Diverse Few-Shot Text Classification with Multiple Metrics

Mo Yu* Xiaoxiao Guo* Jinfeng Yi* Shiyu Chang
Saloni Potdar Yu Cheng Gerald Tesauro Haoyu Wang Bowen Zhou

AI Foundations – Learning, IBM Research
IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

Abstract

We study few-shot learning in natural language domains. Compared to many existing works that apply either metric-based or optimization-based meta-learning to image domain with low inter-task variance, we consider a more realistic setting, where tasks are diverse. However, it imposes tremendous difficulties to existing state-of-the-art metric-based algorithms since a single metric is insufficient to capture complex task variations in natural language domain. To alleviate the problem, we propose an adaptive metric learning approach that automatically determines the best weighted combination from a set of metrics obtained from meta-training tasks for a newly seen few-shot task. Extensive quantitative evaluations on real-world sentiment analysis and dialog intent classification datasets demonstrate that the proposed method performs favorably against state-of-the-art few shot learning algorithms in terms of predictive accuracy. We make our code and data available for further study.[1]

1 Introduction

Few-shot learning (FSL) (Miller et al., 2000; Li et al., 2006; Lake et al., 2015) aims to learn classifiers from few examples per class. Recently, deep learning has been successfully exploited for FSL via learning meta-models from a large number of **meta-training tasks**. These meta-models can be then used for rapid-adaptation for the **target/meta-testing tasks** that only have few training examples. Examples of such meta-models include: (1) metric-/similarity-based models, which learn contextual, and task-specific similarity measures (Koch, 2015; Vinyals et al., 2016; Snell et al., 2017); and (2) optimization-based models, which receive the input of gradients from a FSL task and predict either model parameters or parameter updates (Ravi and Larochelle, 2017; Munkhdalai and Yu, 2017; Finn et al., 2017; Wang et al., 2017).

In the past, FSL has mainly considered image domains, where all tasks are often sampled from one huge collection of data, such as Omniglot (Lake et al., 2011) and ImageNet (Vinyals et al., 2016), making tasks come from a single domain thus related. Due to such a simplified setting, almost all previous works employ a common meta-model (metric-/optimization-based) for all few-shot tasks. However, this setting is far from the realistic scenarios in many real-world applications of few-shot text classification. For example, on an enterprise AI cloud service, many clients submit various tasks to train text classification models for business-specific purposes. The tasks could be classifying customers' comments or opinions on different products/services, monitoring public reactions to different policy changes, or determining users' intents in different types of personal assistant services. As most of the clients cannot collect enough data, their submitted tasks form a few-shot setting. Also, these tasks are significantly diverse, thus a common metric is insufficient to handle all these tasks.

We consider a more realistic FSL setting in this paper, where tasks are diverse. In such a scenario, the optimal meta-model may vary across tasks. Our solution is based on the metric-learning approach (Snell et al., 2017) and the key idea is to maintain multiple metrics for FSL. The meta-learner selects and combines multiple metrics for learning the target task using **task clustering** on the meta-training tasks. During the meta-training, we propose to first partition the meta-training tasks into clusters, making the tasks in each cluster

*Equal contributions from the corresponding authors: yum@us.ibm.com, xiaoxiao.guo@ibm.com, jinfengy@us.ibm.com.

[1] https://github.com/Gorov/DiverseFewShot_Amazon

likely to be related. Then within each cluster, we train a deep embedding function as the metric. This ensures the common metric is only shared across tasks within the same cluster. Further, during meta-testing, each target FSL task is assigned to a task-specific metric, which is a linear combination of the metrics defined by different clusters. In this way, the diverse few-shot tasks can derive different metrics from the previous learning experience.

The key of the proposed FSL framework is the task clustering algorithm. Previous works (Kumar and Daume III, 2012; Kang et al., 2011; Crammer and Mansour, 2012; Barzilai and Crammer, 2015) mainly focused on convex objectives, and assumed the number of classes is the same across different tasks (*e.g.* binary classification is often considered). To make task clustering (i) compatible with deep networks and (ii) able to handle tasks with a various number of labels, we propose a **matrix-completion based task clustering** algorithm. The algorithm utilizes task similarity measured by cross-task transfer performance, denoted by matrix **S**. The (i, j)-entry of **S** is the estimated accuracy by adapting the learned representations on the i-th (source) task to the j-th (target) task. We rely on matrix completion to deal with missing and unreliable entries in **S** and finally apply spectral clustering to generate the task partitions.

To the best of our knowledge, our work is the first one addressing the diverse few-shot learning problem and reporting results on real-world few-shot text classification problems. The experimental results show that the proposed algorithm provides significant gains on few-shot sentiment classification and dialog intent classification tasks. It provides positive feedback on the idea of using multiple meta-models (metrics) to handle diverse FSL tasks, as well as the proposed task clustering algorithm on automatically detecting related tasks.

2 Problem Definition

Few-Shot Learning Since we focus on **diverse metric-based FSL**, the problem can be formulated in two stages: (1) **meta-training**, where a set of metrics $\mathcal{M} = \{\Lambda_1, \cdots, \Lambda_K\}$ is learned on the **meta-training tasks** $\mathcal{T}$. Each Λ_i maps two input (x_1, x_2) to a scalar of similarity score. Here $\mathcal{T} = \{\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_N\}$ is a collection of N tasks. Here K is a pre-defined number (usually $K \ll N$). Each task T_i consists of training, validation, and testing set denoted as $\left\{D_i^{train}, D_i^{valid}, D_i^{test}\right\}$, respectively. Note that the definition of $\mathcal{T}$ is a generalized version of $\mathcal{D}^{(meta-train)}$ in (Ravi and Larochelle, 2017), since each task T_i can be either few-shot (where D_i^{valid} is empty) or regular[2]. (2) **meta-testing**: the trained metrics in $\mathcal{M}$ is applied to **meta-testing tasks** denoted as $\mathcal{T}' = \{\mathrm{T'}_1, \cdots, \mathrm{T'}_{N'}\}$, where each $\mathrm{T'}_i$ is a few-shot learning task consisting of both training and testing data as $\left\{D_i'^{train}, D_i'^{test}\right\}$. $D_i'^{train}$ is a small labeled set for generating the prediction model M_i' for each T_i'. Specifically, M_i's are kNN-based predictors built upon the metrics in $\mathcal{M}$. We will detail the construction of M_i' in Section 3, Eq. (6). It is worth mentioning that the definition of $\mathcal{T}'$ is the same as $\mathcal{D}^{(meta-test)}$ in (Ravi and Larochelle, 2017). The **performance of few-shot learning** is the macro-average of M_i''s accuracy on all the testing set $D_i'^{test}$s.

Our definitions can be easily generalized to other meta-learning approaches (Ravi and Larochelle, 2017; Finn et al., 2017; Mishra et al., 2017). The motivation of employing multiple metrics is that when the tasks are diverse, one metric model may not be sufficient. Note that previous metric-based FSL methods can be viewed as a special case of our definition where $\mathcal{M}$ only contains a single Λ, as shown in the two base model examples below.

Base Model: Matching Networks In this paper we use the metric-based model Matching Network (MNet) (Vinyals et al., 2016) as the base metric model. The model (Figure 1b) consists of a neural network as the embedding function (**encoder**) and an augmented memory. The encoder, $f(\cdot)$, maps an input x to a d-length vector. The learned metric Λ is thus the similarity between the encoded vectors, $\Lambda(x_1, x_2) = f(x_1)^T f(x_2)$, i.e. the metric Λ is modeled by the encoder f. The augmented memory stores a support set $S = \{(x_i, y_i)\}_{i=1}^{|S|}$, where x_i is the supporting instance and y_i is its corresponding label in a one-hot format. The MNet explicitly defines a classifier M conditioned on the supporting set S. For any new data $\hat{x}$, M predicts its label via a similarity function $\alpha(.,.)$

[2]For example, the methods in (Triantafillou et al., 2017) can be viewed as training meta-models from any sampled batches from one single meta-training dataset.

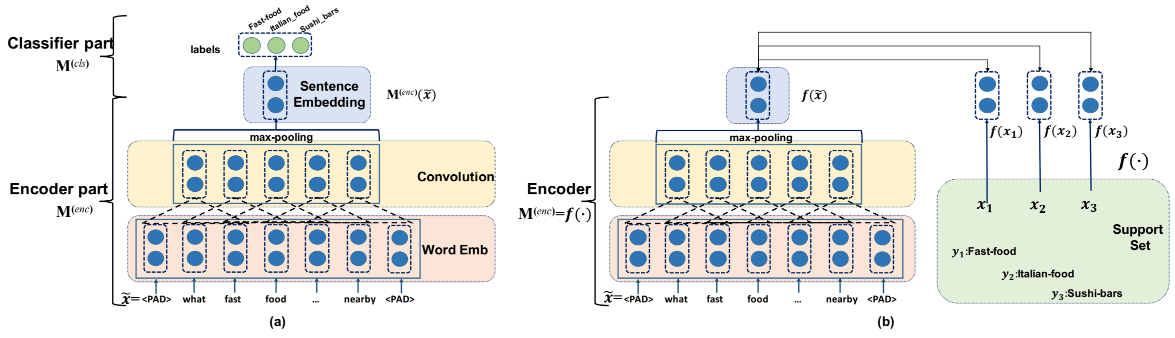


Figure 1: The Convolutional Neural Networks (CNN) used in this work: (a) A CNN classifier. The encoder component takes the sentence as input and outputs a fixed-length sentence embedding vector; the classifier component predicts class labels with the sentence embedding. (b) A Matching Network, which only contains an encoder like in (a), and makes prediction via a k-Nearest-Neighbor classifier with the similarity defined by the encoder.

between the test instance $\hat{x}$ and the support set S:

$$y = P(.|\hat{x}, S) = \sum_{i=1}^{|S|} \alpha(\hat{x}, x_i; \theta) y_i, \tag{1}$$

where we defined $\alpha(.,.)$ to be a softmax distribution given $\Lambda(\hat{x}, x_i)$, where x_i is a supporting instance, i.e., $\alpha(\hat{x}, x_i; \theta) = \exp(f(\hat{x})^T f(x_i)) / \sum_{j=1}^{|S|} \exp(f(\hat{x})^T f(x_j))$, where θ are the parameters of the encoder f. Thus, y is a valid distribution over the supporting set's labels $\{y_i\}_{i=1}^{|S|}$. To adapt the MNet to text classification, we choose encoder f to be a convolutional neural network (CNN) following (Kim, 2014; Johnson and Zhang, 2016). Figure 1 shows the MNet with the CNN architecture. Following (Collobert et al., 2011; Kim, 2014), the model consists of a convolution layer and a max-pooling operation over the entire sentence.

To train the MNets, we first sample the training dataset D for task T from all tasks $\mathcal{T}$, with notation simplified as $D \sim \mathcal{T}$. For each class in the sampled dataset D, we sample k random instances in that class to construct a support set S, and sample a batch of training instances B as training examples, i.e., $B, S \sim D$. The training objective is to minimize the prediction error of the training samples given the supporting set (with regard to the encoder parameters θ) as follows:

$$\mathop{\mathbb{E}}_{D\sim\mathcal{T}} \Big[\mathop{\mathbb{E}}_{B,S\sim D} \Big[\sum_{(x,y)\in B} \log(P(y|x, S; \theta)) \Big] \Big]. \tag{2}$$

Base Model: Prototypical Networks Prototypical Network (ProtoNet) (Snell et al., 2017) is a variation of Matching Network, which also depends on metric learning but builds the classifier M different from Eq. (1):

$$y = P(.|\hat{x}, S) = \sum_{i=1}^{L} \alpha(\hat{x}, S_i; \theta) y_i. \tag{3}$$

L is the number of classes and $S_i{=}\{x|(x, y) \in S \wedge y{=}y_i\}$ is the support set of class y_i. $\alpha(\hat{x}, S_i; \theta) = \exp\Big(f(\hat{x})^T \sum_{x \in S_i} f(x)\Big) / \sum_{j=1}^{L} \exp\Big(f(\hat{x})^T \sum_{x' \in S_j} f(x')\Big)$.

3 Methodology

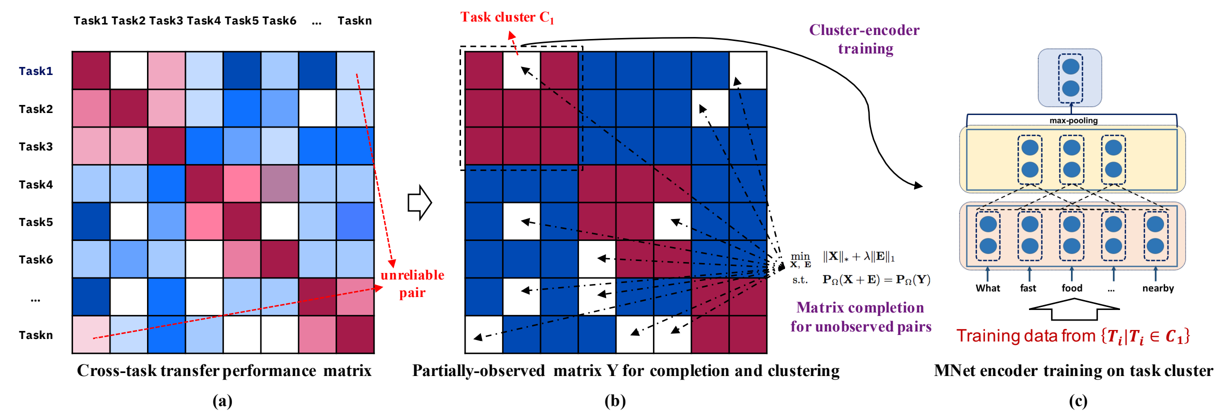
We propose a task-clustering framework to address the diverse few-shot learning problem stated in Section 2. We have the FSL algorithm summarized in Algorithm 1. Figure 2 gives an overview of our idea. The initial step of the algorithm is a novel task clustering algorithm based on matrix completion, which is described in Section 3.1. The few-shot learning method based on task clustering is then introduced in Section 3.2.

3.1 Robust Task Clustering by Matrix Completion

Our task clustering algorithm is shown in Algorithm 2. The algorithm first evaluates the transfer performance by applying a single-task model i to another task j (Section 3.1.1), which will result in a (partially observed) cross-task transfer performance matrix **S**. The matrix **S** is then cleaned and completed, giving a symmetry task similarity matrix **Y** for spectral clustering (Ng et al., 2002).

3.1.1 Estimation of Cross-Task Transfer Performance

Using single-task models, we can compute performance scores s_{ij} by adapting each M_i to each task



Task1 Task2 Task3 Task4 Task5 Task6 ... Taskn
Task1
Task2
Task3
Task4
Task5
Task6
...
Taskn
unreliable pair
Cross-task transfer performance matrix
(a)
Task cluster C_1
Cluster-encoder training
$\min_{X,E} \|X\|_* + \lambda\|E\|_1$ s.t. $P_\Omega(X+E) = P_\Omega(Y)$
Matrix completion for unobserved pairs
Partially-observed matrix Y for completion and clustering
(b)
max-pooling
What fast food ... nearby
Training data from $\{T_i | T_i \in C_1\}$
MNet encoder training on task cluster
(c)

Figure 2: Overview of the idea of our multi-metric learning approach for few-shot learning. (a) an illustration of the sparse cross-tasks transfer-performance matrix with unobserved entries (white blocks) and unreliable values (top-right and bottom-left corners), where red colors indicate positive transfer and blue colors indicate negative transfer; (b) the constructed binary partially-observed matrix with low-rank constraint for matrix completion and clustering (see Section 3.1 for the details); (c) an encoder trained with the matching network objective Eq. (2) on a task cluster (tasks 1, 2 and 3 in the example).

$T_j (j \neq i)$. This forms an $n \times n$ pair-wise classification performance matrix $\mathbf{S}$, called the *transfer-performance matrix*. Note that $\mathbf{S}$ is asymmetric since usually $\mathbf{S}_{ij} \neq \mathbf{S}_{ji}$.

Algorithm 1: RobustTC-FSL: Task Clustering for Few-Shot Learning

```
Input  : $N$ meta-training tasks $\mathcal{T}=\{\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_n\}$;
         number of clusters $K$; $N'$ target few-shot
         meta-testing tasks $\mathcal{T}'$
Output: Meta-model $\mathcal{M} = \{C_{1:K}$ ($K$ task clusters),
         $\mathcal{F} = \{f_1, f_2, \cdots, f_K\}$ ($K$ task encoders)$\}$ .
         One classifier $\mathrm{M}'_i$ for each target task $\mathrm{T}'$.
1 Robust Task Clustering: $C_{1:K}$ = RobustTC($\mathcal{T}$,$K$)
    (Algorithm 2)
2 Cluster-Model Training: Train one encoder (multi-task
    MNet) $f_i$ on each task cluster $C_i$ (Section 3.2.1)
3 Few-Shot Learning on Cluster-models: Train a model
    $\mathrm{M}_{trg}$ on task $\mathrm{T}_{trg}$ with the method in Section 3.2.2.
```

Ideally, the transfer performance could be estimated by training a MNet on task i and directly evaluating it on task j. However, the limited training data usually lead to generally low transfer performance of single-task MNet. As a result we adopt the following approach to estimate $\mathbf{S}$:

We train a CNN classifier (Figure 1(a)) on task i, then take only the encoder M_i^{enc} from M_i and freeze it to train a classifier on task j. This gives us a new task j model, and we test this model on D_j^{valid} to get the accuracy as the transfer-performance $\mathbf{S}_{ij}$. The score shows how the representations learned on task i can be adapted to task j, thus indicating the similarity between tasks.

Algorithm 2: RobustTC: Robust Task Clustering based on Matrix Completion

```
Input  : A set of $n$ tasks $\mathcal{T} = \{\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_n\}$,
         number of task clusters $K$
Output: $K$ task clusters $C_{1:K}$
1 Learning of Single-Task Models: train single-task
    models $\mathrm{M}_i$ for each task $\mathrm{T}_i$
2 Evaluation of Transfer-Performance Matrix: get
    performance matrix $\mathbf{S}$ (Section 3.1.1)
3 Score Filtering: Filter the uncertain scores in $\mathbf{S}$ and
    construct the symmetric matrix $\mathbf{Y}$ using Eq. (4)
4 Matrix Completion: Complete the similar matrix $\mathbf{X}$
    from $\mathbf{Y}$ using Eq. (5)
5 Task Clustering: $C_{1:K}$=SpectralClustering($\mathbf{X}$, $K$)
```

Remark: Out-of-Vocabulary Problem In text classification tasks, transferring an encoder with fine-tuned word embeddings from one task to another is difficult as there can be a significant difference between the two vocabularies. Hence, while learning the single-task CNN classifiers, we always make the word embeddings fixed.

3.1.2 Task Clustering Method

Directly using the transfer performance for task clustering may suffer from both efficiency and accuracy issues. First, evaluation of all entries in the matrix $\mathbf{S}$ involves conducting the source-target transfer learning $O(n^2)$ times, where n is the number of meta-training tasks. For a large number of diverse tasks where the n can be larger than 1,000, evaluation of the full matrix is unacceptable (over 1M entries to evaluate). Second, the estimated cross-task performance (i.e. some $\mathbf{S}_{ij}$ or $\mathbf{S}_{ji}$ scores) is often unreliable due to small data size or label noise. When the number of the uncertain values is large, they can collectively mislead the clustering algorithm to output an incorrect task-partition. To address the aforementioned challenges, we propose a novel task clustering algorithm based on the theory of matrix completion (Candès and Tao, 2010). Specifically, we deal with the huge number of entries by randomly sample task pairs to evaluate the $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ scores. Besides, we deal with the unreliable entries and asymmetry issue by keeping only task pairs (i, j) with consistent $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ scores. as will be introduced in Eq. (4). Below, we describe our method in detail.

Score Filtering First, we use only reliable task pairs to generate a *partially-observed* similarity matrix $\mathbf{Y}$. Specifically, if $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ are high enough, then it is likely that tasks $\{i, j\}$ belong to a same cluster and share significant information. Conversely, if $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ are low enough, then they tend to belong to different clusters. To this end, we need to design a mechanism to determine if a performance is high or low enough. Since different tasks may vary in difficulty, a fixed threshold is not suitable. Hence, we define a dynamic threshold using the mean and standard deviation of the target task performance, i.e., $\mu_j = \text{mean}(\mathbf{S}_{:j})$ and $\sigma_j = \text{std}(\mathbf{S}_{:j})$, where $\mathbf{S}_{:j}$ is the j-th column of $\mathbf{S}$. We then introduce two positive parameters p_1 and p_2, and define high and low performance as $\mathbf{S}_{ij}$ greater than $\mu_j + p_1\sigma_j$ or lower than $\mu_j - p_2\sigma_j$, respectively. When both $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ are high and low enough, we set their pairwise similarity as 1 and 0, respectively. Other task pairs are treated as uncertain task pairs and are marked as unobserved, and don't influence our clustering method. This leads to a partially-observed symmetric matrix $\mathbf{Y}$, i.e.,

$$\mathbf{Y}_{ij}{=}\mathbf{Y}_{ji}{=}\begin{cases} 1 & \begin{array}{l}\text{if } \mathbf{S}_{ij} > \mu_j + p_1\sigma_j \\ \text{and } \mathbf{S}_{ji} > \mu_i + p_1\sigma_i\end{array} \\ 0 & \begin{array}{l}\text{if } \mathbf{S}_{ij} < \mu_j - p_2\sigma_j \\ \text{and } \mathbf{S}_{ji} < \mu_i - p_2\sigma_i\end{array} \\ \text{unobserved} & \text{otherwise}\end{cases} \tag{4}$$

Matrix Completion Given the partially observed matrix $\mathbf{Y}$, we then reconstruct the full similarity matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$. We first note that the similarity matrix $\mathbf{X}$ should be of low-rank (proof deferred to appendix). Additionally, since the observed entries of $\mathbf{Y}$ are generated based on high and low enough performance, it is safe to assume that most observed entries are correct and only a few may be incorrect. Therefore, we introduce a sparse matrix $\mathbf{E}$ to capture the observed incorrect entries in $\mathbf{Y}$. Combining the two observations, $\mathbf{Y}$ can be decomposed into the sum of two matrices $\mathbf{X}$ and $\mathbf{E}$, where $\mathbf{X}$ is a low rank matrix storing similarities between task pairs, and $\mathbf{E}$ is a sparse matrix that captures the errors in $\mathbf{Y}$. The matrix completion problem can be cast as the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{X},\,\mathbf{E}} \quad & \|\mathbf{X}\|_* + \lambda\|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{P}_\Omega(\mathbf{X} + \mathbf{E}) = \mathbf{P}_\Omega(\mathbf{Y}), \end{aligned} \tag{5}$$

where $\| \circ \|_*$ denotes the matrix nuclear norm, the convex surrogate of rank function. Ω is the set of observed entries in $\mathbf{Y}$, and $\mathbf{P}_\Omega : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ is a matrix projection operator defined as

$$[\mathbf{P}_\Omega(\mathbf{A})]_{ij} = \begin{cases} \mathbf{A}_{ij} & \text{if } (i, j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

Finally, we apply spectral clustering on the matrix $\mathbf{X}$ to get the task clusters.

Remark: Sample Efficiency In the Appendix A, we show a Theorem 7.1 as well as its proof, implying that under mild conditions, the problem (5) can perfectly recover the underlying similarity matrix $\mathbf{X}^*$ if the number of observed correct entries is at least $O(n \log^2 n)$. This theoretical guarantee implies that for a large number n of training tasks, only a tiny fraction of all task pairs is needed to reliably infer similarities over all task pairs.

3.2 Few-Shot Learning with Task Clusters

3.2.1 Training Cluster Encoders

For each cluster C_k, we train a multi-task MNet model (Figure 1(b)) with all tasks in that cluster to encourage parameter sharing. The result, denoted as f_k is called the **cluster-encoder** of cluster C_k. The k-th metric of the cluster is thus $\Lambda(x_1, x_2) = f_k(x_1)^\intercal f_k(x_2)$.

3.2.2 Adapting Multiple Metrics for Few-Shot Learning

To build a predictor M with access to only a limited number of training samples, we make the prediction probability by linearly combining prediction from learned cluster-encoders:

$$p(y|x) = \sum_k \alpha_k P(y|x; f_k). \tag{6}$$

where f_k is the learned (and frozen) encoder of the k-th cluster, $\{\alpha_k\}_{k=1}^K$ are adaptable parameters trained with few-shot training examples. And the predictor $P(y|x; f_k)$ from each cluster is

$$P(y = y_l|x; f_k) = \frac{\exp\{f_k(x_l)^\intercal f_k(x)\}}{\sum_i \exp\{f_k(x_i)^\intercal f_k(x)\}} \quad (7)$$

x_l is the corresponding training sample of label y_l.

Remark: Joint Method versus Pipeline Method End-to-end joint optimization on training data becomes a popular methodology for deep learning systems, but it is not directly applicable to diverse FSL. One main reason is that deep networks could easily fit any task partitions if we optimize on training loss only, making the learned metrics not generalize, as discussed in Section 6. As a result, this work adopts a pipeline training approach and employing validation sets for task clustering. Combining reinforcement learning with meta-learning could be a potential solution to enable an end-to-end training for future work.

4 Tasks and Data Sets

We test our methods by conducting experiments on two text classification data sets. We used NLTK toolkit[3] for tokenization. The task are divided into meta-training tasks and meta-testing tasks (target tasks), where the meta-training tasks are used for clustering and cluster-encoder training. The meta-testing tasks are few-shot tasks, which are used for evaluating the method in Eq. (6).

4.1 Amazon Review Sentiment Classification

First, following Barzilai and Crammer (2015), we construct multiple tasks with the multi-domain sentiment classification (Blitzer et al., 2007) data set. The dataset consists of Amazon product reviews for 23 types of products (see Appendix D for the details). For each product domain, we construct three binary classification tasks with different thresholds on the ratings: the tasks consider a review as positive if it belongs to one of the following buckets $=5$ stars, $>=4$ stars or $>=2$ stars.[4] These buckets then form the basis of the task-setup, giving us 23 $\times$ 3=69 tasks in total. For each domain we distribute the reviews uniformly to the 3 tasks. For evaluation, we select 12 (4$\times$3) tasks from 4 domains (*Books, DVD, Electronics, Kitchen*) as the meta-testing (target) tasks out of all 23 domains. For the target tasks, we create 5-shot learning problems.

4.2 Real-World Tasks: User Intent Classification for Dialog System

The second dataset is from an online service which trains and serves intent classification models to various clients. The dataset comprises recorded conversations between human users and dialog systems in various domains, ranging from personal assistant to complex service-ordering or customer-service request scenarios. During classification, intent-labels[5] are assigned to user utterances (sentences). We use a total of 175 tasks from different clients, and randomly sample 10 tasks from them as our target tasks. For each meta-training task, we randomly sample 64% data into a training set, 16% into a validation set, and use the rest as the test set. The number of labels for these tasks varies a lot (from 2 to 100, see Appendix D for details), making regular k-shot settings not essentially limited-resource problems (e.g., 5-shot on 100 classes will give a good amount of 500 training instances). Hence, to adapt this to a FSL scenario, for target tasks we keep one example for each label (one-shot), plus 20 randomly picked labeled examples to create the training data. We believe this is a fairly realistic estimate of labeled examples one client could provide easily.

Remark: Evaluation of the Robustness of Algorithm 2 Our matrix-completion method could handle a large number of tasks via task-pair sampling. However, the sizes of tasks in the above two few-shot learning datasets are not too huge, so evaluation of the whole task-similarity matrix is still tractable. In our experiments, the incomplete matrices mainly come from the score-filtering step (see Eq. 4). Thus there is limited randomness involved in the generation of task clusters.

To strengthen the conclusion, we evaluate our algorithm on an additional dataset with a much larger number of tasks. The results are reported in the multi-task learning setting instead of the few-shot learning setting focused in this paper. Therefore we put the results to a non-archive version of

[3]http://www.nltk.org/

[4]Data downloaded from http://www.cs.jhu.edu/~mdredze/datasets/sentiment/, in which the 3-star samples were unavailable due to their ambiguous nature (Blitzer et al., 2007).

[5]In conversational dialog systems, intent-labels are used to guide the dialog-flow.

this paper[6] for further reference.

5 Experiments

5.1 Experiment Setup

Baselines We compare our method to the following baselines: (1) **Single-task CNN**: training a CNN model for each task individually; (2) **Single-task FastText**: training one FastText model (Joulin et al., 2016) with fixed embeddings for each individual task; (3) **Fine-tuned the holistic MTL-CNN**: a standard transfer-learning approach, which trains one MTL-CNN model on all the training tasks offline, then fine-tunes the classifier layer (i.e. $\mathrm{M}^{(cls)}$ Figure 1(a)) on each target task; (4) **Matching Network**: a metric-learning based few-shot learning model trained on all training tasks; (5) **Prototypical Network**: a variation of matching network with different prediction function as Eq. 3; (6) **Convex combining all single-task models**: training one CNN classifier on each meta-training task individually and taking the encoder, then for each target task training a linear combination of all the above single-task encoders with Eq. (6). This baseline can be viewed as a variation of our method without task clustering. We initialize all models with pre-trained 100-dim Glove embeddings (trained on 6B corpus) (Pennington et al., 2014).

Hyper-Parameter Tuning In all experiments, we set both p_1 and p_2 parameters in (4) to 0.5. This strikes a balance between obtaining enough observed entries in $\mathbf{Y}$, and ensuring that most of the retained similarities are consistent with the cluster membership. The window/hidden-layer sizes of CNN and the initialization of embeddings (random or pre-trained) are tuned during the cluster-encoder training phase, with the validation sets of meta-training tasks. We have the CNN with window size of 5 and 200 hidden units. The single-metric FSL baselines have 400 hidden units in the CNN encoders. On sentiment classification, all cluster-encoders use random initialized word embeddings for sentiment classification, and use Glove embeddings as initialization for intent classification, which is likely because the training sets of the intent tasks are usually small.

Since all the sentiment classification tasks are binary classification based on our dataset construction. A CNN classifier with binary output layer can be also trained as the cluster-encoder for each task cluster. Therefore we compared CNN classifier, matching network, and prototypical network on Amazon review, and found that CNN classifier performs similarly well as prototypical network. Since some of the Amazon review data is quite large which involves further difficulty on the computation of supporting sets, we finally use binary CNN classifiers as cluster-encoders in all the sentiment classification experiments.

Selection of the learning rate and number of training epochs for FSL settings, i.e., fitting αs in Eq. (6), is more difficult since there is no validation data in few-shot problems. Thus we pre-select a subset of meta-training tasks as meta-validation tasks and tune the two hyper-parameters on the meta-validation tasks.

5.2 Experimental Results

Table 1 shows the main results on (i) the 12 few-shot product sentiment classification tasks by leveraging the learned knowledge from the 57 previously observed tasks from other product domains; and (ii) the 10 few-shot dialog intent classification tasks by leveraging the 165 previously observed tasks from other clients' data.

Due to the limited training resources, all the supervised-learning baselines perform poorly. The two state-of-the-art metric-based FSL approaches, matching network (4) and prototypical network (5), do not perform better compared to the other baselines, since the single metric is not sufficient for all the diverse tasks. On intent classification where tasks are further diverse, all the single-metric or single-model methods (3-5) perform worse compared to the single-task CNN baseline (1). The convex combination of all the single training task models is the best performing baseline overall. However, on intent classification it only performs on par with the single-task CNN (1), which does not use any meta-learning or transfer learning techniques, mainly for two reasons: (i) with the growth of the number of meta-training tasks, the model parameters grow linearly, making the number of parameters (165 in this case) in Eq.(6) too large for the few-shot tasks to fit; (ii) the meta-training tasks in intent classification usually contain less training data, making the single-task encoders not generalize well.

In contrast, our RobustTC-FSL gives consistently better results compared to all the baselines.

[6] https://arxiv.org/pdf/1708.07918.pdf

Model	Avg Acc	
	Sentiment	**Intent**
(1) Single-task CNN w/pre-trained emb	65.92	34.46
(2) Single-task FastText w/pre-trained emb	63.05	23.87
(3) Fine-tuned holistic MTL-CNN	76.56	30.36
(4) Matching Network (Vinyals et al., 2016)	65.73	30.42
(5) Prototypical Network (Snell et al., 2017)	68.15	31.51
(6) Convex combination of all single-task models	78.85	34.43
RobustTC-FSL	**83.12**	**37.59**
Adaptive RobustTC-FSL	-	**42.97**

Table 1: Accuracy of FSL on sentiment classification (Sentiment) and dialog intent classification (Intent) tasks. The target tasks of sentiment classification are 5-shot ones; and each intent target task contains one training example per class and 20 random labeled examples.

It outperforms the baselines in previous work (1-5) by a large margin of more than 6% on the sentiment classification tasks, and more than 3% on the intent classification tasks. It is also significantly better than our proposed baseline (6), showing the advantages of the usage of task clustering.

Adaptive RobustTC-FSL Although the RobustTC-FSL improves over baselines on intent classification, the margin is smaller compared to that on sentiment classification, because the intent classification tasks are more diverse in nature. This is also demonstrated by the training accuracy on the target tasks, where several tasks fail to find any cluster that could provide a metric that suits their training examples. To deal with this problem, we propose an improved algorithm to automatically discover whether a target task belongs to none of the task-clusters. If the task doesn't belong to any of the clusters, it cannot benefit from any previous knowledge thus falls back to single-task CNN. The target task is treated as "out-of-clusters" when none of the clusters could achieve higher than 20% accuracy (selected on meta-validation tasks) on its training data. We call this method **Adaptive RobustTC-FSL**, which gives more than 5% performance boost over the best RobustTC-FSL result on intent classification. Note that the adaptive approach makes no difference on the sentiment tasks, because they are more closely related so re-using cluster-encoders always achieves better results compared to single-task CNNs.

5.3 Analysis

Effect of the number of clusters Figure 3 shows the effect of cluster numbers on the two tasks. RobustTC achieves best performance with 5 clusters on sentiment analysis (SA) and 20 clusters on intent classification (Intent). All clustering results significantly outperform the single-metric baselines (#cluster=1 in the figure).

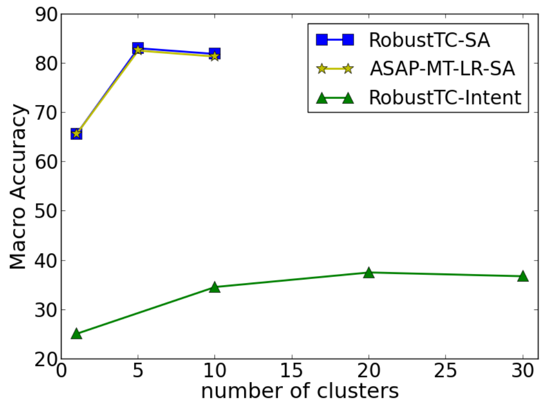


Figure 3: Effect of clusters. RobustTC-SA and RobustTC-Intent: the performance of our RobustTC clusters on the sentiment and intent classification tasks. ASAP-MT-LR-SA: the state-of-the-art ASAP-MT-LR clusters on the sentiment-analysis tasks (the method is not applicable to the intent-classification tasks).

Effect of the clustering algorithms Compared to previous task clustering algorithms, our RobustTC is the only one that can cluster tasks with varying numbers of class labels (e.g. in intent classification tasks). Moreover, we show that even in the setting of all binary classifications tasks (e.g. the sentiment-analysis tasks) that previous task clustering research work on, our RobustTC is still slightly better for the diverse FSL problems. Figure 3 compares with a state-of-the-art logistic regression based task clustering method (**ASAP-MT-LR**) (Barzilai and Crammer, 2015). Our RobustTC clusters give slightly better FSL performance (e.g. 83.12 vs. 82.65 when #cluster=5).

	Clus0	**Clus1**	**Clus2**	**Clus3**	**Clus4**	**Clus5**	**Clus6**	**Clus7**	**Clus8**	**Clus9**
	automotive.t2 camera.t2 health.t2 magazines.t2 office.t2 outdoor.t2 sports.t2 sports.t4	apparel.t2 automotive.t4 baby.t2 cell.t2 computer.t2 computer.t4 computer.t5 jewelry.t4 music.t2 video.t2	baby.t5 magazines.t5 sports.t5 toys.t5 video.t5	automotive.t5 baby.t4 health.t4 health.t5	apparel.t5 camera.t5 grocery.t5 jewelry.t5	beauty.t4 beauty.t5 cell.t5 gourmet.t2 gourmet.t4 grocery.t2 grocery.t4 office.t4 outdoor.t4	camera.t4 software.t2 software.t4	gourmet.t5 magazines.t4 music.t4 music.t5 video.t4	cell.t4 software.t5 toys.t4	apparel.t4 toys.t2
dvd-t4	0.4844	0.4416	0.4625	**0.7843**	**0.7970**	0.7196	**0.8952**	0.3763	0.7155	0.6315
dvd-t5	0.0411	-0.2493	**0.5037**	**0.3567**	0.1686	-0.0355	**0.4150**	-0.2603	-0.0867	0.0547
kitchen-t4	0.6823	0.7268	0.7929	**1.2660**	**1.1119**	0.7255	**1.2196**	0.7065	0.6625	1.0945

Table 2: Visualization of clusters on the Amazon review domain. The top shows the training tasks assigned to the 10 clusters. Here the number N$\in \{2,4,5\}$ refers to the threshold of stars for positive reviews. At the bottom we show three tasks with largest improvement from RobustTC-FSL. The top-3 most relevant task clusters (i.e. with highest weights αs in Eq.6) are highlighted with **blue bold** font.

Visualization of Task Clusters The top rows of Table 2 shows the ten clusters used to generate the sentiment classification results in Figure 3. From the results, we can see that tasks with same thresholds are usually grouped together; and tasks in similar domains also tend to appear in the same clusters, even the thresholds are slightly different (e.g. t2 vs t4 and t4 vs t5).

The bottom of the table shows the weights αs in Eq.(6) for the target tasks with the largest improvement. It confirms that our RobustTC-FSL algorithm accurately adapts multiple metrics for the target tasks.

6 Related Work

Few Shot Learning FSL (Li et al., 2006; Miller et al., 2000) aims to learn classifiers for new classes with only a few training examples per class. Bayesian Program Induction (Lake et al., 2015) represents concepts as simple programs that best explain observed examples under a Bayesian criterion. Siamese neural networks rank similarity between inputs (Koch, 2015). Matching Networks (Vinyals et al., 2016) map a small labeled support set and an unlabeled example to its label, obviating the need for fine-tuning to adapt to new class types. These approaches essentially learn one metric for all tasks, which is sub-optimal when the tasks are diverse. An LSTM-based meta-learner (Ravi and Larochelle, 2017) learns the exact optimization algorithm used to train another learner neural-network classifier for the few-shot setting.

Previous FSL research usually adopts the k-shot, N-way setting, where all the few-shot tasks have the same number of N class labels, and each label has k training instances. Moreover, these few-shot tasks are usually constructed by sampling from one huge dataset, thus all the tasks are guaranteed to be related to each other. However, in real-world applications, the few-shot learning tasks could be diverse: there are different tasks with varying number of class labels and they are not guaranteed to be related to each other. As a result, a single meta-model or metric-model is usually not sufficient to handle all the few-shot tasks.

Task Clustering Previous task clustering methods measure the task relationships in terms of similarities among single-task model parameters (Kumar and Daume III, 2012; Kang et al., 2011); or jointly assign task clusters and train model parameters for each cluster to minimize the overall training loss (Crammer and Mansour, 2012; Barzilai and Crammer, 2015; Murugesan et al., 2017). These methods usually work on convex models but do not fit the deep networks, mainly because of (i) the parameters of deep networks are very high-dimensional and their similarities are not necessarily related to the functional similarities; and (ii) deep networks have flexible representation power so they may overfit to arbitrary cluster assignment if we consider training loss alone. Moreover, these methods require identical class label sets across different tasks, which does not hold in most of the realistic settings.

7 Conclusion

We propose a few-shot learning approach for diverse tasks based on task clustering. The proposed method can use multiple metrics, and performs significantly better compared to previous single-metric based methods when the few-shot tasks come from diverse domains. Future work includes generalizing our method to non-NLP problems, as well as applying the task-clustering idea to other few-shot learning frameworks (Ravi and Larochelle, 2017; Finn et al., 2017; Mishra et al., 2017; Cheng et al., 2017).

References

Aviad Barzilai and Koby Crammer. 2015. Convex multi-task learning by clustering. In *AISTATS*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447.

Emmanuel J Candès and Terence Tao. 2010. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080.

Venkat Chandrasekaran, Sujay Sanghavi, Pablo A Parrilo, and Alan S Willsky. 2011. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596.

Yu Cheng, Mo Yu, Xiaoxiao Guo, and Bowen Zhou. 2017. Few-shot learning with meta metric learners. In *NIPS 2017 Workshop on Meta-Learning*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Koby Crammer and Yishay Mansour. 2012. Learning multiple tasks using shared hypotheses. In *Advances in Neural Information Processing Systems*, pages 1475–1483.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using one-hot lstm for region embeddings. *stat*, 1050:7.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Zhuoliang Kang, Kristen Grauman, and Fei Sha. 2011. Learning with whom to share in multi-task feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 521–528.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Gregory Koch. 2015. *Siamese neural networks for one-shot image recognition*. Ph.D. thesis, University of Toronto.

Abhishek Kumar and Hal Daume III. 2012. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*.

Brenden M Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B Tenenbaum. 2011. One shot learning of simple visual concepts. In *CogSci*, volume 172, page 2.

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

Fei-Fei Li, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Erik G Miller, Nicholas E Matsakis, and Paul A Viola. 2000. Learning from one example through shared densities on transforms. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 464–471. IEEE.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2017. A simple neural attentive meta-learner. In *NIPS 2017 Workshop on Meta-Learning*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. *arXiv preprint arXiv:1703.00837*.

Keerthiram Murugesan, Jaime Carbonell, and Yiming Yang. 2017. Co-clustering for multitask learning. *arXiv preprint arXiv:1703.00994*.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, volume 1, page 6.

Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. 2017. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*, pages 2252–2262.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. 2017. Learning to model the tail. In *Advances in Neural Information Processing Systems 30*, pages 7032–7042.