# Curriculum Meta-Learning for Next POI Recommendation

Yudong Chen*
cyd18@mails.tsinghua.edu.cn
Tsinghua University, Beijing, China

Xin Wang†
xin_wang@tsinghua.edu.cn
Tsinghua University, Beijing, China

Miao Fan†
fanmiao@baidu.com
Baidu Inc., Beijing, China

Jizhou Huang
huangjizhou01@baidu.com
Baidu Inc., Beijing, China

Shengwen Yang
yangshengwen@baidu.com
Baidu Inc., Beijing, China

Wenwu Zhu†
wwzhu@tsinghua.edu.cn
Tsinghua University, Beijing, China

## ABSTRACT

Next point-of-interest (POI) recommendation is a hot research field where a recent emerging scenario, *next POI to search recommendation*, has been deployed in many online map services such as Baidu Maps. One of the key issues in this scenario is providing satisfactory recommendation services for cold-start cities with a limited number of user-POI interactions, which requires transferring the knowledge hidden in rich data from many other cities to these cold-start cities. Existing literature either does not consider the city-transfer issue or cannot simultaneously tackle the data sparsity and pattern diversity issues among various users in multiple cities. To address these issues, we explore *city-transfer next POI to search recommendation* that transfers the knowledge from multiple cities with rich data to cold-start cities with scarce data. We propose a novel Curriculum Hardness Aware Meta-Learning (CHAML) framework, which incorporates hard sample mining and curriculum learning into a meta-learning paradigm. Concretely, the CHAML framework considers both city-level and user-level hardness to enhance the conditional sampling during meta training, and uses an easy-to-hard curriculum for the city-sampling pool to help the meta-learner converge to a better state. Extensive experiments on two real-world map search datasets from Baidu Maps demonstrate the superiority of CHAML framework.

## CCS CONCEPTS

• **Information systems** → **Social recommendation**; • **Theory of computation** → *Sketching and sampling*.

## KEYWORDS

next POI recommendation, meta-learning, curriculum learning

**ACM Reference Format:**
Yudong Chen, Xin Wang, Miao Fan, Jizhou Huang, Shengwen Yang, and Wenwu Zhu. 2021. Curriculum Meta-Learning for Next POI Recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467132

*The work is done during his internship at Baidu Inc.
†Corresponding authors.


*KDD '21, August 14–18, 2021, Virtual Event, Singapore.*

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467132

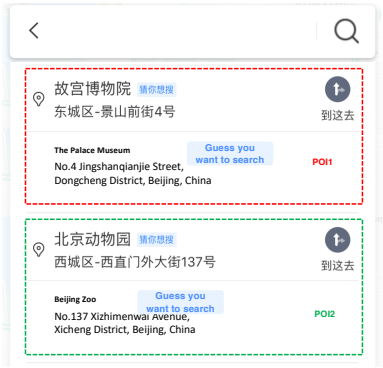
## 1 INTRODUCTION

With the prominence of location-based social networks, next point-of-interest (POI) recommendation [15] has attracted remarkable attention in the past decade. Given the user's check-in history and the current spatio-temporal contexts, the task focuses on recommending the next POI a user may want to go. Recently, an emerging scenario in this field, *next POI to search recommendation*, has been deployed as an important feature in many online map services such as Baidu Maps. Taking Baidu Maps, one of the largest online map applications in the world with over 340 million monthly active users worldwide by the end of December 2016[1] as an example, this new scenario aims to recommend a potential POI that a user would like to search without actually typing any queries. Fig 1(a) and 1(b) show an example of the next POI to search recommendation results to a user who opens up Baidu Maps before typing any input to search a POI. As illustrated in this example, the bubble in Fig 1(a) on the home page and the drop-down boxes in Fig 1(b) (when a user clicks the search input box) provide a user-friendly function that suggests one or a list of POIs a user may want to search according to her search history, location, and current time. If the Palace Museum, the place she plans to search at this moment, is exactly in the bubble or the drop-down boxes, then she will finish the search without even typing any queries. Therefore, next POI to search recommendation plays an essential role in enhancing user experience in both search efficiency and new interest discovery, capable of improving user retention as well as attracting new users.

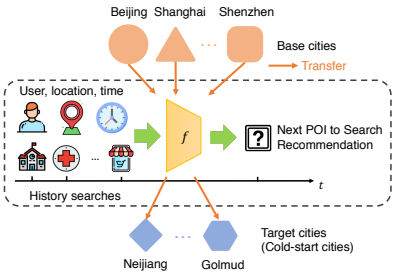
As a functional module in online map services, next POI to search recommendation is supposed to render satisfactory service to users living in every city. However, the available data is usually scarce in some cities with small urban size, population, or market occupancy, i.e., the *cold-start cities* where a limited number of user-POI interactions can be ever collected, which results in the failure of directly applying most state-of-the-art next POI recommendation models to the data of these cities. Therefore, it is necessary to transfer the knowledge learned from other *base cities* with rich data to the recommendation tasks for the cold-start cities. We define this problem as *city-transfer next POI to search recommendation*, which is illustrated in Fig 1(c). Specifically, this problem aims to learn a next POI to search recommender $f$ from multiple base cities

[1] http://ir.baidu.com/static-files/e249a0f8-082a-4f8a-b60d-7417fa2f8e7e


(a) The Bubble (the red dotted box) on the home page recommends a POI the user might want to search.




(b) When the user clicks the search input box, the Drop-down Boxes recommend a list of POIs.




(c) The illustration of the city-transfer next POI to search recommendation problem.

**Figure 1: (a) and (b) are two screenshots of the recommended point of interests (POIs) by the next POI to search recommendation module at different pages of Baidu Maps app. The module help the user search efficiently and discover new interests. (c) illustrates the main problem we explore in this paper (city-transfer next POI to search recommendation): how to leverage the sufficient data in multiple base cities to help the next POI to search recommendation in multiple cold-start cities.**

(e.g., Beijing), and transfer the knowledge to multiple target (cold-start) cities to help improve the recommendation service quality within these cities (e.g., Golumd). The key to this problem is to tailor a suitable transfer algorithm for the knowledge transfer between base and target cities, which faces two main challenges. **1) The shared data among different cities is extremely limited, making the transfer more challenging.** First of all, POIs in different cities have no intersections. Furthermore, the majority of map search records within a city are from its local residents, which heavily dwindles the intersections of users across different cities. **2) The map search patterns of various users in different cities demonstrate a large degree of diversity, bringing great difficulties for recommendation knowledge transfer.** In other words, though some certain types of the map search patterns are trivial to capture (e.g., many users repeatedly search the same POI), further improvement for recommendation quality heavily relies on precisely capturing the diverse search patterns (which are more difficult to predict) among diverse cities and users, thus posing great difficulties in developing suitable transfer algorithm.

Unfortunately, existing approaches in literature cannot tackle the above two challenges simultaneously. First of all, existing next POI recommendation methods mainly focus on how to capture the spatial-temporal relations between users and POIs by either combining statistical machine learning methods such as matrix factorization and Markov chain with geographical modeling [4, 19], or utilizing deep architectures such as graph neural networks [28, 44, 46] and recurrent neural networks [14, 18, 20, 49]. However, these algorithms rely on sufficient training data and rarely consider the city transfer problem, therefore directly applying them on cold-start-city data becomes impractical. Secondly, existing transfer based methods are also not adequate for the following reasons: a) Traditional "pretraining and fine-tuning" methods cannot solve challenge (2): they tend to suffer from pretraining on the base cities due to the high diversity, and thus fail to capture the diverse user preferences in the target cold-start cities. b) Cross-domain recommendation [3] approaches cannot solve challenge (1): they require a large number of overlapping users or items between source domain and target domain, but the shared data among base and target cities is extremely limited. c) Meta-learning [36] has achieved great success as transfer algorithms in many few-shot learning applications [6, 8, 9, 47], providing potential solutions for our problem. Nevertheless, existing meta-learning methods still cannot solve challenge (2), since they do not explicitly take the diversity among various users in different cities into consideration.

To simultaneously tackle these two challenges, we propose a novel Curriculum Hardness Aware Meta-Learning (CHAML) framework to alleviate data sparsity and sample diversity issues through incorporating meta-learning and non-uniform sampling strategies into next POI to search recommendation. To start with, we extend Model-Agnostic Meta-Learning (MAML) [8] to help the knowledge transfer to cold-start cities under the condition of limited shared data. With the next POI to search recommendation in each city regarded as a single task, the initial recommender weights are meta-learned for faster adaption on few training data in target cold-start cities. In addition, we design two components on the sampling strategy in the meta-training process, i.e., hardness aware meta-learning and city-level sampling curriculum, to explicitly consider the sample diversity issue. Firstly, we exploit the idea of hard sample mining [33] to enhance the meta-learner by simultaneously considering user- and city-level hardness during conditional sampling, forcing the model to learn more from combinations of more difficult examples, which are supposed to have highly diverse and more informative map search patterns and contribute more to the generalization boost. Concretely, we divide each meta training round into two stages. In each stage, we first update the meta-learner in a MAML-like manner, and then conditionally re-sample a new batch of tasks containing the most difficult users (Stage 1) and cities (Stage 2) under a loss-based criterion. Secondly, we further propose to draw upon the curriculum learning [40] strategies to help improve the convergence rate and generalization capacity of the

meta-learner under the condition of high city-level diversity, which boosts the transfer performance when the amount of base cities is relatively large (and the diversity of these cities is thus relatively high). The basic idea is to present training tasks for the meta-learner in an *easier first, harder later* paradigm with the difficulties judged by a *teacher*, in order to help the meta-learner converge to a better state. Concretely, we pretrain a teacher recommender for each base city, take the best valid score to measure city-level difficulty, and leverage this knowledge to design a curriculum for the city sampling pool at each meta-training step. Experimental results on two real-world map search datasets from Baidu Maps validate the effectiveness of our proposed CHAML framework.

We summarize our main contributions as follows.

- To the best of our knowledge, we are the first to explore the problem of city-transfer next POI to search recommendation and exploit a meta-learning paradigm for this problem.
- A novel Curriculum Hardness Aware Meta-Learning (CHAML) framework is proposed to alleviate data scarcity and sample diversity issues for the recommendation in cold-start cities through enhancing the meta-learner with user- and city-level hard sample mining and city-level curriculum learning.
- We conduct extensive experiments on two real-world map search datasets from Baidu Maps, and the proposed CHAML framework shows superior performance against several state-of-the-art (SOTA) baselines.

The CHAML framework has been applied for the A/B test in Baidu Maps. The beneficial audience may include the researchers and engineers with interest on POI recommendation.

## 2 PRELIMINARIES AND DEFINITION

In this section, we define the problem of city-transfer next POI to search recommendation and the meta-learning setup for this problem. We assume each city has its unique user set $\mathcal{U}$ and POI set $\mathcal{V}$ without sharing any common users or POIs. $r = (v, t, l)$ is a search record of user $u$, where $v$ is POI ID, $t$ is the timestamp, and $l$ is her GPS location. Her user history $hist_u = \{r_1, r_2, \cdots, r_n\}$ is a sequence of records. Given the user history of $u$, the current spatial-temporal context, and a pool of POI candidates, next POI to search recommendation aims to learn the mapping $f : (u, hist_u, t_{n+1}, l_{n+1}, v_{candi}) \mapsto y \in [0, 1]$, where $y$ is the probability that she does search the POI.

The main problem of this paper is defined as follows.

**City-Transfer Next POI to Search Recommendation** Suppose we have a set of *base cities* $C_{\mathcal{B}} = \{c^{(i)}_{base}\}$ and a set of *target cities* $C_{\mathcal{T}} = \{c^{(j)}_{target}\}$ with a limited amount of map search data. The goal is to transfer the knowledge from base cities' data to improve recommendation performance in target cities.

In a meta-learning setup, recommendation within each city $c$ is regarded as a single task (with its own dataset $\mathcal{D}$). The map search data of $C_{\mathcal{B}}$ and $C_{\mathcal{T}}$ are denoted as training datasets $\mathbb{D}_{train} = \{\mathcal{D}^{(i)}_{base}\}$ and testing datasets $\mathbb{D}_{test} = \{\mathcal{D}^{(j)}_{target}\}$. Suppose a user $u$'s search records are $\{r_1, r_2, \cdots, r_n\}$ and $m$ is the predefined minimum length of user history in a sample, then $n - m$ positive samples $\{(\mathbf{x}_i, y_i)\}$ are generated recurrently:

$$\begin{aligned} \mathbf{x}_i &= (u, hist^i_u, r_i), \quad y_i = 1 \\ hist^i_u &= (r_1, r_2, ..., r_{i-1}), \qquad i = m+1, \cdots, n. \end{aligned} \tag{1}$$

The candidate POIs in negative samples are randomly sampled from the POI set of city $c$ and not appearing in $hist^i_u$.

Moreover, the dataset of each meta-learning task has a support set $\mathcal{D}^{spt}$ for training and a query set $\mathcal{D}^{qry}$ for testing. We chronologically select the first $k$ samples of each user to put into $\mathcal{D}^{spt}$ and the rest into $\mathcal{D}^{qry}$. Finally, our goal is to leverage training datasets ( $\mathbb{D}_{train}$ of base cities $C_{\mathcal{B}}$ ) to learn a meta-learner $F$ such that, given the $\mathcal{D}^{spt}$ of a testing dataset (of target cities $C_{\mathcal{T}}$), $F$ predicts the parameters $\theta$ of recommender $f$ to minimize the recommendation loss $\mathcal{L}$ on the $\mathcal{D}^{qry}$. Formally:

$$\begin{aligned} \omega^* &= \arg\min_{\omega} \sum_{\mathcal{D}=[\mathcal{D}^{spt}, \mathcal{D}^{qry}] \in \mathbb{D}_{test}} \mathcal{L}(f_\theta, \mathcal{D}^{qry} \mid \mathbb{D}_{train}, \mathcal{D}^{spt}) \\ \text{s.t.} \quad \theta &= F_\omega(\mathcal{D}^{spt} \mid \mathbb{D}_{train}), \end{aligned} \tag{2}$$

where $\omega$ and $\theta$ stand for parameters of $F$ and $f$, respectively.

## 3 THE CHAML FRAMEWORK

In this section, we introduce the proposed framework, Curriculum Hardness Aware Meta-Learning (CHAML) for the city-transfer next POI to search problem. We first introduce two basic components in CHAML, i.e., the base recommender and an extension of MAML [8] to our scenario, which enables meta-learning for our problem (Section 3.1 & 3.2). Secondly, we elaborate the two sampling strategy components, i.e., hardness aware meta-learning and city-level sampling curriculum in CHAML, which explicitly consider the sample diversity issue (Section 3.3 & 3.4). The overall algorithm of CHAML is presented in Section 3.5.

### 3.1 Base Recommender

We apply the lightweight yet effective Deep Interest Network (DIN) [51] as the base recommender $f_\theta$. It is composed of 3 modules, mapping sample feature $\mathbf{x}_i = (u, hist^i_u, r_i)$ to $\hat{y}_i \in [0, 1]$, i.e., the predicted probability for $u$ searching $r_i$.

**Embedding module ($\mathbf{x}_i \mapsto (\boldsymbol{e}_{hist}, \boldsymbol{e}_{candi})$).** We simply ignore the user ID and focus on modeling the user history[2]. In this module, embedding matrices $E_{id}, E_{cate}, E_{time}$ are adopted to map POI ID $v_i$, its category, and the hourly timestamp $t_i$ into $d$-dimensional vectors, respectively, which are then concatenated to form the embedding vector $\boldsymbol{e}$ of each record, where $\boldsymbol{e}_{hist}$ is a list of embeddings for user history, and $\boldsymbol{e}_{candi}$ is the embedding for the POI candidate.

**Attention module ($(\boldsymbol{e}_{hist}, \boldsymbol{e}_{candi}) \mapsto \boldsymbol{h}$).** A typical attention module formulated as follows is adopted here to capture the preference relations between the record candidate and user history:

$$\begin{aligned} \boldsymbol{h} &= \text{attention}(\boldsymbol{e}_{candi}, \boldsymbol{e}_{hist}) \\ \text{attention}(K, V) &= \text{softmax}(MLP_{att}([K; V; K - V; K \cdot V]))V, \end{aligned} \tag{3}$$

where $MLP_{att}$ is a two-layer Multi-Layer Perceptron parameterized by $\theta_{att}$ to capture non-linear relations, $\boldsymbol{h}$ is a hidden vector representing the relations and $[;]$ stands for concatenation.

**Output module ($(\boldsymbol{h}, \boldsymbol{e}_{candi}) \mapsto \hat{y}_i$).** Finally, we concatenate $\boldsymbol{h}, \boldsymbol{e}_{candi}$ and use a 3-layer MLP with parameters $\theta_{rec}$ to predict $\hat{y}_i$:

$$\hat{y}_i = MLP_{rec}([\boldsymbol{h}; e_{candi}]). \tag{4}$$

The parameters of $f$, i.e., $\theta = \{E_{cate}, E_{time}, \theta_{att}, \theta_{rec}\}$, are later meta-learned. Note that we include $E_{cate}, E_{time}$ and exclude $E_{id}$

[2]The reason is that users of map search apps often do not have user profiles or social relations, and user IDs are not shared among cities in our setup.

since the POI categories and 24 hours are shared among different cities, while the POI IDs have no intersection. As compensation, $E_{id}$ is pretrained and then freezed (details in Appendix A.1).

## 3.2 Meta-Learning in our problem

To achieve fast adaption to multiple cold-start cities with insufficient data, we extend MAML [8] to our scenario. Intuitively, MAML learns $\omega = \theta_0$, i.e., the initialization of the base recommender $f$, which could adapt to new tasks by few update steps on few support samples, and predict well on the query samples. Each iteration of MAML includes 2 phases: local update and global update on a sampled task batch, illustrated in the upper left part of Fig 2. With the first phase updating $\theta_0$ locally on the $\mathcal{D}^{spt}$ of each task, the second phase globally updates $\theta_0$ by gradient descent to minimize the sum of losses on the $\mathcal{D}^{qry}$s of all tasks.

**Local update.** Firstly, we sample a batch of base cities $\mathcal{B} = \{c\} \subset C_{\mathcal{B}}$ where each city $c$ has its unique user set $\mathcal{U}_c$ and POI set $\mathcal{V}_c$. Then we randomly sample a group of users $u_i \sim \mathcal{U}_c \quad (|\{u_i\}| \ll |\mathcal{U}_c|)$ and form $\mathcal{D}_c^{spt}$ and $\mathcal{D}_c^{qry}$. Next, we calculate the training loss on $\mathcal{D}_c^{spt}$ and locally update $\theta$ by one step:

$$\theta'_c = \theta - \alpha \nabla_\theta \mathcal{L}_c(f_\theta, \mathcal{D}_c^{spt}), \tag{5}$$

where $\mathcal{L}$ is the cross-entropy loss of binary classification, $\alpha$ is the local learning rate, and $\theta'_c$ is the locally updated recommender parameters on each city $c \in \mathcal{B}$.

**Global update.** In the second phase, we start with calculating the testing loss on each $\mathcal{D}_c^{qry}$ with the corresponding $\theta'_c$. Then we globally update the initialization $\theta$ by one gradient step on the sum of all the testing losses:

$$\theta = \theta - \beta \nabla_\theta \sum_{c \in \mathcal{B}} \mathcal{L}_c(f_{\theta'_c}, \mathcal{D}_c^{qry}), \tag{6}$$

where $\beta$ is the global learning rate. In this way, a more transferrable initialization of $\theta$ for fast adaption to cold-start cities can be learned after adequate meta-learning iterations.

## 3.3 Hardness Aware Meta-Learning

In spite of the effectiveness of the MAML extension introduced above, we argue to further adapt the meta-learning model into our city-transfer problem, by considering the unique characteristics of map search data and the awareness of different levels of "hardness".

*3.3.1 User-POI Distance in Map Search Data.* Literature on next POI recommendation is typically designed for check-in data generated by users physically arriving at a POI and checking in. However, in the map search scenario, users may interact with POIs far from their current locations (e.g., some users often search their companies for map navigation before they drive out of their home.). We call this gap *user-POI distance*, which plays a key role in judging whether a user would probably search a POI candidate, according to the observation that the majority of search records happen locally (see Appendix A.2 in supplementary file). To this end, we enhance the base recommender by concatenating the z-score standardized user-POI distance to the embedding vector of each record:

$$\boldsymbol{e}_i \leftarrow [\boldsymbol{e}_i; \frac{d_i - d_{mean}}{d_{std}}], \tag{7}$$

where $\boldsymbol{e}_i$ stands for the original embedding vector of each user history record or candidate record (embedded by embedding module of $f$), $d_i$ is the corresponding user-POI Euclidian distance in Mercator projection, and $d_{mean}, d_{std}$ is the city-wise mean and standard deviation, respectively.

*3.3.2 Meta-Learning with Hardness Awareness.* The MAML extension has a high dimensionality of sampling space, i.e., the possible combinations of different city batches and user batches (see *local update* in Sec 3.2) could be zillion. However, during random sampling, MAML tends to overfit to the simple search patterns (e.g., repeatedly searching the same POI) of many users in many cities, while to fail on more diverse search patterns (e.g., searching various POIs without obvious rules of behavior) which are harder to predict but more important for the overall performance boost. What is more, since the testing distribution on the target cities is unknown, explicitly dealing with this diversity during training is supposed to help improve the generalization capacity of the meta-learned initial parameters. To this end, inspired by the progress in hard sample mining, we propose to online adjust the sampling strategy to adversely train the meta-learner in a hardness aware paradigm, forcing the model to learn more from the harder search patterns, which forms the key idea. Specifically, city-level and user-level hardness are simultaneously considered.

**In general, the "hardness" here is self-judged by the current performance of the model on the query samples.** Concretely, the meta-learner evaluates the city-level hardness by the accuracy on $\mathcal{D}^{qry}$ of each city at the current iteration, and the user-level hardness on the query samples of each specific user.

The pipeline of the meta training process is demonstrated in Fig 2. A complete round of meta-learning consists of 2 stages, each of which contains a full meta step of MAML (Eq.(5)(6)) and a hardness aware sampling operation, detailed as follows.

**Stage 1.** First, a batch of tasks, namely $\mathcal{T}_{hard\_city}$, is initialized or passed to the meta-learner by Stage 2. Next, we do a full meta step with $\mathcal{T}_{hard\_city}$, and sort the $B_u$ users of each city task by average accuracies $\{acc_u\}$ on user-wise query samples, keep the $k_u B_u$ ( $k_u$ is the proportion of hard users) hardest users (with lowest accuracies), and re-sample some new users in each city to form a new batch of tasks $\mathcal{T}_{hard\_user}$:

$$\begin{aligned} \{u\}_c &\sim p(\mathcal{U}_c \mid \{acc_u\}_c) \\ \mathcal{T}_{hard\_user} &\sim p(\mathcal{T} \mid \{u\}_c), \quad c \in \mathcal{B}_{hard\_city}. \end{aligned} \tag{8}$$

Sharing the same cities with $\mathcal{T}_{hard\_city}$, this batch of ***hard-user aware*** new tasks is expected to contain harder user combinations within each city and then passed to Stage 2.

**Stage 2.** In Stage 2, the meta-learner takes a further full meta step with $\mathcal{T}_{hard\_user}$, aiming at improving the meta-learner's generalization ability to harder users. Akin to Stage 1, this time we sort the cities by the accuracies $\{acc_t\}$ on each $\mathcal{D}^{qry}$, keep the $k_c B_c$ hardest cities, and re-sample some new cities to form a city batch $\mathcal{B}_{hard\_city}$ with size $B_c$. Next, we randomly sample a new batch of users for each city to form a new task batch $\mathcal{T}_{hard\_city}$, which is expected to contain a harder combination of cities than before:

$$\begin{aligned} \{c\} &\sim p(C_{\mathcal{B}} \mid \{acc_t\}) \\ \mathcal{T}_{hard\_city} &\sim p(\mathcal{T} \mid \{c\}). \end{aligned} \tag{9}$$

Finally, this ***hard-city aware*** new task batch $\mathcal{T}_{hard\_city}$ is passed to Stage 1 of next round for further meta training.

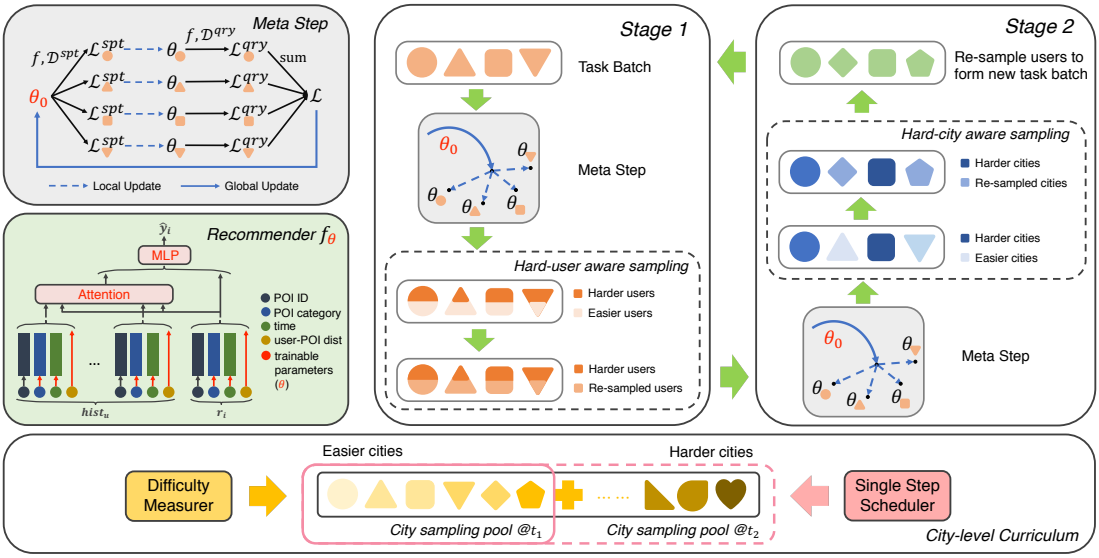



**Figure 2: The framework of Curriculum Hardness Aware Meta-Learning (CHAML): The middle and right parts stand for the pipeline of hardness aware meta training process in Section 3.3.2, the upper left part illustrates a full meta step in Section 3.2 and the middle left part demonstrates the next POI recommender (Section 3.1) enhanced by the unique user-POI distance features in map search scenario (Section 3.3.1). The bottom part shows the curriculum for city sampling pool (Section 3.4).**

It is worth mentioning that the city-level hardness is more difficult to evaluate than user-level, since each time we calculate the accuracies on *all* the query samples of each user, but on only a *small proportion* of the query samples (which belongs to the user batch of the current task) of each city. Therefore, we conservatively evaluate city-level hardness based on accuracies on $\mathcal{T}_{hard\_user}$ (at the beginning of Stage 2, which involves incompletely-randomly-sampled harder user combinations, instead of $\mathcal{T}_{hard\_city}$ (at the beginning of Stage 1, to reduce randomness and improve robustness. This strategy also brings a mutual effect between the user- and city-level hardness throughout the stage rounds. In summary, by this style of hardness aware meta training, the meta-learner gradually finds the optimal $\theta_0$ from which the recommender $f$ could better adapt to harder map search patterns, and thus achieve overall performance boost on highly diverse target cities and users.

## 3.4 City-level Sampling Curriculum

In the last section, the "hardness" is self-judged by the current meta-learner, analogous to students practicing more on the chapters they have failed on. However, if a teacher is available, we could further plan a curriculum for the student from easy chapters to harder ones for better learning, as argued in curriculum learning. Inspired by [42], we propose to invite a transfer learning teacher to pre-plan a city-level curriculum for the meta-learner, and denote this curriculum hardness aware meta-learning framework as CHAML.

**Difficulty Measurer.** To measure the task difficulty of a city, we firstly divide (by users) all the samples into a training set $\mathcal{D}^{train}$ and a valid set $\mathcal{D}^{valid}$. Then the base recommender is trained sufficiently on $\mathcal{D}^{train}$ and the best valid score with metric $\Phi$ (e.g., AUC) is taken to measure city-level difficulty $\delta$: the lower best valid score, roughly, the harder map search patterns in a city. Formally:

$$\delta_c \propto -\max_{\theta} \Phi(f_\theta, \mathcal{D}_c^{valid} \mid \mathcal{D}_c^{train}). \tag{10}$$

**Algorithm 1** Curriculum Hardness Aware Meta-Learning

**Input:** $C_{\mathcal{B}}, \mathbb{D}_{train}, N$: $N$ base cities and all the samples; $f$: base recommender; $\alpha, \beta$: learning rates; $B_u, B_c$: size of user/city batches; $k_u, k_c$: proportions of hard users/cities; $M$: max step of iterations; $\eta$: the number of iterations in each scheduler step; $\xi$: the fraction of cities in the initial scheduler step

**Output:** $\theta$: optimal initial weights of $f$

Randomly initialize $\theta$.
Calculate the difficulty of each city by Eq.(10).
Randomly sample a batch of cities $\mathcal{B}_{init} \sim C_{\mathcal{B}}$.
Randomly sample a batch of users $\{u\} \subset \mathcal{U}_c$ from each city $c \in \mathcal{B}_{init}$ to form $\mathcal{T}_{init}$, $\mathcal{T}_{hard\_city} = \mathcal{T}_{init}$.
**repeat**
  Update the city sampling pool as Eq.(11).
  **[Stage 1]**
  Do a meta step Eq.(5)(6) on $\mathcal{T}_{hard\_city}$;
  Calculate user-level accuracies $\{acc_u\}$ on query samples of each user $u$;
  Select the $k_u B_u$ hardest users by $\{acc_u\}$ and re-sample new users in each city to form $\mathcal{T}_{hard\_user}$ as Eq.(8).
  **[Stage 2]**
  Do a meta step Eq.(5)(6) on $\mathcal{T}_{hard\_user}$;
  Calculate task-level accuracies $\{acc_t\}$ on $\mathcal{D}^{qry}$ of each task;
  Select the $k_c B_c$ hardest cities by $\{acc_t\}$ and re-sample new cities to form $\mathcal{B}_{hard\_city}$ as Eq.(9);
  Randomly sample a batch of users $\{u\} \subset \mathcal{U}_c$ from each city $c \in \mathcal{B}_{hard\_city}$ to form $\mathcal{T}_{hard\_city}$.
**until** Max step of iterations $M$

**Single Step Scheduler for City Pool.** With all base cities $C_{\mathcal{B}} = \{c_1, c_2, \cdots, c_N\}$ sorted by difficulties and the max step of meta iterations $M$, we define a stair-case *scheduler function* $g : [M] \mapsto [N]$ to determine a sequence of city pools $C'_1, \cdots, C'_M \subseteq C_{\mathcal{B}}$, of

**Table 1: Dataset Statistics.**

| Dataset | #Base cities/users | #Target cities/users | #POIs |
|---|---|---|---|
| *MapSmall* | 8 / 160,000 | 6 / 3,000 | 477,218 |
| *MapLarge* | 72/1,090,070 | 43/19,672 | 2,294,879 |

size $|C_i'| = g(i)$, from which we sample the city batches during training. We follow [10] to adopt the single step scheduler:

$$g(i) = \xi^{\mathbb{1}[i<\eta]} \cdot N, \tag{11}$$

where $\xi$ is the fraction of cities in the initial step, $\eta$ is the number of iterations in each step, and $\mathbb{1}$ is indicator function.

Theoretically, the above curriculum enables the model to have a greater probability of taking easier gradient steps in the direction of more favorable optima, which is precisely concluded as Proposition 1. The proof is in Appendix A.3. Note that the easy-to-hard curriculum does not conflict with the hardness aware training mechanism, since practicing more on mistakes in easy chapters is also beneficial.

**Proposition 1** *Under reasonable and mild conditions, the proposed curriculum above effectively changes the optimization landscape of the meta-learner, making the gradient direction overall steeper (which increases the likelihood of escaping local minimums).*

## 3.5 The Algorithm of CHAML

Algorithm 1 summarizes the meta training process of CHAML. Note that $\mathcal{B}$ and $\mathcal{T}$ stand for a batch of base cities and a task batch, where we sample a batch of users from each city in $\mathcal{B}$ to form a task in $\mathcal{T}$.

# 4 EXPERIMENTS

In this section we conduct the experiments and ablation study on two real-world POI search datasets from Baidu Maps to evaluate the performance of the proposed CHAML on warming up recommendation in cold-start cities. The code is released at https://github.com/PaddlePaddle/Research/tree/master/ST_DM/KDD2021-CHAML.

## 4.1 Experimental Setup

*4.1.1 Datasets.* Two datasets, namely *MapSmall* and *MapLarge*, are gathered from the large-scale and real-world search records of Baidu Maps[3] from October to December 2019. The two datasets differ in scale (i.e., city amount) and city tier distribution: *MapSmall* is composed of 14 first- and second-tier cities of China, where we take 8 cities as base cities, 4 second-tier cities as target cities, and the other 2 for validation. We randomly select 20,000 users from each base city and 500 users from each of the other cities to simulate the cold-start scenario. Ten recent records are collected from each user of all cities to generate 5 samples with the predefined minimum history length $m = 5$. In *MapLarge* dataset, we randomly collect 115 cities from all the 338 cities in China, including every tier of cities. Similarly, we take 72, 17, 26 cities as base, valid, target cities, respectively, where the valid and target cities are fifth-tier cities with less than 5,000 users. Thus we randomly select $\min(20,000, |\mathcal{U}_c|)$ and $\min(500, |\mathcal{U}_c|)$ users for base cities and the other cities, respectively. The dataset statistics are in Table 1.

[3]The datasets are not publicly available due to privacy policies.

*4.1.2 Evaluation Metrics.* We adopt two widely-used ranking evaluation metrics: *Hit Ratio at K (HR@K)* and *Normalized Discounted Cumulative Gain at K (NDCG@K)*. Given 1 positive POI and 100 negative POI candidates (sampled from $\mathcal{V}_c$) in the query set of each target city $c$, the model outputs a ranked list. Then *HR@K* measures whether the positive POI shows within the top $K$ in the ranked list, and *NDCG@K* takes into account the position of the positive POI (if it is) in the top $K$ list, penalizing the score when the rank is lower. The metrics are calculated on the query set of target cities, and the final scores are the average metrics weighted by the number of query samples in each city.[4]

*4.1.3 Baselines.* A framework for the city-transfer problem has two aspects: **recommender model** and **how to transfer**, and the contribution of this paper is on the latter. We aim to verify whether incorporating a lightweight recommender (i.e., DIN in Section 3.1) with our proposed transfer algorithm CHAML is adequate enough to outperform the state-of-the-art (SOTA) POI recommenders with traditional transfer methods.

From the view of recommender model, we consider the following deep learning models from the most related academic settings related to our *next POI to search recommendation* problem.

- **NeuMF** [11]: An item recommendation model combining MF with MLP, taken as a proving-correctness baseline.
- **HGN** [23]: A sequential item recommender with feature- and instance-level gating modules for feature selection and an item-item product module for relation modeling.
- **ATST-LSTM** [14]: A recent next POI recommender, attending user embedding on the LSTM outputs with distance and delta time between successive check-ins considered.
- **PLSPL** [43]: A state-of-the-art (SOTA) model for next POI recommendation, learning long-term user preference by attention and short-term preference by two LSTMs.
- **iMTL** [49]: A SOTA model for next POI recommendation, using two-channel encoder and a task-specific decoder for capturing the sequential correlations of activities and location preferences.
- **DIN** [51]: A popular recommender which attends item candidates on the users' historical items. This is also the *base recommender* in Section 3.1.

From the perspective of transfer methods, the following approaches are adopted in our experiments:

- **No transfer (None)**: We do not use any data of base cities and train the above models only on the support samples of each target city, respectively.
- **Pretrain and Fine-Tune (FT)**: We pretrain the above models on all the data of base cities, save the best weights and then fine-tune on the support set of each target city.
- **MAML** [8]: The extension of MAML in Section 3.2.
- **s$^2$Meta** [6]: A SOTA meta-learning algorithm for item recommendation in cold-start scenarios, which meta-learns the weight initialization, update strategy, and early-stop policy. For fair comparison, we adopt DIN (in Section 3.1) as the base recommender in $s^2Meta$.
- **HAML**: The variant of CHAML in Section 3.3.

[4]As introduced in Section 4.1.1, in some target cities $|\mathcal{U}_c| < 500$.

**Table 2: Results on the two datasets. H = Hit Ratio and N = NDCG. The best scores are in bold and second-best underlined.**

| Transfer | Models | *MapSmall* | | | | *MapLarge* | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **H@5** | **H@10** | **N@5** | **N@10** | **H@5** | **H@10** | **N@5** | **N@10** |
| None | NeuMF | 0.1002 | 0.1628 | 0.0646 | 0.0847 | 0.1401 | 0.2125 | 0.0966 | 0.1199 |
| | HGN | 0.0630 | 0.1245 | 0.0392 | 0.0589 | 0.1188 | 0.1941 | 0.0791 | 0.1033 |
| | ATST-LSTM | 0.2705 | 0.3410 | 0.2177 | 0.2403 | 0.2809 | 0.3616 | 0.2183 | 0.2441 |
| | PLSPL | 0.1210 | 0.1855 | 0.0844 | 0.1051 | 0.1824 | 0.2592 | 0.1327 | 0.1573 |
| | iMTL | 0.1208 | 0.2068 | 0.0779 | 0.1054 | 0.1485 | 0.2323 | 0.1009 | 0.1277 |
| | DIN | 0.0787 | 0.1530 | 0.0466 | 0.0704 | 0.0991 | 0.1649 | 0.0646 | 0.0856 |
| FT | NeuMF | 0.1402 | 0.2063 | 0.0982 | 0.1192 | 0.1656 | 0.2442 | 0.1166 | 0.1418 |
| | HGN | 0.2867 | 0.4013 | 0.1956 | 0.2325 | 0.3438 | 0.4689 | 0.2459 | 0.2862 |
| | ATST-LSTM | 0.3218 | 0.4127 | 0.2497 | 0.2788 | 0.3332 | 0.4391 | 0.2513 | 0.2854 |
| | PLSPL | 0.2878 | 0.3952 | 0.2060 | 0.2407 | 0.3700 | 0.4791 | 0.2764 | 0.3117 |
| | iMTL | 0.2820 | 0.3698 | 0.2256 | 0.2538 | 0.4108 | 0.5215 | 0.3254 | 0.3610 |
| | DIN | 0.2985 | 0.4272 | 0.1989 | 0.2405 | 0.3232 | 0.4560 | 0.2246 | 0.2673 |
| Meta | MAML | 0.3478 | 0.4503 | 0.2402 | 0.2736 | 0.4007 | 0.4790 | 0.3252 | 0.3505 |
| | $s^2Meta$ | 0.3395 | 0.4593 | 0.2322 | 0.2709 | 0.4145 | 0.5233 | 0.3114 | 0.3466 |
| | HAML (Ours) | **0.4152** | **0.5632** | **0.2806** | **0.3286** | 0.4465 | 0.5747 | 0.3238 | 0.3653 |
| | CHAML (Ours) | 0.4008 | 0.5543 | 0.2695 | 0.3191 | **0.4571** | **0.5872** | **0.3320** | **0.3742** |

- **CHAML**: The proposed framework in Section 3.4.

**Hyperparameters.** In all the None and FT methods, we set the sample batch size as 1024 and learning rate as 0.001 for the superior performance[5]. Accordingly, for all meta-learning methods[6], we set learning rates $\alpha = \beta = 0.001$ and user batch size $B_u = 256$ (1024 support samples per task) to ensure fair comparison. The task batch size $B_c$ is set as 4 and 16 for *MapSmall* and *MapLarge*, respectively. We set $k_c = k_u = 0.5$, i.e., half of the hardest cities/users are kept in the hardness aware sampling strategies. We set $\xi = 0.5$ and $\eta = 0.5M$ to present the easiest half of city pool in the first half of meta training iterations. Other implementation details can be found in Appendix A.1.

## 4.2 Experimental Results

The experimental results on the two datasets are reported in Table 2, and we have the following key observations:

1) **The superiority of CHAML**. Encouragingly, as presented in both datasets, the proposed model CHAML and its variant HAML observably outperform all the other baselines, including the transfer learning (FT) results with SOTA recommenders and the SOTA meta-learning algorithms (MAML and $s^2Meta$). Specifically, the relative improvement ratio (of the average of all four metrics) over the best baseline is 21.1% and 8.1% on *MapSmall* and *MapLarge*, respectively, which demonstrates the benefits of leveraging the city-level and user-level hardness-aware sampling strategy when training the meta-learner. The advantage of taking a pre-planned curriculum is also verified in *MapLarge* when the city sampling space is dramatically large.

2) **Transfer methods: Meta > FT > None**. In both datasets, training a deep recommender only on the support set of a cold-start city impairs the performance severely, mainly due to overfitting. The pretraining and fine-tuning (FT) strategy obviously helps all the deep recommenders to improve by a large gap. However, meta-learning strategies bring further enhancement, owing to the specialized design for fast adaption to limited data, where our proposed meta-learning strategies achieve the best performance.

3) **HAML v.s. CHAML.** The reason for CHAML having lower performance than HAML on *MapSmall* dataset might be that we only have 8 base cities in total, which lessens the necessity for pre-plan a curriculum for HAML. Particularly, in CHAML, 4 "easiest" base cities are presented at first and later all 8 cities, while the task batch size $B_c$ is exactly 4, so for a long period, the model is only meta-trained on the original batch of base cities, which impairs the performance. However, in *MapLarge* dataset, the curriculum reduces the number of combinations of task batches from $C_{72}^{16}$ to $C_{36}^{16}$ (task batch size $B_c = 16$) by more than 500,000 times for the first-half iterations, which helps the meta-learner take gradient steps in low variance to more probably escape suboptimal parameters.

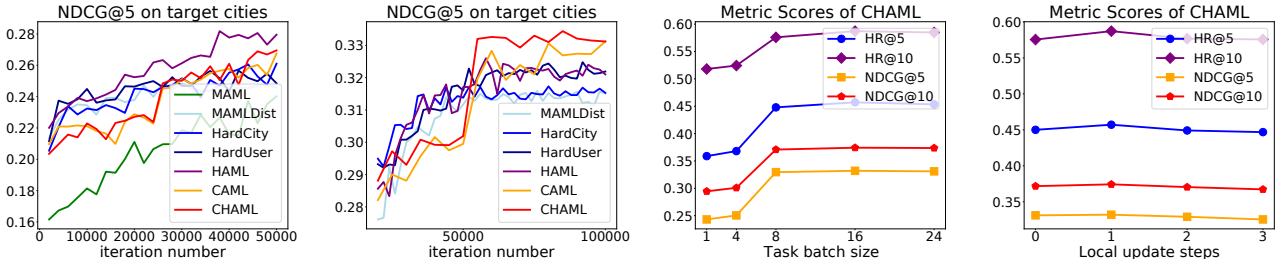
## 4.3 Ablation Study

**Impact of Different Parts.** We analyze the contributions of different components in CHAML by comparing following variants:

- *MAMLDist:* The further adapted version of MAML in Section 4.3.1 which takes into account the user-POI distance in the map search recommendation scenario.
- *HardCity:* A variant of HAML which only introduces the hard-city aware sampling into the training process of HAML, without considering hard users.
- *HardUser:* A variant of HAML, on the other hand, introducing only hard-user aware sampling into the training process of HAML, without considering hard cities.
- *CAML:* A variant of CHAML in which no hard sampling is adopted, and we only retain the curriculum.
- *HAML, CHAML:* The complete versions of the approaches discussed in Section 3.3 and 3.4.

The NDCG@5 results on *MapSmall* are shown in Fig 3(a). We can see that each part of the designs in CHAML, i.e., unique characteristics, hard-city aware and hard-user aware sampling strategies

[5]Except iMTL, for which we use the original learning rate of 0.0001.
[6]Except $s^2Meta$, for which we use the hyperparameters in the released codes.




(a) NDCG@5 on *MapSmall* (b) NDCG@5 on *MapLarge* (c) Sensitivity on task batch size $B_c$ (d) Sensitivity on local-update steps

**Figure 3: (a)(b) are performance comparison for variants of our CHAML model during the whole meta-training process on *MapSmall* and *MapLarge* datasets. (c)(d) show the parameter sensitivity analysis of CHAML on *MapLarge* dataset.**

contributes to the improvement from basic MAML, while the complete version, HAML, achieves the best performance.

On *MapLarge* dataset, a different pattern can be observed as in Fig 3(b). We find that the main contribution of CHAML comes from the hard-user aware sampling, and hard-city brings little improvement. One possible reason is that in *MapLarge*, there is a large gap between the city tier of base cities and target cities, and thus focusing more on harder base cities might bring limited enhancement for the harder ones of the much smaller target cities. Besides, the power of curriculum is highlighted in *MapLarge* with zillion-scale city-batch combinations. A clear leap can be observed in the charts of CAML and CHAML, owing to the single step scheduler in the curriculum. Comparing CHAML with CAML, we find that the hardness aware strategy slightly brings an improvement.

**Parameter Sensitivity Analysis.** Fig 3(c) shows the impact of task batch size $B_c$ when meta-training CHAML on *MapLarge* with other hyperparameters unchanged. We find that increasing the number of sampled cities (i.e., $B_c$) in each full meta step helps to boost the performance only when $B_c$ is relatively small. The improvement verifies our hypothesis that optimizing the initialization $\theta_0$ to be optimal for multiple combinations of base cities help enhance the generalization capacity of the recommender $f$. Fig 3(d) shows the model sensitivity on the number of local-update steps in Eq.5. We empirically find slight difference by updating more than one step, so we finally choose to update only one step for efficiency. Interestingly, the results without any local update steps are comparable to the best results, which might be owing to the feature reuse [30] characteristics of MAML-like algorithms.

# 5 RELATED WORK

## 5.1 Next POI Recommendation

POI recommendation [1, 15] refers to *general* $((user, POI) \mapsto score)$ and *next* POI recommendation $((user, POI, history) \mapsto score)$. General POI recommendation techniques include Matrix Factorization (MF) based [19, 21, 38], incorporating spatial-temporal contexts into vanilla MF, and Graph Embedding (GE) based [28, 44, 46], constructing different relation graphs to generate embeddings for downstream learners. The techniques for next POI recommendation include Markov Chain based [4, 48] and Neural Network based. The latter, including the models derived from RNNs [14, 20], LSTMs [18, 49], Memory Network [52], and GANs [50], has gained wide attention due to the superior performance. However, these models require large-scale training data which are not available in cold-start cities in our scenario. We explore the city-transfer problem and use meta-learning to overcome this defect.

To the best of our knowledge, there has been very few works focusing on the task of next POI *to search* recommendation, and *next POI recommendation* that conventionally refers to POIs *to check-in* is most related to our setting. Note that our task belongs to recommender system instead of search engine (e.g., auto-completion [13, 17]), since there exist no user queries as the input to the system. Besides, "out-of-town recommendation" [45] that recommends POIs to users who travel to a new city does not really relate to our problem.

## 5.2 Transfer Learning for Recommendation

Transfer learning mainly focuses on bridging different transfer knowledge from source domain to target domain to tackle the data sparsity issue. In the recommendation paradigm [29], cross domain recommender systems [3] are proposed to combine the data from different sources by either leveraging the neighborhood information of common users/items [7, 41] or learning a mapping function for latent vector projection through different domains [12, 24].

However, all these existing methods require large amounts of overlapping users/items between domains, which is not satisfied under our scenario. Taking a step back from cross domain recommendation, we design a transfer learning baseline, i.e., pretraining and fine-tuning, for contrast experiment.

## 5.3 Meta-Learning and Its Application in Recommendation

Inspired by human learning from previous related tasks to quickly learn new skills, meta-learning [36], or *learning to learn*, aiming to transfer the experience learned from multiple tasks to efficiently complete variant new tasks, has achieved great success mainly in few-shot learning applications, including few-shot image classification [8, 37], reinforcement learning (RL) [39], machine translation [9] and spatial-temporal prediction [47], etc.

The methodologies of meta-learning can be divided into 3 groups: 1) *Metric* based methods [34, 37] learn a similarity space for nearest-neighbor-based classification. 2) *Memory* based methods [25, 26, 32] store the experience of old tasks to combine with new task information for inference. 3) *Optimization* based methods [8, 31] aim at learning the optimization algorithm to help models converge to optima with limited examples.

Recently, meta-learning methods are also adopted in recommender systems to alleviate the issues of cold-start users [5, 22], items [16, 27] and scenarios [6]. However, the methods for cold-start users/items adopt a different setting from ours, making them not applicable to our scenario, and the $s^2Meta$ algorithm in [6] does not consider sequential modeling and example hardness, resulting in its inferior in capturing the diverse patterns.

### 5.4 Curriculum Learning & Hard Sample Mining

Curriculum learning (CL) [2, 40] is a training strategy that trains models initially on easy data and then on harder data, imitating how human students are taught with curricula. It could help gradient-based models to escape local minimum to some extent in lower-variance gradient directions [42]. The key components of CL include a *difficulty measurer* to decide which data is easy and a *training scheduler* to determine when to add more harder data for training. In this paper, we firstly design a curriculum for a meta-learner by leveraging a transfer-learning-based city-level difficulty measurer and a single-step training scheduler.

Hard sample mining (HSM) [33] takes a different training strategy by always focusing on the harder data, aiming at mining more informative data for training. Theoretically, under different settings, both CL and HSM can be effective [10]. In [35], the authors firstly adopt an online hard-task sampling strategy by taking the hardest classes of each classification task. Differently, we propose to sample not only harder recommendation tasks based on city-level performance, but also harder users in an online manner.

## 6 CONCLUSION

We explore the problem of cross-city next POI to search recommendation and propose a novel Curriculum Hardness Aware Meta-Learning (CHAML) framework, which considers city- and user-level hardness in meta training and a city-level curriculum. Extensive experiments on two real-world datasets with different scales from Baidu Maps show the effectiveness of CHAML.

## ACKNOWLEDGMENTS

This research is supported by the National Key Research and Development Program of China No.2020AAA0106300 and National Natural Science Foundation of China No.62050110.

## REFERENCES

[1] J. Bao *et al.,* 2015. Recommendations in location-based social networks: a survey. In *GeoInformatica* 19, 3(2015), 525–565.
[2] Y. Bengio *et al.,* 2009. Curriculum learning. In *ICML*, 41–48.
[3] I. Cantador *et al.,* 2015. Cross-domain recommender systems. In *Recommender systems handbook*, 919–959.
[4] C. Cheng *et al.,* 2013. Where youlike to go next: Successive point-of-interest recommendation. In *IJCAI.*
[5] M. Dong *et al.,* 2020. MAMO:Memory-Augmented Meta-Optimization for Cold-start Recommendation. In *SIGKDD*, 688–697.
[6] Z. Du *et al.,* 2019. Sequential Scenario-Specific Meta Learner for Online Recommendation. In *SIGKDD*, 2895-2904.
[7] A. Farseev *et al.,* 2017. Cross-domain recommendation via clustering on multi-layer graphs. In *SIGIR*, 195–204.
[8] C. Finn *et al.,* 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 1126–1135.
[9] J. Gu *et al.,* 2018. Meta-learning for low-resource neural machine translation. *arXiv preprint.*
[10] G. Hacohen *et al.,* 2019. On the power of curriculum learningin training deep networks. In *ICML.*
[11] X. He *et al.,* 2017. Neural collaborative filtering. In *WWW*, 173–182.
[12] G. Hu *et al.,* 2018. Conet: Collaborative crossnetworks for cross-domain recommendation. In *CIKM*, 667–676.
[13] J. Huang *et al.,* 2020. Personalized Prefix Embedding for POI Auto-Completion in the Search Engine of Baidu Maps. In *SIGKDD*, 2677–2685.
[14] L. Huang *et al.,* 2019. An Attention-basedSpatiotemporal LSTM Network for Next POI Recommendation. *IEEE Transactionson Services Computing.*
[15] M. Islam *et al.,* 2020. A Survey on Deep Learning Based Point-Of-Interest (POI) Recommendations. *arXiv preprint.*
[16] R. Li *et al.,* 2020. Few-Shot Learning for NewUser Recommendation in Location-based Social Networks. In *WWW*, 2472–2478.
[17] Y. Li *et al.,* 2020. Personalized Query Auto-Completion for Large-Scale POI Search at Baidu Maps. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*
[18] D. Lian *et al.,* 2020. Geography-Aware Sequential Location Recommendation. In *SIGKDD*, 2009–2019.
[19] D. Lian *et al.,* 2014. GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *SIGKDD*, 831–840.
[20] Q. Liu *et al.,* 2016. Predicting the nextlocation: A recurrent model with spatial and temporal contexts. In *AAAI.*
[21] Y. Liu *et al.,* 2014. Exploiting geographicalneighborhood characteristics for location recommendation. In *CIKM*, 739–748.
[22] Y. Lu *et al.,* 2020. Meta-learning on heterogeneousinformation networks for cold-start recommendation. In *SIGKDD*, 1563–1573.
[23] C. Ma *et al.,* 2019. Hierarchical gating networks forsequential recommendation. In *SIGKDD*, 825–833.
[24] T. Man *et al.,* 2017. Cross-DomainRecommendation: An Embedding and Mapping Approach. In *IJCAI*, 2464–2470.
[25] N. Mishra *et al.,* 2018. A simple neural attentive meta-learner. In *ICLR.*
[26] T. Munkhdalai *et al.,* 2017. Meta networks. In *ICML*, 2554–2563.
[27] F. Pan *et al.,* 2019. Warm Up Cold-start Advertisements: Improving CTR Predictions via Learning to Learn ID Embeddings. In *SIGIR*, 695–704.
[28] T. Qian *et al.,* 2019. Spatiotemporal representation learning for translation-based POI recommendation. In *TOIS* 37, 2 (2019), 1–24.
[29] W. Pan, 2016. A survey of transfer learning for collaborative recommendationwith auxiliary data. In *Neurocomputing* 177 (2016), 447–453.
[30] A. Raghu *et al.,* 2019. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint.*
[31] S. Ravi *et al.,* 2017. Optimization as a model for few-shot learning. In *ICLR.*
[32] A. Santoro *et al.,* 2016. Meta-learning with memory-augmented neural networks. In *ICML*, 1842–1850.
[33] A. Shrivastava *et al.,* 2016. Training region-based object detectors with online hard example mining. In *CVPR*, 761–769.
[34] J. Snell *et al.,* 2017. Prototypical networks for few-shot learning. In *NeurIPS*, 4077–4087.
[35] Q. Sun *et al.,* 2019. Meta-transfer learning for few-shot learning. In *CVPR*, 403–412.
[36] J. Vanschoren *et al.,* 2018. Meta-learning: A survey. *arXiv preprint.*
[37] O. Vinyals *et al.,* 2016. Matching networks for one shot learning. In *NeurIPS*, 3630–3638.
[38] H. Wang *et al.,* 2018. Exploiting POI-Specific Geographical Influence for Point-of-Interest Recommendation. In *IJCAI*, 3877–3883.
[39] J. Wang *et al.,* 2016. Learning to reinforcement learn. *arXiv preprint.*
[40] X. Wang *et al.,* 2021. A Survey on Curriculum Learning. *IEEE TPAMI.*
[41] X. Wang *et al.,* 2018. Cross-domain recommendation for cold-start users via neighborhood based feature mapping. In *DASFAA*, 158–165.
[42] D. Weinshall *et al.,* 2018. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *ICML*, 5238–5246.
[43] Y. Wu *et al.,* 2020. Personalized long- and short-term preference learning for next poi recommendation. In *TKDE* (2020).
[44] M. Xie *et al.,* 2016. Learning graph-based poi embedding for location-based recommendation. In *CIKM*, 15–24.
[45] H. Xin *et al.,* 2021. Out-of-Town Recommendation with Travel Intention Modeling. *arXiv preprint.*
[46] X. Xiong *et al.,* 2020. Dynamic discovery of favorite locations in spatio-temporal social networks. In *Information Processing & Management* 57, 6 (2020), 102337.
[47] H. Yao *et al.,* 2019. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *WWW*, 2181–2191.
[48] J. Zhang *et al.,* 2014. Lore: Exploiting sequential influence for location recommendations. In *SIGSPATIAL*, 103–112.
[49] L. Zhang *et al.,* 2020. An Interactive Multi-Task Learning Framework for Next POI Recommendation with Uncertain Check-ins. In *IJCAI*, 3551–3557.
[50] F. Zhou *et al.,* 2019. Adversarial point-of-interest recommendation. In *WWW*, 3462–3468.
[51] G. Zhou *et al.,* 2018. Deep interest network for click-through rate prediction. In *SIGKDD*, 1059–1068.
[52] X. Zhou *et al.,* 2019. Topic-enhanced memory networks for personalised point-of-interest recommendation. In *SIGKDD*, 3018–3028.

# A APPENDIX

In this section, we detail the experimental settings of this paper for reproducibility, with some supplementary figures and discussions and a proof for Proposition 1.

## A.1 Experimental Details

*A.1.1 POI ID Embedding.* As aforementioned, following [6], we pretrain and then freeze the POI ID embedding matrices $E_{id} \in \mathbb{R}^{|\mathcal{V}_c| \times d}$, since each city $c$ has its unique POI set $\mathcal{V}_c$ and thus the POI ID embeddings could not be transferred among cities. From this view, we pretrain $E_{id}$ on the search data within each city in September, a month before the dataset span. Specifically, the NeuMF model, with both user ID and POI ID embedding matrices, is taken for the pretraining. $E_{id}$ is shared among different recommenders.

*A.1.2 Parameter Settings for Recommenders.* In the base recommender $f$, the dimension $d$ of each embedding vector is set as 50. In our scenario, we have 229 POI categories in total, so the shape of $E_{cate}$ is $(229, 50)$. The MLPs in the attention module and the output module have the shapes of $(600 \times 50, 50 \times 1)$ and $(300 \times 300, 300 \times 300, 300 \times 2)$, respectively. The total number of trainable parameters of the base recommender is 224,003, which is more lightweight than most SOTA baselines and thus more suitable for optimization-based meta training algorithms.

As for the baseline recommenders, we strictly follow the original hyper-parameters in [14, 43] and use the officially-released codes of [23, 51], and we carefully adapt the iMTL model [49] to our setting to ensure fair comparison.

*A.1.3 Settings for FT Methods.* In the "pretrain and fine-tune" (FT) strategies, since different cities have unique user and POI sets, we train on the base cities in a "relay race" manner: firstly, we divide the city samples into training set and valid set, as in Section 3.4. Then, we run sufficient (empirically set as 15) epochs on city A, save the best weights based on valid set, then further train the saved weights on city B, and so on. In this way, the common parameter set, $\theta$, is sufficiently trained on all samples, while the pretrained-then-freezed POI ID embedding matrices could vary in different cities. In both non-transfer and FT methods, we tune the model parameters on the support set of each target city by sufficient 10 epochs and keep the best-validated parameters for inference on the query set. Additionally, the sample batch size and learning rate of these methods is uniformly set as 1024 and 0.001 for the superior performance (except iMTL, for which we use a learning rate of 0.0001 according to the released codes).

*A.1.4 Parameter Settings for Meta-Learning Methods.* For all the meta-learning methods (except $s^2$Meta [6], for which we use the parameter setting in the released codes), the task batch size $B_c$ of meta-learning approaches is set as 4 and 16 in *MapSmall* and *MapLarge*, respectively, while the user batch size $B_u$ of each meta-training task is set as 256 in both datasets. We arbitrarily set $k_c = k_u = 0.5$, which means half of the hardest cities/users are kept in the hardness aware mechanisms in CHAML. However, this hyper-parameter can be variant according to different scenarios. In addition, we set the local and global learning rates $\alpha = \beta = 0.001$ as in FT methods, and set the number of local-update steps as 1 for the purpose of fast adaption during inference. Actually, we find only slight improvement when we allow larger number of local updates. In addition, following the officially-released codes of [6], the max step of training iterations $M$ is set as 50,000 and 100,000 in *MapSmall* and *MapLarge* datasets, respectively, which are not large compared to the possible combinations of city batch and user sampling.

In CHAML, we set $\xi = 0.5$ and $\eta = 0.5M$, which means we select the easiest half of cities (judged by transfer methods) to present at the beginning, and when half of the meta training iterations are done, we then present all the cities to the meta-learner. Akin to the empirical findings in [10], we find almost the same performance when we adopt more complicated scheduler strategies (e.g., fixed exponential pacing in [10]).

*A.1.5 Cities in Different Datasets.* The selected cities in *MapSmall* are listed in Table 3. The selected cities in *MapLarge* are omitted due to space limit.

**Table 3: City division in *MapSmall* dataset.**

| City Type | City Names |
|---|---|
| Base City | Beijing, Chengdu, Chongqing, Guangzhou, Shanghai, Shenzhen, Tianjin, Xi'an |
| Valid City | Hangzhou, Nanjing |
| Target City | Dalian, Lanzhou, Qingdao, Xiamen |

## A.2 Supplementary Figures and Discussions

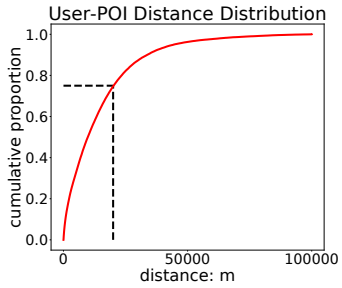



**Figure 4: The cumulative proportion of map search records in Beijing according to user-POI distance.**

*A.2.1 User-POI Distance.* Fig 4 demonstrates the cumulative proportion of the map search records in Beijing as a function of user-POI distance, where we randomly sample 100,000 users and collect their records from Sep. to Dec., 2019. One could clearly see that in about 3 quarters of the records the user-POI distances are less than 20 kilometers, which supports the conclusion that "most searches happen locally" in Section 3.3.1.

## A.3 Proof and Analysis for Proposition 1

This subsection provides the proof and analysis for **Proposition 1** in Section 3.4, following the work in [10].

To begin with, we recall and define some notations. In our setting of CHAML, we firstly sort the base cities from easiest to hardest by

the criteria of city-level difficulty $\delta_c$ defined in Eq.(10). We denote the sorted cities as $C_{\mathcal{B}} = \{c_1, c_2, \cdots, c_N\}$. For the first $\eta$ out of total $M$ iterations, we only present the $\xi N$ easiest base cities, i.e. $\{c_1, c_2, \cdots, c_{\xi N}\}$, as city sampling pool. For the rest of iterations, we present all the base cities $C_{\mathcal{B}}$. This strategy is what we call *single step pacing*, formalized in Eq.(11).

One could view the CHAML algorithm from the perspective of optimization. Recall that $\theta$ stands for the initial parameters of our base recommender $f$. Let $L_\theta(c)$ denote the cross entropy loss on the query set of city $c$ ($\mathcal{D}_c^{qry}$) after locally updating the recommender parameters on the support set of city $c$ ($\mathcal{D}_c^{spt}$) from $\theta$. Our goal is then to find the best recommender parameters $\tilde{\theta}$ to minimize the expectation of $L_\theta(c), c \in C_{\mathcal{B}}$. We can also write our goal as maximizing the following average *Utility* $U_\theta(c) = e^{-L_\theta(c)}$ of the observed data, which can be justified from first principles and defined as follows[7]:

$$\begin{aligned} \mathcal{U}(\theta) &= \hat{\mathbb{E}}[U_\theta] = \frac{1}{N}\sum_{i=1}^{N} U_\theta(c_i) \triangleq \frac{1}{N}\sum_{i=1}^{N} e^{-L_\theta(c)} \\ \tilde{\theta} &= \arg\max_\theta \mathcal{U}(\theta) \end{aligned} \tag{12}$$

In our curriculum setting in CHAML, the city-level difficulty $\delta_c$ effectively provides a prior probability for city sampling: $p(c_i) = p_i$. With this prior, the optimization goal above is changed as follows:

$$\begin{aligned} \mathcal{U}_p(\theta) &= \hat{\mathbb{E}}_{\mathbf{p}}[U_\theta] = \sum_{i=1}^{N} U_\theta(c_i)p(c_i) = \sum_{i=1}^{N} e^{-L_\theta(c)}p_i \\ \tilde{\theta} &= \arg\max_\theta \mathcal{U}_p(\theta) \end{aligned} \tag{13}$$

The prior probability $p(c_i)$ is determined by the *difficulty evaluation* in Eq.(10) and the *pacing function* in Eq.(11). Concretely, it is formalized as follows:

$$p(c_i) = \begin{cases} \frac{1}{K} & i \le K \triangleq \xi N \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

In our setting with single step pacing, $\mathcal{U}_p(\theta)$ is the optimization function for the first $\eta$ iterations, and $\mathcal{U}(\theta)$ for the rest iterations. Now we explore the difference between the two optimization goals.

**Lemma 1** *The difference between the expected utility computed with or without the prior probability* $\mathbf{p}$*, i.e.,* $\mathcal{U}_p(\theta)$ *and* $\mathcal{U}(\theta)$*, is the covariance of two random variables* $U_\theta(c)$ *and* $p(c)$*.*

*Proof.* From Eq.(12) (13) and (14):

$$\begin{aligned} &\mathcal{U}_p(\theta) - \mathcal{U}(\theta) = \frac{1}{K}\sum_{i=1}^{K} U_\theta(c_i) - \frac{1}{N}\sum_{i=1}^{N} U_\theta(c_i) \\ &= \sum_{i=1}^{K} U_\theta(c_i)(\frac{1}{K} - \frac{1}{N}) + \sum_{i=K+1}^{N} U_\theta(c_i)(0 - \frac{1}{N}) \\ &= \sum_{i=1}^{N} U_\theta(c_i)(p(c_i) - \frac{1}{N}) = \sum_{i=1}^{N} U_\theta(c_i)(p(c_i) - \hat{\mathbb{E}}(p)) \end{aligned} \tag{15}$$

Because $\sum_{i=1}^{N}(p(c_i) - \hat{\mathbb{E}}(p)) = \sum_{i=1}^{N} p(c_i) - N \cdot \frac{1}{N} = 0$:

$$\begin{aligned} &\mathcal{U}_p(\theta) - \mathcal{U}(\theta) \\ &= \sum_{i=1}^{N} U_\theta(c_i)(p(c_i) - \hat{\mathbb{E}}(p)) - \sum_{i=1}^{N}(p(c_i) - \hat{\mathbb{E}}(p))(\hat{\mathbb{E}}(U_\theta)) \\ &= \sum_{i=1}^{N}(U_\theta(c_i) - \hat{\mathbb{E}}(U_\theta))(p(c_i) - \hat{\mathbb{E}}(p)) = \hat{\mathrm{Cov}}[U_\theta, p]. \end{aligned} \tag{16}$$

Let us now have a closer look at the two random variables $U_\theta(c)$ and $p(c)$. First of all, we claim that both the variables are metrics for city-level difficulty, and they are positively correlated. To demonstrate this, we have to assume a reasonable and mild condition.

**Condition 1.** *$L_\theta(c)$ increases with the city-level difficulty $\delta_c$ defined by the difficulty evaluation in Eq.*(10).

In fact, this condition is a basic premise for $\delta_c$, since with the current $\theta$ fixed, a higher city-level difficulty value $\delta_c$ should cause a higher loss $L_\theta(c)$. Also, sharing the same structure between transfer teacher in Eq.(10) and base recommender $f$ in CHAML is motivated to meet this conditon. With Condition 1, we hold[8]:

$$U_\theta(c) = e^{-L_\theta(c)} \propto -L_\theta(c) \propto -\delta_c \propto p(c) \tag{17}$$

Furthermore, one may regard $p(c)$ as an approximate oracle for city-level difficulty, and $U_\theta(c)$ as the difficulty metric provided by the current recommender $f_\theta$. At the very beginning of meta training, parameters in $\theta$ are randomly initialized and thus the metric of $U_\theta(c)$ are unreliable. However, when $\theta \to \tilde{\theta}$, we assume the metric of $U_\theta(c)$ achieves its best reliability, i.e., it could evaluate the city-level difficulty and approximate the oracle $p(c)$ better than ever. This assumption brings another condition.

**Condition 2.** $\tilde{\theta} = \arg\max_\theta \mathcal{U}_p(\theta) = \arg\max_\theta \hat{\mathrm{Cov}}[U_\theta, p]$.

With the two reasonable conditions above, we could prove the following lemma.

**Lemma 2** $\mathcal{U}_p(\tilde{\theta}) - \mathcal{U}_p(\theta) \ge \mathcal{U}(\tilde{\theta}) - \mathcal{U}(\theta), \forall\theta$

*Proof.* From Lemma 1 and Condition 2:

$$\begin{aligned} &\mathcal{U}_p(\tilde{\theta}) - \mathcal{U}_p(\theta) = \mathcal{U}_p(\tilde{\theta}) - \mathcal{U}(\theta) - \hat{\mathrm{Cov}}[U_\theta, p] \\ &\ge \mathcal{U}_p(\tilde{\theta}) - \mathcal{U}(\theta) - \hat{\mathrm{Cov}}[U_{\tilde{\theta}}, p] = \mathcal{U}(\tilde{\theta}) - \mathcal{U}(\theta). \end{aligned} \tag{18}$$

Lemma 2 shows that during the first $\eta$ meta training iterations of CHAML with the optimization function $\mathcal{U}_p(\tilde{\theta})$, the gradient directions are overall steeper than those with the optimization function $\mathcal{U}(\tilde{\theta})$. Therefore, with the single step curriculum in the initial step of iterations, CHAML gets a higher likelihood of escaping local minumums by taking steeper global update steps to the global optimal $\tilde{\theta}$. This conclusion is actually the same as **Proposition 1** and the effectiveness of CHAML is further verified in our experiments.

[7] $\hat{A}$ denotes the empirical estimate of A for any A.

[8] $\propto$ stands for positive correlation in our context.