance as the Gaussian distribution's Fisher matrix. By employing this formulation, the joint likelihood is calculated.

Chapter 3

Methodology

3.1 EHRec

We introduce Energy-based Contrastive Learning for Sequentail Recommendation Model, named EHRec.

3.1.1 Constructing Negative Samples

In previous works addressing item embedding in sequential recommendation problems through contrastive learning, the focus was mainly on augmenting positive samples. However, defining dissimilar sequences posed challenges as the recommendation domain exhibits high sparsity and significant variation in item popularity. It is difficult to establish meaningful distinctions between sequences. To tackle this issue, we approached the problem from the perspective of analyzing methods for constructing negative samples, which is the main idea of EHRec (Energy-based Contrastive Learning for Sequential Recommendation Model). Specifically, we defined the relationship between item sequences based on similarity and incorporated

this relationship into the contrastive loss.

The contrastive learning and batch-based positive and negative pair selection methods employed in SimCLR were both simple and intuitive. However, they do have limitations, such as the occurrence of false negative problems. Therefore, constructing well-designed negative pairs holds great importance in contrastive learning. We assume that the model learns to some extent the ability to distinguish false negative pairs from genuine negative pairs. If, during training, the model fails to differentiate between genuine negative pairs and false negative pairs, it can introduce bias into the overall learning process. Careful analysis and mitigation of such biases are crucial to ensure fair and accurate training outcomes.

Negative samples were constructed based on the similarity with other samples within the batch. In a scenario where the batch size is N, we first formed the positive pairs as suggested in DuoRec[25]. The negative samples were then selected from the remaining $2N - 2$ samples, considering only those with similarity below a certain threshold.

We refer to this threshold as ***SimThres***, and it allows for the generation of diverse negative samples based on the chosen *SimThres* value. To determine *SimThres*, we proposed two methods. These approaches enable us to define negative samples by considering the similarity threshold, allowing for better exploration of the dissimilarity between item sequences in the context of contrastive learning.

3.1.2 Determining the *SimThres*

For user $u_i \in \mathcal{U}$ and item $v_j \in \mathcal{V}$, the sequence of user-item interaction for u_i, is denoted as $s_i = [v_1^{u_i}, v_2^{u_i}, \cdots, v_t^{u_i}, \cdots, v_{n_{u_i}}^{u_i}]$. Let $\mathcal{S} = \{s_i, i = 1, \cdots, |\mathcal{U}|\}$ represent the set of user-item interaction. Also the model f be parameterized by θ and s_i^+ is

the augmented positive pair of s_i by model f.

For each s_i, we consider an ordered set, $\{s_{i,j}\}$, where j is the order index, sorted based on the similarity value with respect to s_i. For instance,

$$s_{i,1} = \underset{(s_j \in \mathcal{S}) \wedge (s_j \neq s_i, s_i^+)}{\arg\min} \text{sim}(f(s_i;\theta), f(s_j;\theta)). \tag{3.1}$$

We aim to construct negative samples for u_i using values from this set that do not exceed a certain threshold. In other words, the negative samples are given by

$$\{s_{i,j} \mid \text{sim}(f(s_i;\theta), f(s_{i,j};\theta)) < \mathbf{k}\}. \tag{3.2}$$

We refer to the threshold denoted as k in Equation 3.2 as *SimThres*.

Statistical *SimThres* In a statistical manner, we set the *SimThres* based on the similarity values with the entire set of users. It can be expressed as $k_{\text{stat}} = \text{sim}(f(s_i;\theta), f(s_K;\theta))$. Depending on K, we defined k_{stat} as the median, quantile, or the top 10% value of the similarity values. This approach bears resemblance to the concept of the most offending sample introduced in PromCSE[17], specifically in the context of computing the Energy-Hinge loss. However, while PromCSE[17] aimed to extract negative samples that are difficult to differentiate from positive pairs, our objective was to construct explicit negative samples. Therefore, we selected samples with low similarity values. Through experimentation, we found that setting the *SimThres* to the top 10% value, i.e., $K = 0.9 \times |\mathcal{S}|$, proved to be effective. As an implementation detail, we calculated expectation value of $\text{sim}(f(s_i;\theta), f(s_K;\theta))$ for each batch and input to successive threshold in next epoch.

Learnable *SimThres* where statistically regularized In order to obtain the value of $k_{\text{learn}} = g(\{f(s_i;\theta)\}_i)$ as a trainable parameter, we introduce an additional submodel g that enables us to set the *SimThres* adaptively. Rather than relying solely on prior information, we construct a parameterized submodel, denoted as g, specifically designed to determine the 1-dimensional threshold value. We employ a straightforward Multi-Layer Perceptron (MLP) architecture for submodel g. By incorporating this trainable submodel, we empower our methodology to learn and adapt the value of k_{learn} during the training process, enhancing the flexibility and effectiveness of our approach in capturing the desired negative sample characteristics.

To control the value of *SimThres* and acknowledge that $\text{sim}(f(s_i;\theta), f(s_j;\theta))$ is also a learned quantity, we introduce a regularization term to our methodology. The regularization term, denoted as $\mathcal{L}_{reg}$, is implemented by adding a loss term of the form $\|g(\{f(s_i;\theta)\}_i) - k_{\text{stat}}\|_2^2$, where k_{stat} serves as the control parameter. This additional term allows us to enforce a desired level of regularization on the value of $g(\{f(s_i;\theta)\}_i)$. We refer to the resulting value $g(\{f(s_i;\theta)\}_i)$ as k_{learn}, representing the regulated and learned threshold. By incorporating this regularization mechanism, we aim to strike a balance between adapting the threshold value through training and maintaining consistency with the desired statistical properties represented by k_{stat}.

3.2 Model Ensemble

In our methodology, we perform model ensemble based on different types of loss functions. BERT4Rec[28], CL4SRec[32], DuoRec[25], and EHRec share the basic

structure of BERT4Rec[28]. However, with the introduction of contrastive learning loss, there are differences in the refined process of pushing and pulling representations.

Figure 3.1 represents the overview of parameter merging process. Sharing structure of model, where parametrized with diverse learning framework, we were able to take advantage of ensemble. In the upcoming chapter, we experimentally examine how these changes in the learning framework and loss computation affect the form of representations. Furthermore, inspired by previous studies demonstrating the effectiveness of ensemble models trained using various learning methods, we apply parameter merging techniques, namely Parameter Averaging, described in Section 3.2.1, and Fisher-weighted Parameter Merging, described in Section 3.2.1, to combine these models.

3.2.1 Understanding Model Ensemble

Following work of [22], let us consider a scenario where we have models with the same structure, denoted as $model_1, model_2, \cdots, model_M$, with corresponding parameters $\theta_1, \theta_2, \cdots, \theta_M$. Our objective is to find the parameter θ^* that maximizes the joint likelihood of the posteriors of these parameters. In this study, we aim to apply this research effort to the field of sequential recommendation.

The posterior of θ_i can be represented as $p(\theta_i|D)$, where $D = \{x_i, y_i\}_i$ denotes the data. Since obtaining this posterior directly is generally challenging, it can employ approximation methods such as Laplace Approximation to make assumptions and seek the parameter θ^*[21][7]. Furthermore, the joint posterior can be expressed

using Bayesian Rule as follows.

$$p(\theta_1, \theta_2, \cdots, \theta_M \mid \theta) = p(\theta \mid \theta_1, \theta_2, \cdots, \theta_M) p(\theta_1) p(\theta_2) \cdots p(\theta_M) / p(\theta) \tag{3.3}$$

Let us interpret the process of finding θ^* as maximizing the joint likelihood, $p(\theta \mid \theta_1, \theta_2, \cdots, \theta_M)$. Assuming that $p(\theta_i \mid D)$ follows a Gaussian distribution, we set the mean of this Gaussian distribution as the observed θ_i and examine the procedure for averaging parameters and Fisher Merging separately, depending on the method used to assume the variance.

Averaging Parameters Let us assume that the posterior $p(\theta_i \mid D)$ follows a Gaussian distribution $\mathcal{N}(\hat{\theta}_i, I)$. Here, $\hat{\theta}_i$ represents the parameters of the trained $model_i$, and I denotes the identity matrix. In this case, the desired solution θ^* can be obtained as the average of the parameters of the candidate models, as shown in Equation 3.4.

$$\theta^* = \arg\max_{\theta} \sum_i \log p(\theta \mid \theta_i, I) = \frac{1}{M} \sum_i \theta_i \tag{3.4}$$

Fisher Merging Let us consider the posterior $p(\theta_i|D)$ following a Gaussian distribution $\mathcal{N}(\hat{\theta}_i, H^{-1})$. Here, $\hat{\theta}_i$ represents the parameters of the trained $model_i$, and H^{-1} corresponds to the Hessian matrix of θ_i obtained through the second-order Taylor expansion at the mode of the posterior. It has been established that the Hessian matrix in this distribution coincides with the Fisher information, but for computational efficiency, we only utilize the diagonal elements of the Fisher matrix.

The desired solution θ^* can be expressed as shown in Equation 3.5, capturing the essence of the Fisher likelihood.

$$\theta^* = \arg\max_{\theta} \sum_i \lambda_i \log p(\theta \mid \theta_i, F_i), \tag{3.5}$$

$$\text{where } F_i = \mathbb{E}_x \left[\mathbb{E}_{y \sim p_{\theta^i}(y|x)} \nabla_\theta \log p_\theta(y|x) \nabla_\theta \log p_\theta(y \mid x)^T \right] \tag{3.6}$$

The closed-form solution for θ^* can be obtained as shown in Equation 3.7, which directly incorporates the Fisher matrix. In practice, we utilize an empirical estimate of the Fisher matrix, denoted as $\hat{F}$, as shown in Equation 3.8[18].

$$\theta^{*(j)} = \frac{\sum_i \lambda_i F_i^{(j)} \theta_i^{(j)}}{\sum_i \lambda_i F_i^{(j)}}, \tag{3.7}$$

$$\text{where } F_i = \frac{1}{N} \mathbb{E}_{y \sim p_\theta(y|x)} (\nabla_\theta \log p_\theta(y \mid x))^2 \tag{3.8}$$

3.2.2 Applying Model Ensemble

By expressing the Fisher matrix we intend to compute in eq.3.7 in terms of recommendation factors, we can decompose it into the following components:

$$\begin{aligned} &\mathbb{E}_{x_i} \mathbb{E}_{y \sim p_\theta(y|x_i)} \left(\nabla_\theta \log p_\theta \left(y \mid x_i \right) \right)^2 \\ &= \frac{1}{N} \sum_i \sum_j p_\theta \left(y_j \mid x_i \right) \left(\nabla_\theta \log p_\theta \left(y_j \mid x_i \right) \right)^2 \\ &= \frac{1}{|\mathcal{U}|} \sum_i^{|\mathcal{U}|} \sum_j^{|\mathcal{V}|} p_\theta \left(v_j \mid s_i \right) \left(\nabla_\theta \log p_\theta \left(v_j \mid s_i \right) \right)^2 . \end{aligned} \tag{3.9}$$

There are two computational challenges associated with the above equation. First, calculations need to be performed for each individual sample s_i. Second, calculations need to be performed for each item v_j within a single sample. The reason

why these points acts as a drawback in recommendation systems is due to the large number of users and items in the data. For instance, in the case of MovieLens-1M dataset[13], there are about 6000 users and 3500 items. However, performing Fisher matrix calculations that require differentiation with respect to θ for each user and item becomes a computational burden.

Sampling sequences

Batch-wise Computation To address the first challenge of performing computations on individual samples, we reinterpret the equation and carry out the calculations on a batch basis. It should be noted that $p_\theta(v_j|s_i)$ can vary for each sample s_i. Therefore, we perform the sorting of $p_\theta(v_j|s)$ to address this variation, where BS indicates batch size:

$$\begin{aligned}
&\sum_{i}^{|\mathcal{U}|}\sum_{j}^{|\mathcal{V}|} p_\theta(v_j \mid s_i)\left(\nabla_\theta \log p_\theta(v_j \mid s_i)\right)^2 \\
&= \sum_{\mathrm{BS}_k}\sum_{j}^{|\mathcal{V}|}\left(\sum_{i}^{\mathrm{BS}_k} p_\theta(v_j|s_i)\right)\left(\nabla_\theta \sum_{i}^{\mathrm{BS}_k} \log p_\theta(v_j|s_i)\right)^2 .
\end{aligned} \tag{3.10}$$

Sampling items

To alleviate the computational burden associated with iterating over all j values, which scales with $|\mathcal{V}|$, we employ a sampling-based approach within the methodology. This sampling strategy aims to reduce the computational cost while maintaining the representativeness of the calculations.

Random Sampling We compute the eq.3.2.2 by randomly sampling j from the total number of items. This process was performed to calculate the Fisher matrix

without any specific assumptions or prior knowledge.

Top-k Sampling The probability which is output by the model can be interpreted as the preference or likelihood of the recommended items for a given sample. Based on this interpretation, we select a set of n items that are most likely to be of interest to the corresponding user, i.e. $p_\theta\left(v_j \mid s_i\right)$. Subsequently, we compute the Fisher matrix with these selected items as the focal points. By focusing on this subset of items that are expected to be of highest interest, we aim to capture the relevant information for optimizing the model's performance effectively.

$$\begin{aligned} &\sum_{j}^{|\mathcal{V}|} p_\theta\left(v_j \mid s\right)\left(\nabla_\theta \log p_\theta\left(v_j \mid s\right)\right)^2 \\ &\approx \sum_{j}^{\text{top-}k} p_\theta\left(v_j \mid s\right)\left(\nabla_\theta \log p_\theta\left(v_j \mid s\right)\right)^2 . \end{aligned} \tag{3.11}$$

Model-based Sampling To select a subset of items for further analysis, we randomly sampled items based on their conditional probability $p_\theta(v_j|s_i)$ using a weighted random selection process. The selection probability of each item was determined by its associated probability stored in the model's output. By selecting items with higher probabilities, we focused on a specific number of items that were more likely to align with the user's preferences or interests. This allowed us to analyze and evaluate the subset of items based on their associated probabilities obtained from the model's output. With N denotes the sample size, his approximation

can be represented as:

$$\begin{aligned} &\mathbb{E}_{y\sim p_\theta(y|x)} \left(\nabla_\theta \log p_\theta \left(y \mid x\right)\right)^2 \\ &\approx \frac{1}{N} \sum_{v_j \sim p_\theta(v_j|s)}^{N} \left(\nabla_\theta \log p_\theta \left(v_j \mid s\right)\right)^2 . \end{aligned} \tag{3.12}$$

Calculate with target item We compute the Fisher matrix based on the target item, disregarding other items with limited direct relevance. By employing this approach, we focus solely on the target item and its associated information to calculate the Fisher matrix. Our rationale behind this decision is to prioritize the target item's impact on the model's optimization process, as it is directly linked to the specific objective or task at hand. Consequently, we exclude items with minimal direct relevance to ensure a more targeted and meaningful computation of the Fisher matrix.

$$p_\theta \left(v_j^* \mid s\right) \left(\nabla_\theta \log p_\theta \left(v_j^* \mid s\right)\right)^2 , \tag{3.13}$$

where v_j^* is the target item.

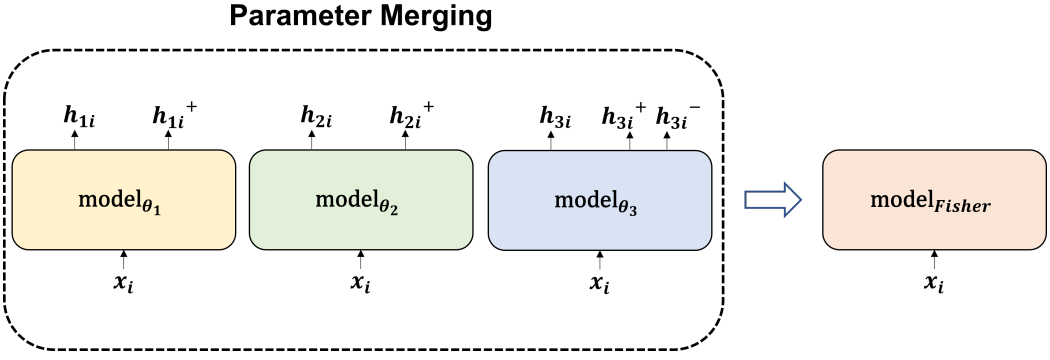


Figure 3.1: Parameter Merging

Chapter 4

Experiment

4.1 EHRec Analysis

4.1.1 Comparison with Baseline Models

We conducted comparisons with existing studies in the field of Sequential Recommendation using contrastive learning. The comparisons were performed on the MovieLens-1M dataset[13] denoted as ML-1M, and we measured the performance metric NDCG@10 (Normalized Discounted Cumulative Gain at 10), the ranking evaluation metric, under the Full[19], Random100, and Popular100[16] settings, predicting next item recommendation task as previous work[14, 28, 32, 25]. The last movie is considered as the test set, and the validation data is used to predict the preceding movies. During training, we adopt a masked language modeling approach similar to BERT[8], where we mask certain movies in the sequentially ordered list and task the model with predicting them. The evaluation method used in this study is the Normalized Discounted Cumulative Gain at 10 (NDCG@10), which is a ranking-based evaluation approach[14]. It ranks the top 10 items pre-

Table 4.1: Comparison to Baseline Models. The **bold** represents the best performance compared to other models in the specific metric setting.

			FULL	RANDOM	POPULAR
	CL4SRec		0.0955	0.5156	**0.0479**
sup.	DuoRec		0.1348	0.5613	0.0454
	EHRec(Ours)	k_{stat}	0.1338	0.558	0.0458
		k_{learn}	**0.1357**	**0.5676**	0.0449
-	DuoRec		**0.1382**	0.5581	0.0458
	EHRec(Ours)	k_{stat}	0.1365	**0.5638**	0.0454
		k_{learn}	0.1337	0.5601	0.046

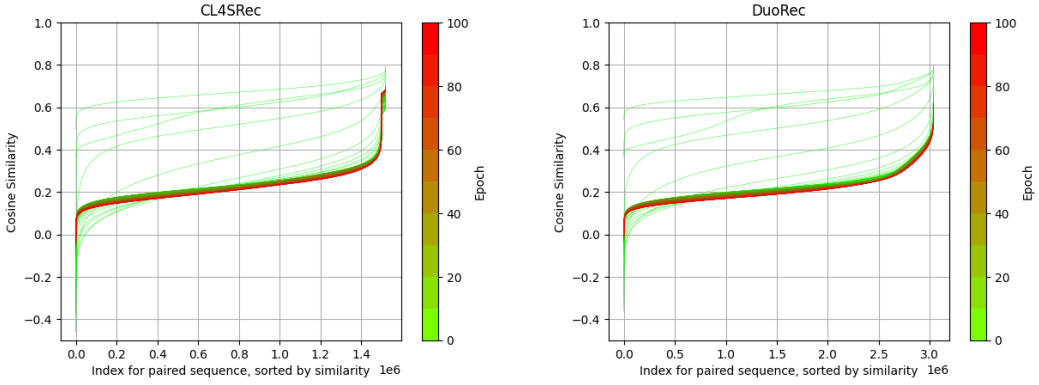
dicted by the model based on their perceived preference and considers the actual ranking of the preferred items. A higher NDCG value, closer to 1, indicates better performance. Different NDCG values can be obtained depending on the selection of items, such as from the full item pool, a random set of 100 items, or the top 100 most popular items.

Looking at Table 4.1, we can observe that EHRec achieved the best performance in the Random setting. However, when comparing the performance of EHRec and DuoRec[25] in the Random setting, we can see that while the negative sample construction led to performance improvement when using the same positive pairs, it fell short in the Full and Popular settings.

4.1.2 SimThres

Selecting k_{stat} We initially examined the changes in similarity values. We observe as the model's training progresses, there is a distinct shift in the distribution of high similarity values, as shown in Figure 4.1. We identified this range as the region where the differentiation between negative and false negative pairs takes place. The rapid change was observed near the top 10 percentile range. Therefore, we set the

threshold, k_{stat}, to 0.9, considering the bottom 90 percentile as negative pairs.

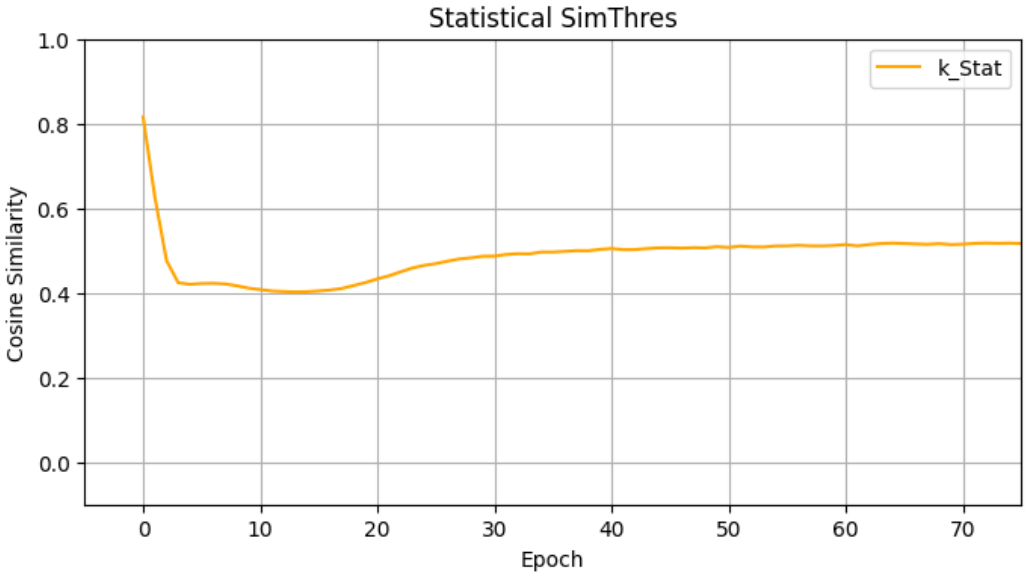


(a) Embedded Sequences of CL4SRec[32] (b) Embedded Sequences of DuoRec[25]

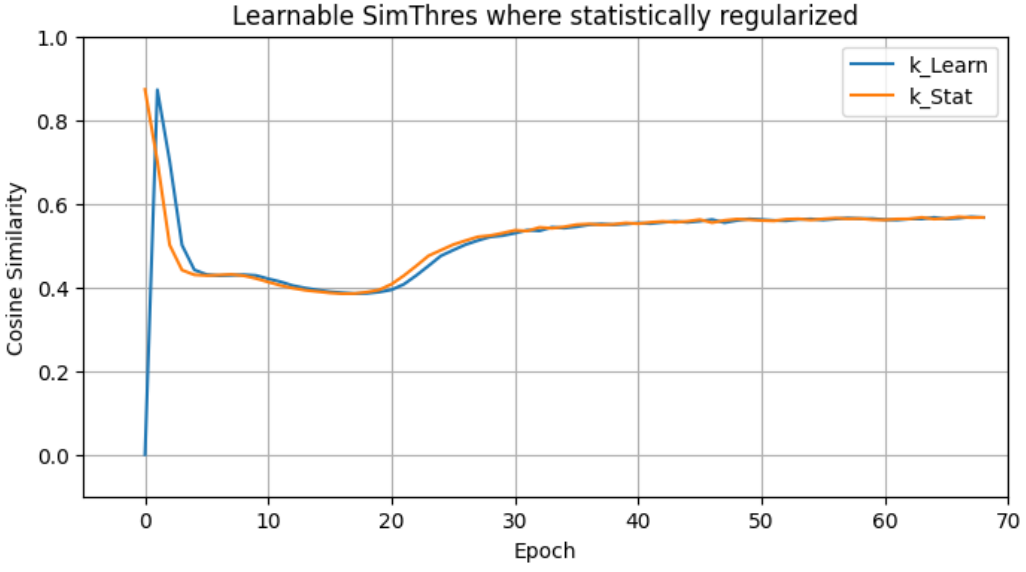
Figure 4.1: Similarity distrubution

Effect of Regularizer k_{stat} in k_{learn} While the statistical approach for determining a similarity threshold is effective, it has limitation that it assumes that the same threshold value should be shared across all sequences. However, similarity is computed for each batch in practical training scenarios, and the local similarity distribution may differ from the overall distribution used to determine the global threshold k_{stat}. This lead to the necessity of each sequence requiring a different threshold value based on the given sample. To address this, we utilize a small MLP model to learn the optimal threshold value for each batch and sequence. As mentioned earlier, rapid threshold learning can help reduce bias in overall training. Therefore, we incorporate a regularizer for k_{learn}, utilizing k_{stat}. The effect of a regularizer is shown in Figure 4.2.

As k_{stat} is calculated based on the similarity values of previous epoch, k_{learn} follows the value of k_{stat}. Also, with k_{stat} regularization, the final value differs and leads to difference in distibution between sequences.



(a) Similarity Threshold of k_{stat}



(b) Similarity Threshold of k_{learn}

Figure 4.2: Similarity Threshold

4.1.3 Item Embedding Quality

To assess the quality of embeddings, we utilized the performance metric of uniformity, commonly used in contrastive learning evaluations. Building upon previous research efforts such as DuoRec[25], which focused on improving uniformity, we examined the uniformity metric in the context of EHRec. Given EHRec's emphasis on the composition of negative samples rather than positive pairs, we specifically investigated the impact on uniformity. The experimental setup consisted of measuring the uniformity for baseline models employing different augmentation techniques in contrastive learning, and subsequently evaluating the uniformity when incorporating EHRec with each augmentation method. The uniformity metric[29], denoting the degree of dispersion, is represented as a value, where a lower value indicates a more desirable outcome Figure 4.3 presents the obtained results. For convenience, we plotted the negative of uniformity in the figure.

When comparing against the same positive sampling strategy, we observe that employing our proposed adaptive method for constructing negative samples leads to reduced uniformity, indicating a more widely dispersed formation of item embeddings.

4.2 Model Ensemble Analysis

4.2.1 Results of Model Merging

Examine the results through Table 4.2 and Table 4.3. Table 4.2 presents the results obtained by training models, namely BET4Rec[28], CL4SRec[32], and DuoRec[25]. We merge these models using Fisher methods. While Table 4.2 demonstrates the results of models trained solely from scratch. Table 4.3 represents the results of fine-

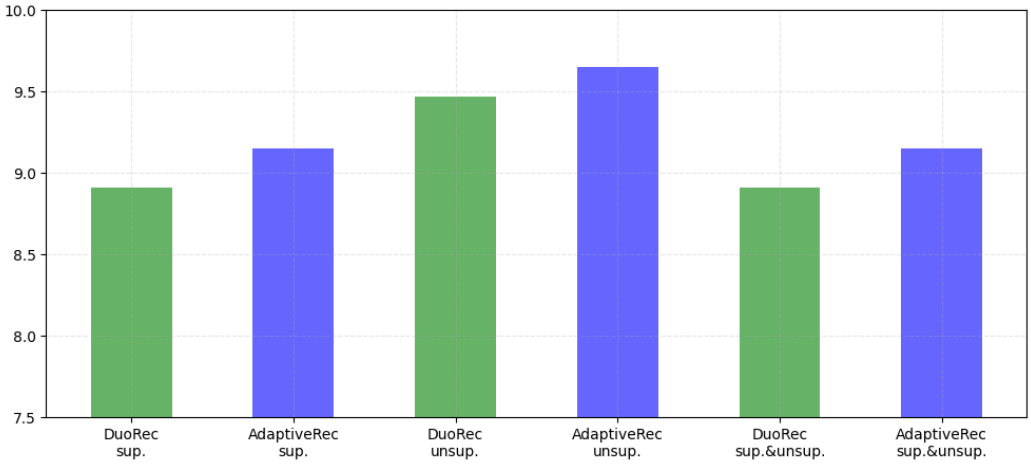


Figure 4.3: Comparing Uniformity with Baseline Model

tune setting. We train the baseline model without contrastive loss for 20 epochs, which is the convergence point of the baseline experiment without any additional contrastive loss, similar to BERT4Rec. Following this, each model; BERT4Rec[28], CL4SRec[32], DuoRec[25], underwent fine-tuning according to their respective methods, and the results were merged using Fisher methods. In both conditions, we fine-tuned addtional epoch after merging process.

Fisher merge fails to improve the performance of individual models in baseline setting. When Fisher merge is applied during the fine-tuning setting, it leads to improved performance compared to individual models. This finding aligns with previously reported phenomena[9] where individual models tend to achieve higher performance than merged model in the baseline setting. However, the results of Fisher merge in the fine-tuning setting show comparable performance with the individual models in baseline setting, while the individual recipe models of fine-tuning setting do not exceed. Also, the results indicate that even for models that have not been sufficiently trained such as CL4SRec in our setting, merging parameters resulted

in comparable performance to other models, demonstrating robustness.

Table 4.2: Results of parameter merging; Fisher-merge on Baseline Settings. The recipes used for merging were trained for the same number of epochs. 'POS.' refers to the method of constructing positive pair, 'sup' to supervised augmentation and '-' to supervised and unsupervised augmentation in subsection 2.1.2

MODEL	POS	FULL	RANDOM	POPULAR
BASELINE		**0.1398**	**0.5651**	**0.0482**
CL4SREC		0.0955	0.043	0.0429
DUOREC	SUP	0.1348	0.5575	0.044
	UNSUP	0.1301	<u>0.5592</u>	0.0438
	-	<u>0.1382</u>	0.5588	<u>0.0464</u>
FISHER		0.1289	0.5495	0.0472

Table 4.3: Results of parameter merging; Fisher-merge on fine-tune Settings. The recipes used for merging were trained on baseline model (without contrastive loss) and fine-tuned on each model.

MODEL	POS	FULL	RANDOM	POPULAR
BASELINE		0.135	0.5573	0.0426
CL4SREC		0.0585	0.0513	**0.0466**
DUOREC	SUP	0.1346	0.5547	<u>0.0454</u>
	UNSUP	<u>0.1358</u>	<u>0.5594</u>	0.0445
	-	0.1351	0.554	0.0423
FISHER		**0.1386**	**0.5618**	0.0428

Table 4.4: Effect of Sampling Method and Sampling Size. We merge models in settings of Table 4.3, the fine-tune setting. We merged models with 4 sampling methods; random sampling, top-k sampling, model-based sampling, and calculate on target item, on 3 different sampling size; n=10, n=30 and n=50. **Bold** represents the best variant in each evaluation setting, and underlines indicates the second best variation.

	sample size	FULL		RANDOM		POPULAR	
		NDCG@10	NDCG@20	NDCG@10	NDCG@20	NDCG@10	NDCG@20
baseline		0.135	0.1601	0.5573	0.5786	0.0426	0.0706
CL4SRec		0.0585	0.0751	0.0513	0.043	**0.0466**	0.0701
DuoRec (sup.)		0.1346	0.1591	0.5547	0.58	0.0454	0.068
DuoRec (unsup.)		0.1358	0.1609	0.5594	0.5782	0.0445	0.0742
DuoRec (sup.&unsup.)		0.1351	0.1599	0.554	0.5732	0.0423	0.0724
random sampling	10	0.1379	0.1638	0.5606	0.5825	0.0457	0.0691
	30	0.1366	0.1624	0.5584	0.58	0.0477	**0.0726**
	50	0.1386	0.1636	0.5598	0.5813	0.0419	0.0419
top-k sampling	10	0.1364	0.1624	0.5602	0.5817	0.0446	0.0689
	30	0.1373	0.1616	**0.5637**	**0.5835**	0.0457	0.0708
	50	**0.1387**	0.1635	0.5592	0.5807	0.0424	0.0672
model-based sampling	10	0.1358	0.1619	0.5564	0.5782	0.044	0.0696
	30	0.1385	**0.1646**	0.5579	0.5784	0.0446	0.0689
	50	0.138	0.1632	0.5605	0.5814	0.0465	0.0719
calculate on target item		0.1386	0.1628	0.5618	0.5806	0.0428	0.0725

4.2.2 The Validity of Batch-wise Computation

We performed batch-wise computations with the aim of implementing an efficient Fisher matrix calculation. Compared to computing on individual samples, grouping samples into batches allowed us to achieve computational efficiency.

The following Figure 4.6 illustrates the method for minimizing errors when performing calculations on a batch basis. The figure demonstrates that within a batch containing 10 samples, denoted as s_i, there is a phenomenon where the probabilities of item v_j decrease in a similar manner. By sorting the samples s_i based on the probability of v_j, even when grouping them into batches, it is possible to minimize the error described by the eq.3.10. Furthermore, the figure illustrates the rationale behind top-k sampling. For the top-k items, the probabilities hold meaningful information, whereas for the remaining items, the probabilities are nearly zero or close to it.

4.2.3 Effect of Sampling Methods and Size

To investigate the effect of sampling methods, we conduct experiments by varying the number of sampled samples and the sampling techniques employed. Specifically, we consider three sample sizes: $n = 10, n = 30$ and $n = 50$, and four different sampling methods: random sampling, top-k sampling, model-based sampling, and calculate with target item. The results of these experiments can be observed in Table 4.4. The table provides insights into the performance of each sampling method under different sample sizes, allowing us to analyze their respective effects on the task at hand. Note that this result is calculated on batched data. To examine the results of parameter merging, we conducted experiments in fine-tuning setting, explained in 4.3.

The experiments revealed effective ensemble results, particularly showcasing the robust performance of CL4SRec[32]. Despite having significantly lower performance compared to other models during the parameter merging process, the model with poor performance exhibited robust performance in the Fisher merge results. Regarding the sampling methods, top-k sampling demonstrated the best performance. This can be attributed to the concentration of probabilities assigned to specific items by the model, effectively approximating the Fisher criterion sought in the evaluation. Also, the model-based sampling method exhibits a more pronounced improvement in performance as the sampling size increases compared to other models. We interpret these results as being rooted in the direct interpretation of the equation defined for Fisher merging. Interestingly, despite the fact that calculating Fisher matrix on target item has a single sample, the method demonstrated sampling effectiveness by achieving good performance even with a small sample size. These findings shed light on the interpretation of experimental results in the context of deep learning research.

4.2.4 Computational Cost

Figure 4.4 demonstrates computational cost in terms of time consumed during calculating Fisher matrix for single model. The concept of parameter merging involves additional computation on the existing parameters. Therefore, it is important to ensure efficiency in this process. To achieve efficiency, considerations such as calculating the Fisher matrix in batch units and performing sampling are necessary. It is observed that, except for the calculation on the target item, the computational complexity increases linearly with the sampling size. As for the calculation on the target item, the sampling size remains fixed at 1 since each sequence has a

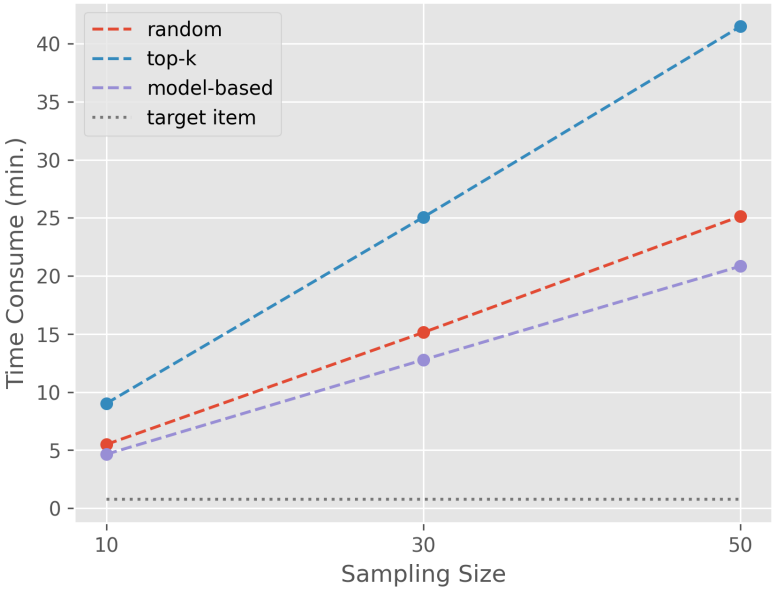


Figure 4.4: Measured Time Consumed for Each Sampling Method and Size. We sample items from MovieLens-1M dataset. Time is measured in batch-wise setting, where batch size is 256.

single target item. Thus, our research is significant as it approximates the Fisher matrix calculation with a much smaller number of items (around 3000) compared to calculating it on the entire item set.

4.2.5 Visualization of Merged Weights

We present a visual illustration to aid in the intuitive understanding of the merged weights. Figure 4.5 represents the fine-tuning setting of 4.3, where the three centroids correspond to the weights of individual models. The plane visualized in 4.5 encompasses these three weights. The scattered points, projected onto the plane, depict 100 samples drawn from $\mathcal{N}(\theta_m, F_m)$. It is observed that the baseline weight exhibits the largest variance. This can be attributed to the experimental setup where

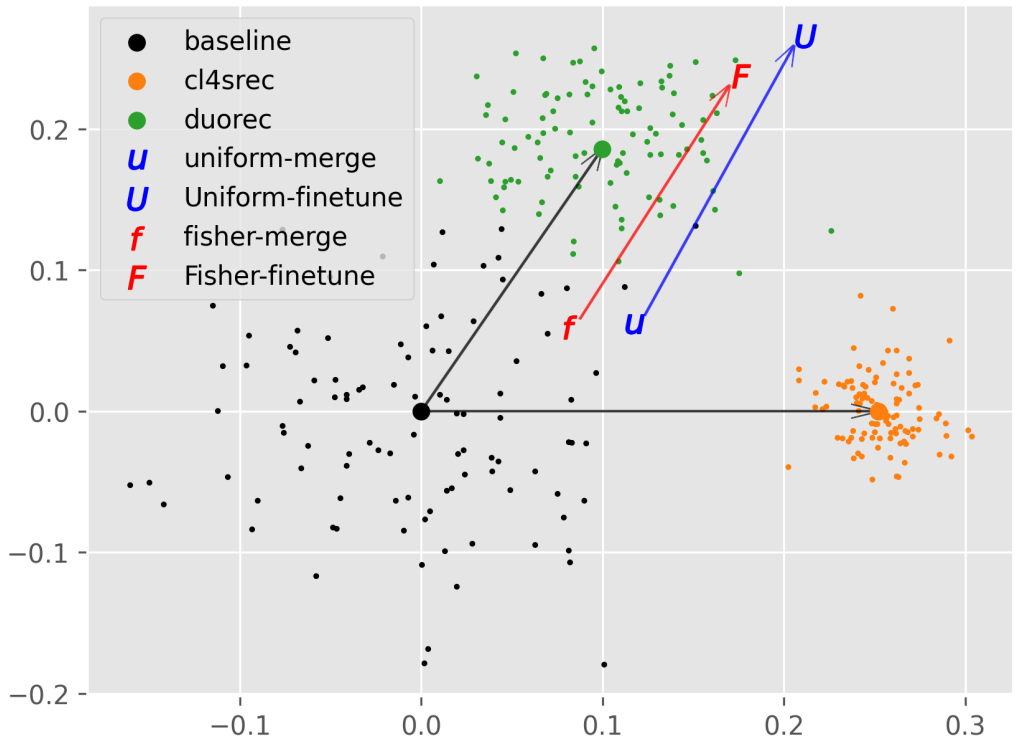


Figure 4.5: Visualization for Weights of Merged Models. Based on the plane containing 64-dimension parameters of three model, we visualised its weight, 100 samples from each posteriors and merged parameters

the baseline is pre-trained and then fine-tuned with CL4SRec[32] and DuoRec[25]. The weights obtained through uniform merging are represented as the average of the three centroid points, while the weights obtained through Fisher merging take into account the variances of these recipe weights. It can be seen that the weights obtained through Fisher merging considered posterior and variance with Laplace approximation and provides nice initial point for fine-tune.

4.2.6 Motivation : Error Inconsistency

Table 4.5: Effect of Constructing Pair of Contrastive Loss (%). We observe that models with divergent training methodologies exhibit distinct generalization behavior, resulting in highly uncorrelated errors.

	SIMILAR		DISSSIMILAR
CL4SREC	8.05	<	11.41
DUOREC	8.67	<	11.18

Previous research[12, 33] demonstrated the increased effectiveness of ensemble methods as error inconsistency grows. Building upon the existing research discourse, we conducted the current experiment. In this study, we analyze the impact of Fisher merging in the context of sequential recommendation systems, attributing its effectiveness to the selection of recipe models trained using different frameworks.

In our experiments, we employ a model based on the BERT4Rec[28] architecture as our baseline. To enhance the performance of the model, we apply various data augmentation techniques to enable contrastive learning.

To analyze the effects of contrastive loss, we divide the training frameworks into two categories: similar and dissimilar. The similar learning frameworks are trained using the same loss function but with slight variations such as different seeds and hyperparameters, indicating the relationship among models trained with small changes. On the other hand, the dissimilar learning frameworks involve different data augmentation techniques, resulting in variations in the construction of positive and negative pairs for contrastive loss[29].

Error inconsistency[11] refers to the percentage of data where two models have different classification results, with one model making correct predictions while the

other model makes incorrect predictions. Since we are not dealing with classification, we considered a model to have made a correct prediction if the value of NDCG@10 is above 0.5.

By comparing the error inconsistency between similar framework and dissimilar framework, we observe the effectiveness of contrastive loss. An observation that can be inferred from Table 4.5 is that the constructing positive pair for contrastive loss significantly affects the similarity of the samples that the models predict accurately. As the method for constructing positive pair varies, the models demonstrate a considerable difference in their ability to predict samples correctly. This finding highlights the sensitivity of the models to the specific construction of the contrastive loss, which in turn impacts their predictive performance.

4.2.7 Robustness of Fisher Merging; Recipe Selection

We compared two different recipe selection in Table 4.6; Fisher merged parameters with least performance model and Fisher merged parameters without the model. In our experimental setup, CL4SRec did not exhibit superior performance compared to other models, considering the chosen hyperparameter settings and other factors. Therefore, we aim to leverage the elements of the recipe to demonstrate the robustness effect of Fisher merge. Our findings confirm that by removing underperforming models as individual components and applying Fisher merge, the resulting ensemble demonstrates robustness.

4.2.8 Visualization of Sorted Probability

The figure displays the sorted probabilities of the top 50 items for 10 sequences, where single line represents single sequence. The cumulative probability values for

Table 4.6: Ablation Study of Results of parameter merging; Fisher-merge on fine-tune Settings. The recipes used for merging were trained on baseline model (without contrastive loss) and fine-tuned on each model. Ablation Study illustrates the situation of recipe without the model with least performance.

MODEL	POS	FULL	RANDOM	POPULAR
BASELINE		0.135	0.5573	0.0426
CL4SREC		0.0585	0.0513	<u>0.0466</u>
DUOREC	SUP	0.1346	0.5547	<u>0.0454</u>
	UNSUP	0.1358	0.5594	0.0445
	-	0.1351	0.554	0.0423
FISHER (WITH)		**0.1386**	**0.5618**	0.0428
FISHER (W.O.)		<u>0.1373</u>	<u>0.5603</u>	**0.0487**

sample sizes of 10, 30, and 50 are 0.381, 0.569, and 0.658, respectively. With the exception of a few largest ones, the majority of probabilities approximate 0.

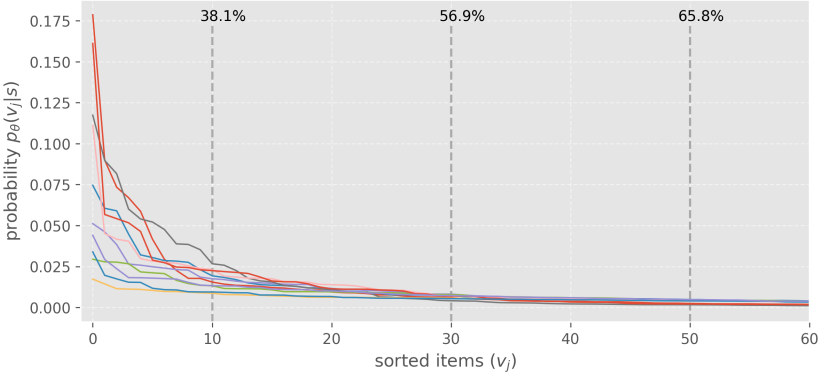


Figure 4.6: Sorted Probability $p_\theta(v_j|s_i)$.

Chapter 5

Conculsion

This thesis addresses the challenges in sequential recommendation systems by proposing two significant contributions. Firstly, we introduce an advanced approach to contrastive learning to overcome data sparsity limitations. By adaptively selecting threshold based on sequence similarity and construct negative samples, we enhance the quality of item embeddings and mitigate false negative problems. Secondly, we present a novel ensemble technique, Fisher merging, for sequential models, enabling robust fine-tuning through parameter merging. Our experimental results demonstrate the effectiveness of these proposed methods in improving recommendation performance. These contributions have the potential to advance the field of sequential learning and recommendation systems, offering valuable insights for future research and practical applications.

Bibliography

[1] L. Breiman, *Bagging predictors*, Machine learning, 24 (1996), pp. 123–140.

[2] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A simple framework for contrastive learning of visual representations*, in International conference on machine learning, PMLR, 2020, pp. 1597–1607.

[3] Y.-S. Chuang, R. Dangovski, H. Luo, Y. Zhang, S. Chang, M. Soljačić, S.-W. Li, W.-t. Yih, Y. Kim, and J. Glass, *Diffcse: Difference-based contrastive learning for sentence embeddings*, arXiv preprint arXiv:2204.10298, (2022).

[4] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, *Electra: Pre-training text encoders as discriminators rather than generators*, arXiv preprint arXiv:2003.10555, (2020).

[5] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, *Supervised learning of universal sentence representations from natural language inference data*, arXiv preprint arXiv:1705.02364, (2017).

[6] R. Dangovski, L. Jing, C. Loh, S. Han, A. Srivastava, B. Cheung, P. Agrawal, and M. Soljačić, *Equivariant contrastive learning*, arXiv preprint arXiv:2111.00899, (2021).

[7] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig, *Laplace redux-effortless bayesian deep learning*, Advances in Neural Information Processing Systems, 34 (2021), pp. 20089–20103.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805, (2018).

[9] M. A. Ganaie, M. Hu, A. Malik, M. Tanveer, and P. Suganthan, *Ensemble deep learning: A review*, Engineering Applications of Artificial Intelligence, 115 (2022), p. 105151.

[10] T. Gao, X. Yao, and D. Chen, *Simcse: Simple contrastive learning of sentence embeddings*, arXiv preprint arXiv:2104.08821, (2021).

[11] R. Geirhos, K. Meding, and F. A. Wichmann, *Beyond accuracy: quantifying trial-by-trial behaviour of cnns and humans by measuring error consistency*, Advances in Neural Information Processing Systems, 33 (2020), pp. 13890–13902.

[12] R. Gontijo-Lopes, Y. Dauphin, and E. D. Cubuk, *No one representation to rule them all: Overlapping features of training methods*, arXiv preprint arXiv:2110.12899, (2021).

[13] F. M. Harper and J. A. Konstan, *The movielens datasets: History and context*, Acm transactions on interactive intelligent systems (tiis), 5 (2015), pp. 1–19.

[14] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, *Neural collaborative filtering*, in Proceedings of the 26th international conference on world wide web, 2017, pp. 173–182.

[15] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, *Parameter-efficient transfer learning for nlp*, in International Conference on Machine Learning, PMLR, 2019, pp. 2790–2799.

[16] J. Huang, W. X. Zhao, H. Dou, J.-R. Wen, and E. Y. Chang, *Improving sequential recommendation with knowledge-enhanced memory networks*, in The 41st international ACM SIGIR conference on research & development in information retrieval, 2018, pp. 505–514.

[17] Y. Jiang, L. Zhang, and W. Wang, *Improved universal sentence embeddings with prompt-based contrastive learning and energy-based learning*, in Findings of the Association for Computational Linguistics: EMNLP 2022, 2022, pp. 3021–3035.

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., *Overcoming catastrophic forgetting in neural networks*, Proceedings of the national academy of sciences, 114 (2017), pp. 3521–3526.

[19] W. Krichene and S. Rendle, *On sampled metrics for item recommendation*, in Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020, pp. 1748–1757.

[20] P. H. Le-Khac, G. Healy, and A. F. Smeaton, *Contrastive representation learning: A framework and review*, Ieee Access, 8 (2020), pp. 193907–193934.

[21] D. J. MacKay, *A practical bayesian framework for backpropagation networks*, Neural computation, 4 (1992), pp. 448–472.

[22] M. S. Matena and C. A. Raffel, *Merging models with fisher-weighted averaging*, Advances in Neural Information Processing Systems, 35 (2022), pp. 17703–17716.

[23] A. Natekin and A. Knoll, *Gradient boosting machines, a tutorial*, Frontiers in neurorobotics, 7 (2013), p. 21.

[24] A. v. d. Oord, Y. Li, and O. Vinyals, *Representation learning with contrastive predictive coding*, arXiv preprint arXiv:1807.03748, (2018).

[25] R. Qiu, Z. Huang, H. Yin, and Z. Wang, *Contrastive learning for representation degeneration problem in sequential recommendation*, in Proceedings of the fifteenth ACM international conference on web search and data mining, 2022, pp. 813–823.

[26] N. Reimers and I. Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, arXiv preprint arXiv:1908.10084, (2019).

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, The journal of machine learning research, 15 (2014), pp. 1929–1958.

[28] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, *Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer*, in Proceedings of the 28th ACM international conference on information and knowledge management, 2019, pp. 1441–1450.

[29] T. Wang and P. Isola, *Understanding contrastive representation learning through alignment and uniformity on the hypersphere*, in International Conference on Machine Learning, PMLR, 2020, pp. 9929–9939.

[30] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, et al., *Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time*, in International Conference on Machine Learning, PMLR, 2022, pp. 23965–23998.

[31] X. Wu, C. Gao, Z. Lin, J. Han, Z. Wang, and S. Hu, *Infocse: Information-aggregated contrastive learning of sentence embeddings*, arXiv preprint arXiv:2210.06432, (2022).

[32] X. Xie, F. Sun, Z. Liu, S. Wu, J. Gao, J. Zhang, B. Ding, and B. Cui, *Contrastive learning for sequential recommendation*, in 2022 IEEE 38th international conference on data engineering (ICDE), IEEE, 2022, pp. 1259–1273.

[33] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, *Understanding neural networks through deep visualization*, arXiv preprint arXiv:1506.06579, (2015).

국문초록

온라인 플랫폼과 서비스의 급격한 성장으로 인해, 추천 시스템은 사용자의 선호도에 기반하여 관련 아이템을 식별하는 데 필수적인 역할을 한다. 특히, 순차적 추천 분야에서는 사용자의 시간에 따라 변화하는 선호도를 포착하는 것을 목표이다. 이를 해결하기 위해, 추천 시스템에서 한정된 사용자-아이템 상호작용으로 인한 데이터 희소성 문제에 대응하기 위해 대조적 학습 방법이 제안되었으나, 이러한 방법들은 유사한 샘플과 상이한 샘플을 구분하는 데에 대한 한계가 있다.

본 논문에서는 순차적 학습 분야에서의 두 가지 주요 기여를 제시하고자 한다. 첫째로, 순차적 추천 시스템을 위해 설계된 대조적 학습 방법을 고도화합니다. 상이한 샘플을 적응적으로 구성함으로써 아이템 임베딩의 품질을 향상시켜 추천 작업의 성능을 향상시키고자 한다. 둘째로, Fisher 병합을 적용하여 순차적 모델의 협업 기법을 소개한다. 이 접근 방식은 다중 모델의 매개변수를 병합하여 파인튜닝을 실현하며, 견고한 성능을 강조한다. 실험을 통해 제안한 방법의 효과를 입증하고, 이러한 방법이 순차적 학습 및 추천 시스템 분야의 최신 기술 발전에 기여할 수 있는 잠재력을 강조한다.

주요어휘: 대조학습, 앙상블, 순차 추천

학번: 2021-26380

감사의 글

감사의 글 내용