Cui JW, Lu W, Zhao X *et al.* Efficient model store and reuse in an OLML database system. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 36(4): 792–805 July 2021. DOI 10.1007/s11390-021-1353-5

Efficient Model Store and Reuse in an OLML Database System

Jian-Wei Cui, *Member, CCF*, Wei Lu, *Member, CCF*, Xin Zhao, and Xiao-Yong Du*, *Fellow, CCF*

Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education, Renmin University of China Beijing 100872, China

School of Information, Renmin University of China, Beijing 100872, China

E-mail: {cjwruc, uqwlu, xinzhao, duyong}@ruc.edu.cn

Received February 4, 2021; accepted June 27, 2021.

Abstract Deep learning has shown significant improvements on various machine learning tasks by introducing a wide spectrum of neural network models. Yet, for these neural network models, it is necessary to label a tremendous amount of training data, which is prohibitively expensive in reality. In this paper, we propose OnLine Machine Learning (OLML) database which stores trained models and reuses these models in a new training task to achieve a better training effect with a small amount of training data. An efficient model reuse algorithm AdaReuse is developed in the OLML database. Specifically, AdaReuse firstly estimates the reuse potential of trained models from domain relatedness and model quality, through which a group of trained models with high reuse potential for the training task could be selected efficiently. Then, multi selected models will be trained iteratively to encourage diverse models, with which a better training effect could be achieved by ensemble. We evaluate AdaReuse on two types of natural language processing (NLP) tasks, and the results show AdaReuse could improve the training effect significantly compared with models training from scratch when the training data is limited. Based on AdaReuse, we implement an OLML database prototype system which could accept a training task as an SQL-like query and automatically generate a training plan by selecting and reusing trained models. Usability studies are conducted to illustrate the OLML database could properly store the trained models, and reuse the trained models efficiently in new training tasks.

Keywords model selection, model reuse, OnLine Machine Learning (OLML) database

1 Introduction

Deep learning has been widely adopted in various machine learning tasks and shown generalization effects improvement. Neural networks usually achieve better effects when trained on a large amount of data. However, it is time consuming to label sufficient training data, and the training of neural models is also costly. From data-centric view, the neural model training process will generate various computed data artefacts, such as processed training data and trained models. Therefore, reusing existing computed data artefacts is a hopeful way to reduce the training cost and improve the training effect for a new task. In this paper, we propose OnLine Machine Learning (OLML) database which extends the traditional database system naturally to online respond to a training task by efficiently reusing the trained models. In the OLML database, better training effects could be achieved on limited training data by model reuse.

The OLML database views a training task as an adhoc query and aims to online respond to the query. We summarize the challenges of the OLML database into three aspects.

1) *Efficient Model Selection.* As we may have dozens of trained models available, the model reuse algorithm should efficiently select a group of models with high reuse potential for the training task.

2) *Effective Model Reuse.* The model reuse algorithm should combine and reuse the selected models

Regular Paper

Special Section on AI4DB and DB4AI

The work was supported by the National Natural Science Foundation of China under Grant No. 62072458.

*Corresponding Author

effectively to achieve a better training effect.

3) *System Architecture Design.* The architecture of the OLML database should extend the traditional database system naturally to online respond to the training tasks, and properly save the trained models to form a model training-and-reuse cycle.

For model selection, a direct way to estimate the reuse potential of a trained model for a training task is computing the prediction effect on the test data of the task. However, computing prediction effects for all the trained models may be inefficient. In this paper, we propose to estimate the model reuse potential from two aspects: domain relatedness and model quality, which could be computed very efficiently. For domain relatedness, we adopt a data-driven way to represent the domain of a model or a training task from its training data. For example, the domain of text-type training data could be represented as TF-IDF vectors, which could be used to measure the domain relatedness between a trained model and the training task. For model quality, the effects of trained models may differ due to many factors, such as training data size and data quality. In this paper, we adopt the average prediction effect on all trained tasks as a prior quality indicator for a trained model. Then, domain relatedness and model quality will be combined together to identify models with high reuse potential for the training task.

For model reuse, a common approach is initializing model parameters of a new training task from a trained model, also known as warm-up [1–4]. As we may select multi models with high reuse potential, each selected model will be trained from its warm-up and served as a base model for ensemble to boost the training effect for the training task. To encourage diverse base models, we weight the training data during training inspired by AdaBoost [5]. Specifically, the selected models will be trained iteratively on the training data of the task. After one selected model finishes training, it will predict the labels of training data, and the error predicted data items will be set with higher weights. In this way, selected models will be encouraged to focus on different items of training data, which introduce a higher model diversity that leads to a better model ensemble effect.

For system architecture design, the architecture of the OLML database is complied with the layered design of the traditional database system, including language layer, training layer, access layer and storage layer. The prototype system is built upon UER [6], which represents the structures and training processes of various neural networks in a universal format. This enables model configurations and training options to be specified in a descriptive way. The language layer accepts a training task as a SQL-like query and parses specified options for lower layers. The training layer implements the model selection and model reuse algorithms, and automatically generates a training plan for the query. The access layer provides read and write interfaces of data artefacts needed by the model selection and model reuse algorithms. In the storage layer, the trained models will be stored by the training task type, where domain vectors and model quality will be computed to make these models able to be reused in the new training tasks. The layered architecture of the OLML database enables each layer to be extended independently, where new data artefacts and model reuse algorithms could be integrated into the system easily.

The main contributions of this paper are summarized as the follows.

• We propose the OLML database which naturally extends the traditional database system to store the trained models and reuse these models in the new training tasks to achieve better training effects when the training data is limited.

• We propose a new model reuse algorithm. The algorithm efficiently selects models with high reuse potential through domain relatedness and model quality, and improves the model reuse effect by encouraging the training of diverse base models for ensemble. We evaluate the proposed model reuse algorithm on two NLP tasks. The experimental results demonstrate the proposed algorithm achieves a higher accuracy and training efficiency compared with the model trained from scratch on 5x larger data size for the sentiment classification task.

• Based on the proposed model reuse algorithm and system architecture, we implement an OLML database prototype. The prototype system could accept a training task as a SQL-like query and online respond to the query by automatically selecting and reusing stored models. The usability studies demonstrate the trained models could be stored properly in the system and form a model training-and-reuse cycle.

2 Related Work

The proposed method in this paper measures the domain relatedness in a data-driven way and reuses multiple models by ensemble. Meanwhile, the OLML database prototype could be viewed as a data management system for machine learning. In this section, we review related work on text data relatedness, model reuse and ensemble, and data management systems for machine learning.

2.1 Text Data Relatedness

The relatedness between text data could be measured in both unsupervised and supervised ways. For the unsupervised way, the text is usually represented as a vector and cosine similarity could be computed to measure the relatedness. A traditional way to convert the text to a vector is representing the text in word space with the weight of each dimension computed as TF-IDF of that word. In deep learning, the latent distribution semantics of a word could be learned from large volume of corpus[7–9], where each word is represented as a dense vector, also known as word embedding. Based on word embedding, the embedding of paragraph could be computed to present the semantics for the paragraph[10], which could be used to evaluate relatedness between paragraphs.

For the supervised way, the training data of the new task usually serves as positive training samples, and the language model or classification model could be trained to evaluate the data relatedness and used in data selection[11, 12]. In [11], an in-domain language model is trained and the perplexity is computed on out-domain data to evaluate the relatedness. In [12], a binary classification is trained as in-domain data serves as positive samples and out-domain data serves as negative samples, and then out-domain data with high positive classification probability will be selected to be combined with in-domain data to improve the effect of in-domain task.

In this paper, as we only have limited training data for the training task, the supervised method for data relatedness measurement can also achieve limited effect. Therefore, we adopt unsupervised methods to measure the data relatedness which is usually more efficient compared with the supervised methods as no extra language model or classification model needs to be trained.

2.2 Model Reuse and Ensemble

Model reuse could be viewed as an application of transfer learning[13], where the parameters of the trained model are transferred to the model of the new task to improve the training performance. A common approach to reusing trained models is warm-up, which starts a new training with parameters initialized from a trained model. Many previous studies have adopted this technique, and obtained training performance improvement. For effectiveness improvement, Rafiki[2] adopts warm-up to train models for hyper-parameter turning, and achieves better effect. In [14], the model trained on large out-of-domain data is reused as the base model, and the training is continued on in-domain data to improve translation quality. For efficiency improvement aspect, COLUMBUS[15] proposes to cache models and choose a nearby model for a warm-up, achieving significant training speed-up. EDDE[3] initializes parameters of lower layers for a new training from previous trained models, and the result demonstrates such a reuse method could train base models more efficiently for ensemble. In [4], models trained on the same data are reused as warm-up in collaborative environment to achieve a higher training efficiency. In summary, these studies mainly focus on reusing a single model for the same task trained on the same data. Recently, the LEEP score[16] is proposed to estimate the transferability of existing models for a new task or dataset. However, the LEEP score needs to compute the prediction results for all existing models, which is not sufficient enough when there are dozens of existing models. In this paper, we propose methods to select models with high reuse potential efficiently and boost training effect by reusing multi models trained on different datasets.

Model ensemble is widely adopted to boost the training effect from multi models, such as Bagging[17], Boosting[18] and Stacking[19]. In [20], MoreBoost is proposed to select a group of existing models to cover the training data domain of the new task, and the selected models will then boost the prediction effect by ensemble. Different from MoreBoost, there is training data for the new task in our setting. We therefore borrow the ideas from AdaBoost[5] to continuously train multi selected models in an adaptive way, achieving better ensemble effect by encouraging diverse base models.

2.3 Data Management System for Machine Learning

There are many efforts to improve machine learning performance through data management techniques. For model management, ModelHub[21] focuses on the storage of models for multi versions, which seeks to balance the computing and the storage cost. ModelDB[22] proposes a model-centric method to store and exhibit various data artefacts, which facilitates to trace the training process. For dataset management, DataHub[23] manages multi-versions of the same dataset to avoid recomputation in the new task. Compared with these systems, the OLML database implemented in this paper focuses on the model reuse capability to respond to new training tasks. Therefore, the data storage and management techniques of existing systems could be integrated into the OLML database.

For automatic model training, MLBase[24] and Rafiki[2] search algorithms to achieve good effect for a given task. Different from these systems, the OLML database aims to online respond to training tasks. Therefore, we prefer efficient model reuse methods to automatically generate the training plan for the new task. Helix[25] and [4] adopt DAG to represent the data dependency for training, which reuses existing data and models trained on the same dataset for the new task. Compared with these systems, the OLML database supports the reuse of multi models trained on different datasets, and also develops adaptive training techniques to improve the effect of model ensemble. Therefore, the OLML database could serve as an advanced model reuse module in the whole machine learning pipeline.

3 Model Reuse

In this section, we first formally define the model reuse problem, and then introduce the proposed model selection and reuse methods.

3.1 Problem Definition

For a training task T, with training data and test data denoted as $D(T)$ and $TD(T)$ respectively, assume $D(T)$ is not sufficient to train a model achieving a high accuracy. Meanwhile, assume we have dozens of models trained on different datasets with the same input and the output type of T. We denote $R = \{M_1, M_2, ..., M_t\}$ to represent such model repository. The training data and the test data of M_i are denoted as $DS(M_i)$ and $TD(M_i)$ respectively. The text of each training data item is viewed as a sentence, and the domains of training data for each model may be different. Then, the goal of the proposed methods is to select and reuse K models from R to achieve better training effect for task T, where K is specified.

3.2 Model Selection

Model selection estimates model reuse potential to efficiently select a group of trained models to be reused in the training task. We measure the reuse potential from two aspects: domain relatedness and model quality. For domain relatedness, models trained on data sharing a similar domain with the training task tend to have higher reuse potential. Therefore, we measure the domain relatedness between a trained model M in R and the training task T through the relatedness between training data. Specifically, as the training data size of trained models may be large, we keep a sampling for the training data of M and T, denoted as $DS(M)$ and $DS(T)$ respectively. Then, the domain vector of data item d in $DS(M)$ or $DS(T)$, denoted as $v(d)$, could be computed by TF-IDF or average of word embedding vectors. The domain relatedness between M and T is denoted as $Rel(M, T)$ and computed as:

$$\begin{aligned} Rel(M,T) = & \frac{1}{|DS(M)| \times |DS(T)|} \times \\ & \sum_i \sum_j \text{cosine}(v(d_i), v(d_j)), \end{aligned}$$

where $d_i \in DS(M)$ and $d_j \in DS(T)$. For trained models, the samplings of training data and corresponding domain vectors could be calculated offline, and only the data sampling and domain vectors of the training task need to be calculated online. Therefore, $Rel(M, T)$ could be computed efficiently between T and models in R.

For model quality, models from similar domains may also perform differently when reusing on other tasks. For example, models trained on higher quality and sufficient training data tend to perform better compared with models trained on lower quality and insufficient training data. Therefore, we estimate the model quality of M by the prediction effect on the test data of other models in R. Specifically, the quality of model M is denoted as $Qual(M)$ and computed as:

$$Qual(M) = \text{Avg}\left(\frac{effect(DT(M_j)|M)}{effect(M_j)}\right),$$

where $M_j \in R$ and $M_j \neq M$, and $effect(DT(M_j)\ |M)$ is normalized by $effect(M_j)$. $effect(M)$ could be defined based on the task type of M. For the classification task, $effect(M)$ could be defined as the classification accuracy, and for the sequence labelling task, $effect(M)$ could be defined as $F1$ score of labelled slots. As the model quality is evaluated on the existing tasks, $Qual(M)$ could be computed offline as new models are added in the repository. Therefore, $Qual(M)$ could be viewed as a priori reuse potential predictor, where no extra online computation is needed for the new task.

Combining domain relatedness and model quality, we estimate the reuse potential of model M for given task T as:

$$P(T|M) = Rel(T, M) \times Qual(M).$$

3.3 Model Reuse

For the training task T, a simple model reuse method is to initialize the parameters from a trained model with the maximum $P(T|M)$ score and then continuously train the model on $D(T)$. This could be viewed as single model reuse. As we may have multi models with high reuse potential, it is naturally to boost the accuracy from multiple trained models. The intuition behind multiple model reuse is that each trained model may help the training task to achieve a higher accuracy from different aspects. Therefore, it is possible to achieve better effect if multiple models are combined.

A direct way to combine multiple trained models is selecting K models with the maximum $P(T|M)$ scores, and continuously training each model on $D(T)$. The converged models will serve as base models for ensemble. Specifically, the softmax layer outputs of each continuously trained model will be averaged to jointly decide the prediction. This could be viewed as selecting top K models to reuse (TopK for short). Meanwhile, as indicated in [3], the effect of ensemble will be more pronounced if base models are more diverse. Consequently, we develop the AdaReuse algorithm inspired by AdaBoost (shown in Algorithm 1) to improve the model diversity by weighting training data during continuous training.

Algorithm 1. AdaReuse

```
Input:
  task $T$ with training data
  $D = (x_1, y_1)(x_2, y_2)...(x_n, y_n)$, initial data weight $\boldsymbol{W}^1$:
  $w_i^1 = 1$ for $1 \leqslant i \leqslant n$, and max-weight: $\beta$;
  model repository $R = (M_1, M_2, ..., M_t)$;
  the number of reuse models $K$ ($K \geqslant 1$)
Output:
  ensemble model $H$
1: Select $K$ models as $\{M_1, M_2, ..., M_k\}$ from $R$ with the
   highest $P(T|M)$ and sort descending
2: for $k = 1$ to $K$ do
3:     $h_k = Warmup(M_k)$
4:     $h_k = ContinousTraining(h_t, D, \boldsymbol{W}^k)$
5:     $e_k = 1 - \sum_i^n I(h_t(x_i), y_i)/n$
6:     $\alpha_k = \frac{1}{2}\log((1 - e_k)/e_k)$
7:     $w_i^{k+1} = \min(w_i^k \times \max(1, \exp(\alpha_k)), \beta)$ for $h_t(x_i) \neq y_i$
8:     $H = \max\limits_{1 \leqslant k \leqslant K} \left(\sum_i^k \alpha_k \times h_i\right)$
```

In Algorithm 1, all the training data have equal weight before training, and AdaReuse firstly selects K models with the maximum $P(T|M)$ scores (line 1). Then, the training is processed iteratively from the model with maximum $P(T|M)$ score. In each iteration, the result model h_k is initialized from a selected model M_k and then continuously trained on D weighted by $\boldsymbol{W}^k$. After the training converged, the error rate of h_k is calculated as e^k (line 5), where $I(h_t(x_i), y_i)$ equals 1 for $h_t(x_i) = y_i$ and equals 0 for $h_t(x_i) \neq y_i$. The judge of $h_t(x_i) = y_i$ depends on the task type. For classification task, $h_t(x_i) = y_i$ means the classification label of x_i predicted by the model equals the truth label. While for the sequence labelling task, $h_t(x_i) = y_i$ means the labels predicted by the model are equal to corresponding truth labels at any position of x_i. The contribution ratio of h_k to the final ensemble model is computed as α_k (line 6), where a lower error rate will lead to a higher contribution ratio. After that, the weight of training data item error predicted by h_k will increase by $\exp(\alpha_k)$ (line 8). Here, as the training data is limited, we do not decrease the weight of correctly predicted data to avoid ignoring any training data. Meanwhile, the weight is restricted to range $[1, \beta]$ to avoid too large value caused by very small e^k. In this way, the error-predicted data will get more attention in next iteration training, which helps to improve the model diversity. After K models are all continuously trained, they are served as base models for ensemble where the softmax layer outputs of models are averagely weighted by α_k (line 8).

In our situation, the effect of K continuously trained models may vary greatly, and more base models may not always achieve better effect for ensemble, as base models with significant lower accuracy may affect the ensemble effect. Therefore, we simply search the best number of base models for ensemble to achieve the best effect (line 8).

4 Experiment

To validate the effectiveness of proposed work, we conduct experiments for two types of NLP tasks, sentiment classification (SC) and name entity recognition (NER). In this section, we first introduce the datasets and experimental settings. Then, we study the effect of model selection and single model reuse on different data sizes. After that, different methods to reuse multiple models are compared to validate the effect of the AdaReuse algorithm. And lastly, we compare the training efficiency between AdaReuse and a data selection method.

4.1 Datasets and Experimental Settings

For sentiment classification, there are dozens of public available datasets for Chinese sentiment classification. These datasets are released by different organizations from different domains. Table 1 lists the summary for all collected datasets, where we align the outputs of all the datasets to positive and negative categories. We select hotel and book review sentiment classifications as test tasks, and the others to build the model repository.

Table 1. Dataset Summary and Accuracy of SC

Usability	Name	Size ($\times10^3$)	Pos:Neg	Acc
Model repository	Waimai	9	2:1	0.834
	Weibo	20	1:1	0.972
	Clothing	8	1:1	0.916
	Computer	2	0.95:1	0.851
	Fruit	8	1:1	0.885
	Pda	8	1:1	0.887
	Shampoo	8	1:1	0.901
	Stockmarket	7	1:1	0.915
	Shop	20	0.9:1	0.912
	Inews	1	1:1	0.586
	Movie	20	1:1	0.833
	Dianping	20	1:1	0.821
Test task	Hotel	5	0.46:1	0.812
	Book	20	1:1	0.746

For name entity recognition, we conduct experiments on CLUENER[26] which consists of 10 types of name entities. The dataset is divided by the name entity type, and Table 2 lists the summary for each type of name entity. We select movie and book entity recognition as test tasks, and the others are used to build the model repository.

Table 2. Dataset Summary and $F1$ of NER

Usability	Name	Size ($\times10^3$)	$F1$
Model repository	Address	2.10	0.330
	Organization	1.90	0.443
	Company	2.20	0.346
	Game	1.90	0.549
	Government	1.50	0.333
	Name	2.80	0.389
	Position	2.50	0.404
	Scene	0.95	0.322
Test task	Movie	0.80	0.495
	Book	0.90	0.481

We adopt convolution neural network (CNN) and Transformer[27] as the neural network architectures for sentiment classification. For name entity recognition, as the training data is too limited, we adopt a Chinese pre-trained model[①] RoBERTa-Tiny[28] as the base model and fine-tune it on the training data. Table 3 shows the hyper-parameters for the three neural network architectures. Batch size and learning are 32 and $2e-5$ respectively, and keep the same for training on all the datasets. We conduct all the experiments by using UER[6], with which the training effect could be reproduced easily.

Table 3. Hyper-Parameters of Neural Networks

Architecture	Hyper-Parameter
CNN (sentiment classification)	Layer num = 4, hidden size = 256, kernel size = 3
Transformer (sentiment classification)	Layer num = 12, hidden size = 128
RoBERTa-Tiny (name entity recognition)	Heads num = 2, Feedforward size = 512

Table 1 and Table 2 list the accuracy and $F1$ scores for models trained on each dataset respectively. For sentiment classification, most models achieve accuracies higher than 0.8. For name entity recognition, most models achieve $F1$ lower than 0.5 as the training data is more limited.

4.2 Model Selection

We evaluate the effect of model reuse potential estimation in this subsection.

For sentiment analysis, $acc(DT(T)|M)$ is the prediction accuracy of model M on the test data of task T, and is used as the ground truth of model reuse potential. A good reuse potential estimation method should achieve a high $acc(DT(T)|M)$ score. Table 4 shows the comparison of $acc(DT(T)|M)$ scores estimated by different methods, where the average $acc(DT(T)|M)$ score on two test tasks and top K models are computed. RANDOM means randomly selecting K models from R, and BEST means testing all trained models on $DT(T)$ and selecting K models with the maximum $acc(DT(T)|M)$ scores. For $Rel(M,T)$ and $P(T|M)$, the data relatedness in terms of TF-IDF vector and average of word embedding vectors (EMB) are computed.

As shown in Table 4, both $Rel(M,T)$ and $Qual(M)$ perform significantly better than RANDOM especially

[①]https://huggingface.co/uer/chinese_roberta_L-2_H-128, June 2021.

Table 4. Average $acc(DT(T)|M)$ of Model Reuse Potential Estimation for Sentiment Analysis

K	RANDOM	$Rel(M,T)$ (TF-IDF)	$Rel(M,T)$ (EMB)	$Qual(M)$	$P(T\|M)$ (TF-IDF)	$P(T\|M)$ (EMB)	BEST
1	0.60	**0.73**	0.62	0.65	**0.73**	0.65	0.73
2	0.60	0.69	0.63	0.64	**0.71**	0.64	0.71
3	0.60	0.62	0.64	0.65	**0.68**	0.65	0.69
4	0.60	0.63	0.64	0.64	**0.67**	0.64	0.68

Note: The best results are in bold.

when K is smaller, demonstrating both $Rel(M,T)$ and $Qual(M)$ can help to estimate model reuse potential. The improvement compared with RANDOM becomes smaller when K is larger, as the accuracy tends to be close to the average accuracy of all models in R. $P(T|M)$ performs better than both $Rel(M,T)$ and $Qual(M)$ for different K, illustrating combining domain relatedness and model quality could help to find models with high reuse potential. Compared with word embedding vectors, $Rel(M,T)$ in terms of TF-IDF vector performs better as corresponding $P(T|M)$ achieves a close accuracy to the BEST method. Table 5 illustrates the effect of reuse potential estimation for name entity recognition. $F1(DT(T)|M)$ is the prediction $F1$ score of model M on the test data of T, and is used as ground truth. Table 5 shows similar results to Table 4, demonstrating the effectiveness of model reuse potential estimation for the NER task.

Table 6 shows the computation details of $P(T|M)$ for hotel review sentiment classification. For $Rel(M,T)$, the top 4 models are from tasks of dianping, computer, waimai and shop. On the other hand, $Qual(M)$ scores of models from the computer task and the waimai task are significant lower than those of shop task. Consequently, domain relatedness and model quality are complementary to some extent, and combing the two scores could better estimate model reuse potential. Meanwhile, the top 3 models in terms of $acc(DT(T)|M)$ for the hotel task are different from those of the book task. This demonstrates trained models have different reuse potential for different training tasks. Therefore, it is necessary for different training tasks to find corresponding models to reuse.

4.3 Single Models Reuse

We study the training effect of single model reuse for different data sizes. Figs.1(a) and 1(b) show the accuracies of hotel and book review sentiment classification tasks, respectively, with CNN for training without model reuse (No-Reuse), training on a random selected model (Rand1), and on a model with the maximum $P(T|M)$ score (Top1). When the training data size is small, Rand1 and Top1 both achieve an accuracy much higher than No-Reuse, demonstrating the effect of model reuse when training data is very limited. Specifically, the accuracies are 0.696, 0.728 and 0.814 respectively for the hotel task with the data size of 1 000, and Top1 reveals about 12% accuracy improvement. And the training for No-Reuse cannot converge when the data size is smaller than 4 000, while the accuracy of Top1 gradually increases. For the book task

Table 5. Average $F1(DT(T)|M)$ of Model Reuse Potential Estimation for Name Entity Recognition

K	RANDOM	$Rel(M,T)$ (TF-IDF)	$Rel(M,T)$ (EMB)	$Qual(M)$	$P(T\|M)$ (TF-IDF)	$P(T\|M)$ (EMB)	BEST
1	0.16	**0.43**	0.10	0.20	**0.43**	0.20	0.43
2	0.16	**0.33**	0.20	0.31	**0.33**	0.31	0.34
3	0.16	**0.30**	0.23	0.25	**0.30**	0.25	0.30
4	0.16	0.24	0.22	0.25	**0.25**	0.25	0.26

Table 6. Computation Details of Model Reuse Potential Estimation for the Hotel Task of Sentiment Classification

Item	waimai	weibo	cloth	compu	fruit	pda	shamp	stock	shop	inews	movie	dianp
$Qual(M)$	0.75	0.59	0.82	0.75	**0.85**	**0.84**	**0.83**	0.58	0.82	0.53	0.71	0.73
$Rel(M,T)$	**0.15**	0.09	0.14	**0.16**	0.14	0.14	0.13	0.03	**0.15**	0.02	0.08	**0.25**
$acc(hotel)$	0.41	0.63	0.72	0.70	**0.73**	0.70	0.69	0.55	**0.74**	0.30	0.72	**0.76**
$acc(book)$	0.55	0.47	0.54	0.54	0.58	0.56	0.59	**0.60**	0.58	0.52	**0.70**	**0.63**

Note: compu and dianp are short for computer task and dianping task respectively, and $acc(hotel)$ and $acc(book)$ are short for $acc(DT(T)|M)$ for the hotel task and the book task respectively.

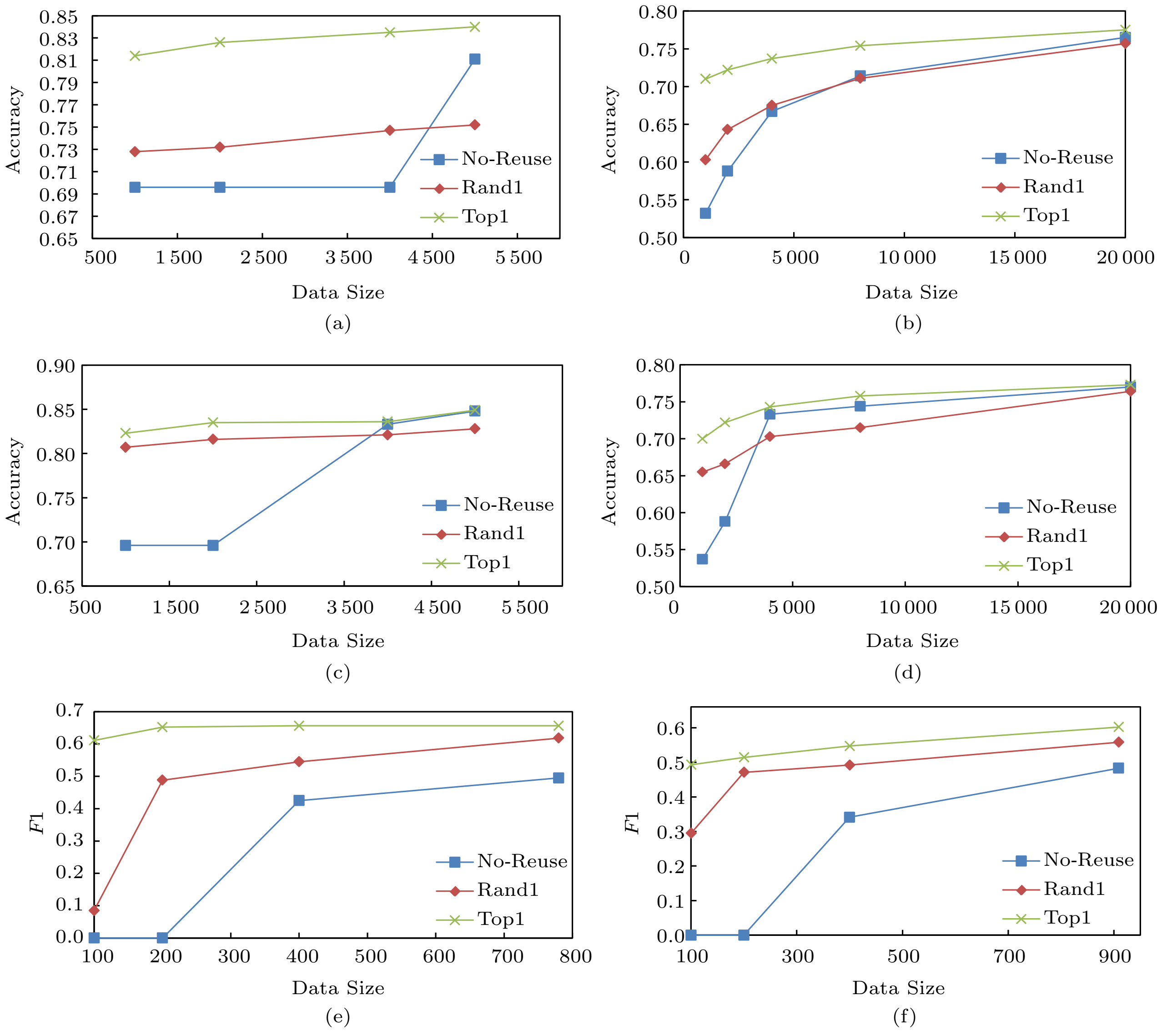


Fig.1. Effect of single model reuse. (a) Accuracy for hotel review sentiment classification with CNN. (b) Accuracy for book review sentiment classification with CNN. (c) Accuracy for hotel review sentiment classification with Transformer. (d) Accuracy for book review sentiment classification with Transformer. (e) $F1$ score for movie entity recognition with RoBERTa-Tiny. (f) $F1$ score for book name entity recognition with RoBERTa-Tiny.

with the data size of 1 000, the accuracies are 0.532, 0.603, and 0.710 respectively, and Top1 reveals about 18% accuracy improvement.

When the training data size becomes larger, the accuracy of No-Reuse will exceed that of Rand1, but still lower than that of Top1 even on all training data. Specifically, the accuracies are 0.811, 0.752 and 0.840 respectively for the hotel task with the data size of 5 000, and Top1 reveals about 3% accuracy improvement. For the book task with the data size of 20 000, the accuracies are 0.765, 0.757, and 0.775 respectively, and Top1 reveals about 1% accuracy improvement. Figs.1(c) and 1(d) show the accuracy of single model reuse with Transformer, which shows similar results to Figs.1(a) and 1(b). Compared with CNN, we can see the accuracy of Transformer improves faster as the training data size increases. Figs.1(e) and 1(f) show the $F1$ score of single model reuse for movie and book entity recognition tasks, respectively. We can see Top1 and Rand1 both perform significantly better compared with No-Reuse as the training data size is more limited. In summary, Fig.1 demonstrates model reuse usually improves the training effect significantly when the training data is very limited, and selecting a model with high reuse potential could benefit the training effect for a large range of training the data size.

4.4 Multi Models Reuse

We study the effect of multi models reuse when training data is limited in this subsection. Table 7 illustrates the accuracy for hotel and book review sentiment classification tasks with CNN where data size is 1 000, 2 000 and 4 000. For AdaReuse, the hyper-parameter of maximum weight β is set to 2^k. In Table 7, Rand2 (or Rand3) and Top2 (or Top3) represent the accuracy of model ensemble on 2 (or 3) models randomly selected or with the maximum $P(T|M)$ scores. Ada2 (or Ada3) means the AdaReuse algorithm with K equal to 2 (or 3).

As shown in Table 7, the accuracies of different settings for Top2 (Top3) and Ada2 (Ada3) are both higher than those of Rand2 (Rand3), illustrating the effectiveness of model reuse potential estimation. Meanwhile, Top2 (Top3) and Ada2 (Ada3) both achieve higher accuracies compared with Top1 for the data size of 1 000 and 2 000, showing the model ensemble could also improve the accuracy for high reuse potential models when training data is limited. And Ada2 (Ada3) performs better than Top2 (Top3), demonstrating model diversity introduced by AdaReuse could help to improve the ensemble effect. On the other hand, Top3 does not always achieve a higher accuracy compared with Top2, indicating the ensemble of more models with lower accuracies cannot always improve the training effect. Therefore, it is necessary to search for the best base model number in AdaReuse. For the data size of 4 000, Top2 (Top3) and Ada2 (Ada3) cannot improve the accuracy significantly compared with Top1. This is because the base models will all achieve higher accuracies when training data is more sufficient. In this situation, the relative improvement of model ensemble will become small.

Table 8 shows the accuracies of hotel and book review sentiment classification tasks with Transformer, which have similar results to Table 7. Table 9 illustrates the $F1$ score of movie and book entity recognition tasks with RoBERTa-Tiny. We can see Top2 (Top3) and Ada2 (Ada3) both achieve $F1$ scores significantly higher than those of Rand2 (Rand3). However, Top2 and Top3 perform worse than Top1, and Ada2 and Ada3 could achieve $F1$ scores slightly higher than Top1 only in some settings. The reason is that the $F1$ score of the model for Top1 is significantly higher than those of other models in the model repository, making combining other trained models together with the model of Top1 unable always to improve the result. Based on the above analysis, we can conclude that multi-model reuse tends to perform better when there are more models

Table 7. Accuracy of Multi Model Reuse for Hotel and Book Review Sentiment Classification with CNN

Model Reuse Algorithm	Hotel Task			Book Task		
	Size = 1 000	Size = 2 000	Size = 4 000	Size = 1 000	Size = 2 000	Size = 4 000
Rand1	0.754	0.774	0.781	0.609	0.641	0.669
Rand2	0.790	0.797	0.802	0.656	0.671	0.687
Rand3	0.790	0.804	0.811	0.658	0.685	0.701
Top1	0.809	0.823	0.833	0.717	0.728	**0.744**
Top2	0.828	0.831	0.840	0.710	0.731	0.737
Top3	0.823	0.833	**0.839**	0.689	0.712	0.723
Ada2	**0.835**	0.835	0.836	**0.721**	0.731	**0.744**
Ada3	**0.835**	**0.837**	0.836	**0.721**	**0.737**	**0.744**

Table 8. Accuracy of Multi Model Reuse for Hotel and Book Review Sentiment Classification with Transformer

Model Reuse Algorithm	Hotel Task			Book Task		
	Size = 1 000	Size = 2 000	Size = 4 000	Size = 1 000	Size = 2 000	Size = 4 000
Rand1	0.788	0.815	0.822	0.643	0.662	0.698
Rand2	0.803	0.820	0.832	0.685	0.685	0.713
Rand3	0.808	0.823	0.839	0.692	0.692	0.720
Top1	0.823	0.835	0.832	0.700	0.722	0.743
Top2	0.834	0.840	0.845	0.717	0.720	0.736
Top3	0.833	0.835	0.837	0.705	0.722	0.734
Ada2	**0.838**	0.842	**0.848**	0.710	0.723	**0.744**
Ada3	**0.838**	**0.843**	**0.848**	**0.719**	**0.726**	**0.744**

Table 9. *F*1 Score of Multi-Model Reuse for Movie and Book Name Entity Recognition with RoBERTa-Tiny

Model Reuse Algorithm	Movie Task			Book Task		
	Size = 100	Size = 200	Size = 400	Size = 100	Size = 200	Size = 400
Rand1	0.320	0.424	0.581	0.190	0.372	0.481
Rand2	0.282	0.532	0.606	0.229	0.441	0.516
Rand3	0.355	0.530	0.610	0.252	0.474	0.523
Top1	0.669	0.669	**0.672**	**0.533**	**0.538**	0.569
Top2	0.614	0.632	0.646	0.505	0.511	0.543
Top3	0.625	0.634	0.660	0.505	0.527	0.558
Ada2	0.669	0.669	0.672	**0.533**	**0.538**	0.569
Ada3	**0.672**	**0.678**	**0.672**	**0.533**	**0.538**	**0.570**

with comparable reuse potential existing in the model repository, and the improvement tends to be more significant when the training data size is very limited.

4.5 Comparison with MoreBoost and LEEP

We compare AdaReuse with MoreBoost[19] and LEEP[15] for multi-model reuse. Particularly, when selecting existing models to reuse, MoreBoost needs a reusability indicator to indicate whether a training sample of the training task is similar to the training data of a model in the model repository[19]. We therefore train a binary classifier for each existing model with training data of the existing model serving as positive samples, and the same number of data items randomly sampled from training data of other existing models serving as negative samples. Then, the positive possibility of the classifier for an existing model is used as reusability indicator. For LEEP, the LEEP scores of existing models are computed based on the prediction results of training data of the training tasks.

Table 10 shows the effect comparison among AdaReuse, MoreBoost and LEEP in terms of hotel and book review sentiment classifications (SC). We can see AdaReuse performs better than MoreBoost in all settings. This is because AdaReuse considers not only the domain relativeness but also the model quality when selecting an existing model to reuse. On the other hand, AdaReuse also performs better than LEEP in most settings, demonstrating the effectiveness of the model selection and model diversity introducing methods proposed in this paper.

For efficiency comparison, MoreBoost compares all the existing models by reusability indicator to select only one model to reuse. Therefore, the time complexity for model selection is $O(K)$ when selecting K models to reuse. LEEP needs to compute the prediction results for all the existing models on the training data of the training task. Therefore, the time complexity for model selection is $O(N)$ when there are N existing models. On the other hand, AdaReuse only needs to compute the domain relatedness, which is a cosine similarity between two vectors, and select K models after comparing all the existing models only once. Therefore, the time complexity for model selection of AdaReuse could be viewed as $O(1)$. Based on the above comparison, we can see AdaReuse is more scalable when there are more existing models in the model repository. Consequently, AdaReuse is more suitable for the OLML database to online respond to the training tasks.

4.6 Efficiency Study

In this subsection, we study the training efficiency of AdaReuse for sentiment classification. The comparison is conducted in terms of training steps for different methods to achieve the same accuracy. Table 11 shows the comparison result. We set the accuracy tar-

Table 10. Accuracy Comparison of AdaReuse, MoreBoost and LEEP in Terms of Hotel and Book Review SC

Model Reuse Algorithm	Hotel Task			Book Task		
	Size = 1 000	Size = 2 000	Size = 4 000	Size = 1 000	Size = 2 000	Size = 4 000
Ada2 (CNN)	**0.835**	0.835	0.836	**0.721**	0.731	**0.744**
MoreBoost2	0.811	0.825	0.825	**0.721**	0.723	0.740
LEEP2	0.809	0.831	**0.840**	0.703	0.724	0.737
Ada3 (CNN)	**0.835**	**0.837**	0.836	**0.721**	**0.737**	**0.744**
MoreBoost3	0.812	0.821	0.833	0.718	0.727	0.739
LEEP3	0.817	0.835	0.835	0.706	0.729	0.742

Table 11. Training Efficiency Comparison (Accuracy/Number of Steps)

Task	Ada2@1000	MoreData	DataSel@4000	DataSel@8000	DataSel@20000
Hotel	**0.835/186**	0.810/468	0.720/375	0.772/750	0.808/1 875
Book	**0.731/186**	0.731/1404	0.625/375	0.682/750	0.679/1 875

get as the accuracy of Ada2 when the training data size is 1 000 in Table 7. MoreData means training a single model from scratch with more training data. Data selection is another way to improve the training effect with limited data by selecting task relevant data from other domains. In our situation, we first train a data selection classifier which uses training data of training task as positive samples, and randomly samples the same amount of data from the training data of other tasks as negative samples. Then, the positive probability of training data from other tasks is predicted by the classifier, with which the data samples with the highest positive probabilities are selected. DataSel@4000 means firstly training a base model on 4 000 selected data samples, and then continuously training on 1 000 data samples of the training task.

In Table 11, Ada2@1000 achieves about 0.835 and 0.731 accuracy after 186 steps training as it trains the two base models for ensemble respectively. For the hotel task, MoreData trains a single model on all the training data, which takes 468 steps on 5 000 training data. However, the accuracy does not achieve 0.835. For the book task, MoreData achieves 0.731 accuracy when training on about 15 000 data samples, which takes about 1 404 steps. This demonstrates AdaReuse could achieve a higher accuracy compared with the model trained on 5x larger training data with at least 2.5x more training steps. More importantly, as labelling training data is usually time consuming, labelling 5x larger data may take more time than training. For data selection methods, DataSel@8000 and DataSel@20000 both achieve a higher accuracy compared with DataSel@4000 as more relevant data is selected. However, DataSel@20000 cannot always perform better than DataSel@8000 as more selected data is not relevant enough. Although DataSel@8000 spends 4x training steps and DataSel@20000 spends 10x training steps, they fail to achieve the same accuracy of Ada2@2000. This also illustrates the efficiency and effectiveness of AdaReuse, as the data selection classifier cannot achieve a high enough accuracy when training data is limited and the model must be trained from scratch on selected data.

5 Implementation of a Prototype System

In this section, we first introduce the architecture of OLML database prototype, and then conduct a usability case study.

5.1 System Architecture

Fig.2 shows the architecture of the OLML database prototype system.

Language Layer. The layer is built based on UER[6] which provides universal representation for neural network architecture and training process of NLP tasks. The layer accepts a training task as a SQL-like description query, which simplifies training options issuing.

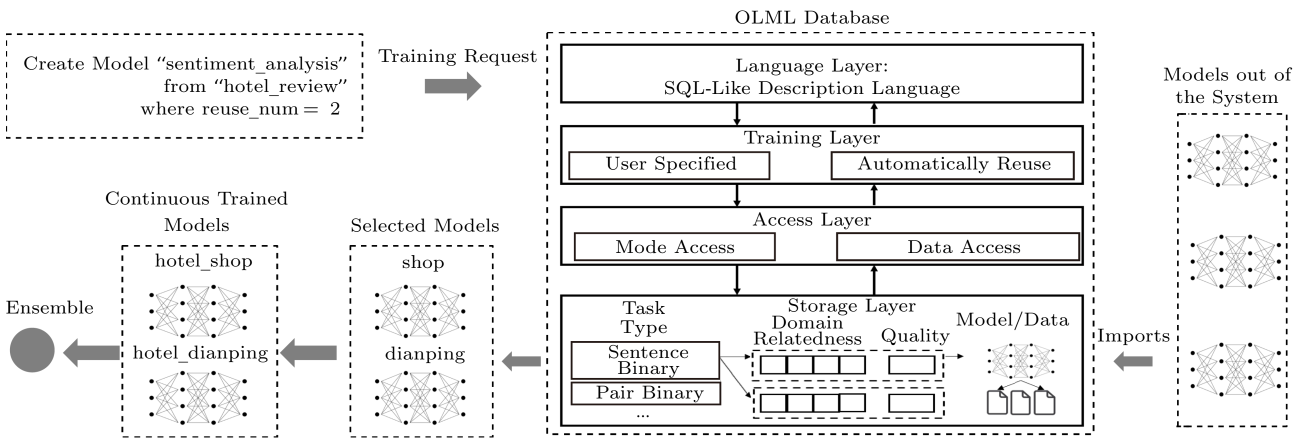


Fig.2. System architecture of OLML database.

Fig.3(a) shows the example for the shop task, where the training plan is specified by the user. The Create keyword indicates the model name which is the unique identifier for the training task. The From keyword specifies the data file directory containing training data, validate data and test data, where the file format explains the task type, such as single sentence binary classification and sentence pair binary classification. And the Where keyword specifies the training options, where only a few necessary options must be set by the user as most options have suitable default values. Fig.3(b) shows the SQL-like query for training the hotel task, where the neural network architecture is not specified. The system will automatically select and reuse two trained models to achieve a high accuracy.

```
Create Model shop_cls
  From "shop_review"
  Where
   encoder = "CNN"
   epcho_num = "3"
```

```
Create Model hotel_cls
  From "hotel_review"
  Where
   reuse_algo = "AdaRe"
   reuse_num = "2"
```

Fig.3. SQL-like queries for the hotel task. (a) Training plan by user. (b) Automatically reuse.

An advantage of the SQL-like language is that the OLML database could be viewed as a natural extension of the traditional database system. For example, the model could be extended as a built-in data type of database where training options could be managed by the meta-table. Meanwhile, the training data path could also be a database table. In this way, the model could be trained inside the database, which avoids moving data out of the database system.

Training Layer. This layer executes the training plan for a training task. The training plan could be specified by the user or generated automatically by implementing the AdaReuse algorithm. All the trained models will be stored in the system for future reuse.

Access Layer. This layer provides high-level data read and write interfaces, such as reading K models with the highest $P(T|M)$ scores for the training task, and reading a sample of training data of some trained model.

Storage Layer. This layer stores models by task type and associates various data artefacts with corresponding models. Specifically, each model is represented as a domain vector with model quality, which enables models with high reuse potential for a training task to be selected efficiently. Then, the training options are stored as a configuration file, and associated with the model together with training data, developing data and test data, etc.. This supports access to various data artefacts through the model name. For models trained in the OLML database, they will be stored in the system by default. For available models out of the system, they could be converted into UER format and imported into the system.

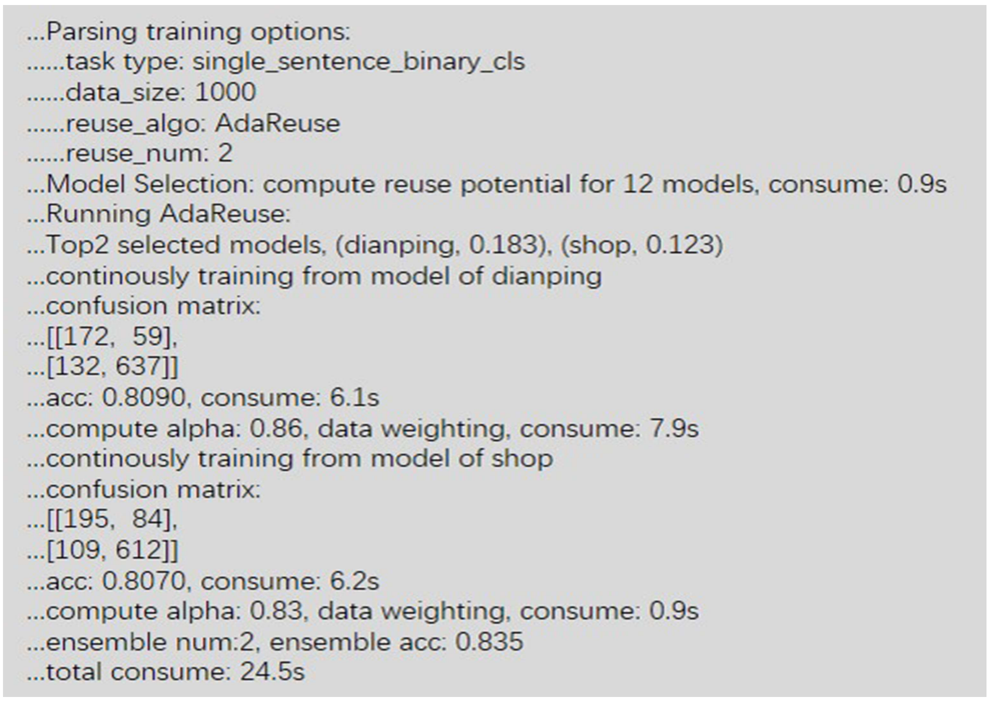
The benefit of such a layered design is that each layer provides APIs with clear semantics to its upper layer and could be extended independently. For example, if we need to add new data selection algorithms in the training layer, only slight changes need to be done in other layers, such as adding "data_selection" option in the language layer. Consequently, the prototype system could be extended to store and reuse more types of data artefacts.

5.2 Usability Study

We study the implementation of the training task in Fig.3(b). Fig.4 demonstrates the result. The system firstly computes reuse potential scores for the trained models, where the computation only takes less than 1 second for 12 trained models. Then, the training plan will be constructed by the AdaReuse algorithm with the models of dianping task and shop task selected as they have the maximum $P(T|M)$ scores. After that, the iterative training of AdaReuse starts. The training for each selected model takes about six seconds as the training data size is small. Then, key information, such as confusion matrix, accuracy and contribution ratio, will be computed. When the iterative training finishes, the best base model number is searched and the accuracy of ensemble is shown. The whole training process takes about 24.5 seconds on one Nvidia V100 GPU, and achieves about 0.835 accuracy for the hotel task. And as stated above, the continuously trained models from the dianping task and the shop task will be saved in the storage layer as models for the hotel task. The domain vectors have been computed and model quality will be computed offline immediately in the background. In this way, these trained models could be reused in new tasks quickly.

6 Conclusions

This paper proposed the OLML database which stores trained models and reuses these models in a new training task to improve the training effect. The AdaReuse algorithm was developed in the OLML



```
...Parsing training options:
......task type: single_sentence_binary_cls
......data_size: 1000
......reuse_algo: AdaReuse
......reuse_num: 2
...Model Selection: compute reuse potential for 12 models, consume: 0.9s
...Running AdaReuse:
...Top2 selected models, (dianping, 0.183), (shop, 0.123)
...continously training from model of dianping
...confusion matrix:
...[[172,  59],
...[132, 637]]
...acc: 0.8090, consume: 6.1s
...compute alpha: 0.86, data weighting, consume: 7.9s
...continously training from model of shop
...confusion matrix:
...[[195,  84],
...[109, 612]]
...acc: 0.8070, consume: 6.2s
...compute alpha: 0.83, data weighting, consume: 0.9s
...ensemble num:2, ensemble acc: 0.835
...total consume: 24.5s
```

Fig.4. Implementation details of Fig.3(b).

database to support efficient model selection and effective model reuse. The experimental results demonstrated that AdaReuse could significantly improve the training effect for sentiment analysis and name entity recognition tasks when the training data size is small. Meanwhile, the usability studies illustrated that the prototype system could properly store the trained models and form a model training-and-reuse cycle.

As the experiments conducted in this paper are all based on NLP tasks, how to support machine learning tasks of other modal in the OLML database could be studied in the future work.

References

[1] Yosinski J, Clune J, Bengio Y, Lipson H. How transferable are features in deep neural networks? arXiv:1411.1792, 2014. https://arxiv.org/abs/1411.1792, Nov. 2020.

[2] Wang W, Wang S, Gao J, Zhang M, Chen G, Ng T K, Ooi B C. Rafiki: Machine learning as an analytics service system. arXiv:1804.06087, 2018. https://arxiv.org/abs/1804.06087, Apr. 2021.

[3] Zhang W, Jiang J, Shao Y, Cui B. Efficient diversity-driven ensemble for deep neural networks. In *Proc. the 36th IEEE International Conference on Data Engineering*, Apr. 2020, pp.73-84. DOI: 10.1109/ICDE48307.2020.00014.

[4] Derakhshan B, Mahdiraji A R, Abedjan Z, Rabl T, Markl V. Optimizing machine learning workloads in collaborative environments. In *Proc. the 2020 ACM SIGMOD International Conference on Management of Data*, Jun. 2020, pp.1701-1716. DOI: 10.1145/3318464.3389715.

[5] Schapire R E. Explaining AdaBoost. In *Empirical Inference*, Schölkopf B, Luo Z, Vovk V (eds.), Springer, 2013, pp.37-52. DOI: 10.1007/978-3-642-41136-6_5.

[6] Zhao Z, Chen H, Zhang J, Zhao X, Liu T, Lu W, Chen X, Deng H, Ju Q, Du X. UER: An open-source toolkit for pre-training models. arXiv:1909.05658, 2019. https://arxiv.org/abs/1909.05658, April 2021.

[7] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv:1301.3781, 2013. https://arxiv.org/abs/1301.3781, Jan. 2021.

[8] Pennington J, Socher R, Manning C D. Glove: Global vectors for word representation. In *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing*, Oct. 2014, pp.1532-1543. DOI: 10.3115/v1/D14-1162.

[9] Zhao Z, Liu T, Li S, Li B, Du X. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *Proc. the 2017 Conference on Empirical Methods in Natural Language Processing*, Sept. 2017, pp.244-253. DOI: 10.18653/v1/D17-1023.

[10] Dai A M, Olah C, Le Q V. Document embedding with paragraph vectors. arXiv:1507.07998, 2015. https://arxiv.org/abs/1507.07998, April 2021.

[11] Axelrod A, He X, Gao J. Domain adaptation via pseudo in-domain data selection. In *Proc. the 2011 Conference on Empirical Methods in Natural Language Processing*, Jul. 2011, pp.355-362.

[12] Chen B, Huang F. Semi-supervised convolutional networks for translation adaptation with tiny amount of in-domain data. In *Proc. the 20th SIGNLL Conference on Computational Natural Language Learning*, Aug. 2016, pp.314-323. DOI: 10.18653/v1/K16-1031.

[13] Pan S J, Yang Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010, 22(10):1345-1359. DOI: 10.1109/TKDE.2009.191.

[14] Freitag M, Al-Onaizan Y. Fast domain adaptation for neural machine translation. arXiv:1612.06897, 2016. https://arxiv.org/abs/1612.06897, Dec. 2020.

[15] Zhang C, Kumar A, Ré C. Materialization optimizations for feature selection workloads. *ACM Transactions on Database Systems*, 2016, 41(1): Article No. 2. DOI: 10.1145/2877204.

[16] Nguyen C, Hassner T, Seeger M, Archambeau C. LEEP: A new measure to evaluate transferability of learned representations. In *Proc. the 37th International Conference on Machine Learning*, July 2020, pp.7294-7305.

[17] Dietterich T G. Ensemble methods in machine learning. In *Proc. the 1st International Workshop on Multiple Classifier Systems*, Jun. 2000, pp.1-15. DOI: 10.1007/3-540-45014-9_1.

[18] Fu F, Jiang J, Shao Y, Cui B. An experimental evaluation of large scale GBDT systems. *Proceedings of the VLDB Endowment*, 2019, 12(11): 1357-1370. DOI: 10.14778/3342263.3342273.

[19] Breiman L. Stacked regressions. *Machine Learning*, 1996, 24(1): 49-64. DOI: 10.1023/A:1018046112532.

[20] Ding Y X, Zhou Z H. Boosting-based reliable model reuse. In *Proc. the 12th Asian Conference on Machine Learning*, November 2020, pp.145-160.

[21] Miao H, Li A, Davis LS, Deshpande A. ModelHub: Towards unified data and lifecycle management for deep learning. arXiv:1611.06224, 2016. https://arxiv.org/abs/1611.06224, Nov. 2020.

[22] Vartak M, Subramanyam H, Lee W E, Viswanathan S, Husnoo S, Madden S, Zaharia M. MODELDB: A system for machine learning model management. In *Proc. the Workshop on Human-in-the-Loop Data Analytics*, June 26–July 1, 2016, Article No. 14. DOI: 10.1145/2939502.2939516.

[23] Bhardwaj A, Bhattacherjee S, Chavan A, Deshpande A, Elmore A J, Madden S, Parameswaran A G. Datahub: Collaborative data science & dataset version management at scale. arXiv:1409.0798, 2014. https://arxiv.org/abs/1409.0798, April 2021.

[24] Kraska T, Talwalkar A, Duchi J C, Griffith R, Franklin M J, Jordan M I. *MLbase*: A distributed machine-learning system. In *Proc. the 6th Biennial Conference on Innovative Data Systems Research*, Jan. 2013.

[25] Xin D, Ma L, Liu J, Macke S, Song S, Parameswaran A. HELIX: Accelerating human-in-the-loop machine learning. arXiv:1808.01095, 2018. https://arxiv.org/abs/1808.01095, April 2021.

[26] Xu L, Dong Q, Liao Y, Yu C, Tian Y, Liu W, Li L, Liu C, Zhang X. CLUENER2020: Fine-grained named entity recognition dataset and benchmark for Chinese. arXiv:2001.04351, 2020. https://arxiv.org/abs/2001.04351, Jan. 2021.

[27] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L, Polosukhin I. Attention is all you need. arXiv:1706.03762, 2017. https://arxiv.org/abs/1706.03762, April 2021.

[28] Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692, 2019. https://arxiv.org/abs/1907.11692, April 2021.

Jian-Wei Cui is currently a Ph.D. candidate in the Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education, and School of Information at Renmin University of China, Beijing. His research interests include natural language processing, machine translation and DB4AI. He is a member of CCF.

Wei Lu is currently an associate professor in the Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education, and School of Information, Renmin University of China, Beijing. He received his Ph.D. degree in computer science from Renmin University of China, Beijing, in 2011. His research interests include query processing in the context of spatiotemporal, cloud database systems and applications. He is a member of CCF.

Xin Zhao received her B.S. degree in computer science from Renmin University of China, Beijing, in 2018. She is currently a postgraduate in Renmin University of China, Beijing, and was an intern with the Tencent Technology and Engineering Group (TEG), Beijing.

Xiao-Yong Du is a professor in the Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education, and School of Information at Renmin University of China, Beijing. He received his Ph.D. degree in computer science from Nagoya Institute of Technology, Nagoya, in 1997. His research focuses on intelligent information retrieval, high-performance database and unstructured data management. He is a fellow of CCF.