ROBUST TASK CLUSTERING FOR DEEP AND DIVERSE MULTI-TASK AND FEW-SHOT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate task clustering for deep learning-based multi-task and few-shot learning in the settings with large numbers of tasks. Our method measures task similarities using cross-task transfer performance matrix. Although this matrix provides us critical information regarding similarities between tasks, the uncertain task-pairs, i.e., the ones with extremely asymmetric transfer scores, may collectively mislead clustering algorithms to output an inaccurate task-partition. Moreover, when the number of tasks is large, generating the full transfer performance matrix can be very time consuming. To overcome these limitations, we propose a novel task clustering algorithm to estimate the similarity matrix based on the theory of matrix completion. The proposed algorithm can work on partially-observed similarity matrices based on only sampled task-pairs with reliable scores, ensuring its efficiency and robustness. Our theoretical analysis shows that under mild assumptions, the reconstructed matrix perfectly matches the underlying "true" similarity matrix with an overwhelming probability. The final task partition is computed by applying an efficient spectral clustering algorithm to the recovered matrix. Our results show that the new task clustering method can discover task clusters that benefit both multi-task learning and few-shot learning setups for sentiment classification and dialog intent classification tasks.

1 INTRODUCTION

This paper leverages knowledge distilled from a large number of learning tasks (Barzilai & Crammer, 2015; van Seijen et al., 2017), or *MAny Task Learning (**MATL**)*, to achieve the goal of (i) improving the overall performance of all tasks, as in *multi-task learning (**MTL**)*; and (ii) rapid-adaptation to a new task by using previously learned knowledge, similar to *few-shot learning (**FSL**)* and transfer learning. Previous work on multi-task learning and transfer learning used small numbers of related tasks (usually $\sim$10) picked by human experts. By contrast, MATL tackles hundreds or thousands of tasks (Barzilai & Crammer, 2015; van Seijen et al., 2017), with unknown relatedness between pairs of tasks, introducing new challenges such as task diversity and model inefficiency.

MATL scenarios are increasingly common in a wide range of machine learning applications with potentially huge impact. Examples include reinforcement learning for game playing – where many numbers of sub-goals are treated as tasks by the agents for joint-learning, e.g. van Seijen et al. (2017) achieved the state-of-the-art on the Ms. Pac-Man game by using a multi-task learning architecture to approximate rewards of over 1,000 sub-goals (reward functions). Another important example is enterprise AI cloud services – where many clients submit various tasks/datasets to train machine learning models for business-specific purposes. The clients could be companies who want to know opinion from their customers on products and services, agencies that monitor public reactions to policy changes, and financial analysts who analyze news as it can potentially influence the stock-market. Such MATL-based services thus need to handle the diverse nature of clients' tasks.

Challenges on Handling Diverse (Heterogeneous) Tasks Previous multi-task learning and few-shot learning research usually work on homogeneous tasks, e.g. all tasks are binary classification problems, or tasks are close to each other (picked by human experts) so the positive transfer between tasks is guaranteed. However, with a large number of tasks in a MATL setting, the above assumption may not hold, i.e. we need to be able to deal with tasks with larger diversity. Such diversity can be reflected as (i) **tasks with varying numbers of labels**: when tasks are diverse, different tasks could

have different numbers of labels; and the labels might be defined in different label spaces without relatedness. Most of the existing multi-task and few-shot learning methods will fail in this setting; and more importantly (ii) **tasks with positive and negative transfers**: since tasks are not guaranteed to be similar to each other in the MATL setting, they are not always able to help each other when trained together, i.e. *negative transfer* (Yosinski et al., 2014) between tasks. For example, in dialog services, the sentences "*What fast food do you have nearby*" and "*Could I find any Indian food*" may belong to two different classes "*fast_food*" and "*indian_food*" for a restaurant recommendation service in a city; while for a travel-guide service for a park, those two sentences could belong to the same class "*food_options*". In this case the two tasks may hurt each other when trained jointly with a single representation function, since the first task turns to give similar representations to both sentences while the second one turns to distinguish them in the representation space.

A Task Clustering Based Solution To deal with the second challenge above, we propose to partition the tasks to clusters, making the tasks in each cluster more likely to be related. Common knowledge is only shared across tasks within a cluster, thus the negative transfer problem is alleviated. There are a few task clustering algorithm proposed mainly for convex models (Kumar & Daume III, 2012; Kang et al., 2011; Crammer & Mansour, 2012; Barzilai & Crammer, 2015), but they assume that the tasks have the same number of labels (usually binary classification). In order to handle tasks with varying numbers of labels, we adopt a similarity-based task clustering algorithm. The task similarity is measured by cross-task transfer performance, which is a matrix $\mathbf{S}$ whose (i, j)-entry $\mathbf{S}_{ij}$ is the estimated accuracy by adapting the learned representations on the i-th (source) task to the j-th (target) task. The above task similarity computation does not require the source task and target task to have the same set of labels, as a result, our clustering algorithm could naturally handle tasks with varying numbers of labels.

Although cross-task transfer performance can provide critical information of task similarities, directly using it for task clustering may suffer from both efficiency and accuracy issues. First and most importantly, 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 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 noises. 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 & 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; and deal with the unreliable entries by keeping only task pairs (i, j) with consistent $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ scores. Given a set of n tasks, we first construct an $n \times n$ partially-observed matrix $\mathbf{Y}$, where its observed entries correspond to the sampled and reliable task pairs (i, j) with consistent $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ scores. Otherwise, if the task pairs (i, j) are not sampled to compute the transfer scores or the scores are inconsistent, we mark both $\mathbf{Y}_{ij}$ and $\mathbf{Y}_{ji}$ as unobserved. Given the constructed partially-observed matrix $\mathbf{Y}$, our next step is to recover an $n \times n$ full similarity matrix using a robust matrix completion approach, and then generate the final task partition by applying spectral clustering to the completed similarity matrix. The proposed approach has a 2-fold advantage. First, our method carries a strong theoretical guarantee, showing that the full similarity matrix can be *perfectly* recovered if the number of observed correct entries in the partially observed similarity matrix is at least $O(n \log^2 n)$. This theoretical result allows us to only compute the similarities of $O(n \log^2 n)$ instead of $O(n^2)$ pairs, thus greatly reduces the computation when the number of tasks is large. Second, by filtering out uncertain task pairs, the proposed algorithm will be less sensitive to noise, leading to a more robust clustering performance.

The task clusters allow us to handle (i) diverse MTL problems, by model sharing only within clusters such that the negative transfer from irrelevant tasks can be alleviated; and (ii) diverse FSL problems, where a new task can be assigned a task-specific metric, which is a linear combination of the metrics defined by different clusters, such that the diverse few-shot tasks could derive different metrics from the previous learning experience. Our results show that the proposed task clustering algorithm, combined with the above MTL and FSL strategies, could give us significantly better deep MTL and FSL algorithms on sentiment classification and intent classification tasks.

2 RELATED WORK

Task/Dataset Clustering on Model Parameters This class of task clustering methods measure the task relationships in terms of model parameter similarities on individual tasks. Given the parameters of convex models, task clusters and cluster assignments could be derived via matrix decomposition (Kumar & Daume III, 2012) or k-means based approach (Kang et al., 2011). The parameter similarity based task clustering method for deep neural networks (Yang & Hospedales, 2016) applied low-rank tensor decomposition of the model layers from multiple tasks. This method is infeasible for our MATL setting because of its high computation complexity with respect to the number of tasks and its inherent requirement on closely related tasks because of its parameter-similarity based approach.

Task/Dataset Clustering with Clustering-Specific Training Objectives Another class of task clustering methods joint assign task clusters and train model parameters for each cluster that minimize training loss within each cluster by K-means based approach (Crammer & Mansour, 2012) or minimize overall training loss combined with sparse or low-ranker regularizers with convex optimization (Barzilai & Crammer, 2015; Murugesan et al., 2017). Deep neural networks have flexible representation power and they may overfit to arbitrary cluster assignment if we consider training loss alone. Also, these methods require identical class label sets across different tasks, which does not hold in most of the real-world MATL settings.

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) maps 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 & Larochelle, 2017) learns the exact optimization algorithm used to train another learner neural-network classifier for the few-shot setting. However, it requires uniform classes across tasks. Our FSL approach can handle the challenges of diversity and varying sets of class labels.

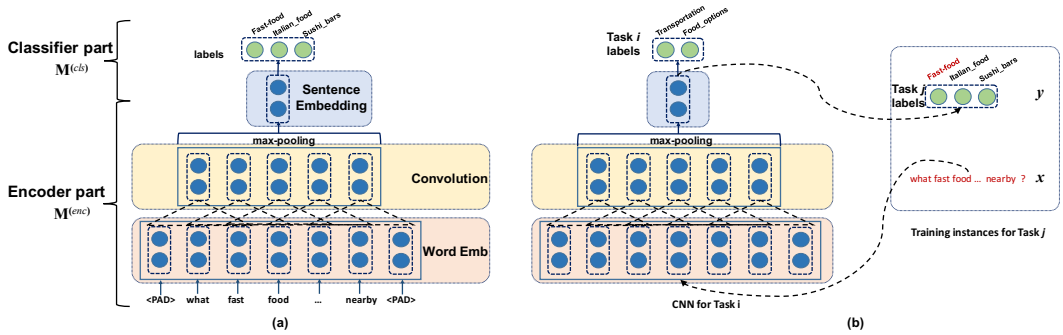

Figure 1: The convolutional neural networks used in this work: (a) a single-task CNN. The encoder component takes the sentence as input and outputs a fixed-length sentence embedding vector; the classifier component predicts class labels with the sentence embedding. (b) the evaluation on transfer performance from task i to j, where the encoder of task i was taken to encode task j's sentences and then predict task j's labels (dashed arcs).

3 METHODOLOGY

Let $\mathcal{T} = \{\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_n\}$ be the set of n tasks to be clustered, and each task T_i consists of a train/validation/test data split $\{D_i^{train}, D_i^{valid}, D_i^{test}\}$. We consider text classification tasks, comprising labeled examples $\{x, y\}$, where the input x is a sentence or document (a sequence of words) and y is the label. We first train each classification model M_i on its training set D_i^{train}, which yields a set of models $\mathcal{M} = \{\mathrm{M}_1, \mathrm{M}_2, \cdots, \mathrm{M}_n\}$. We use convolutional neural network (CNN), which has reported results near state-of-the-art on text classification (Kim, 2014; Johnson & Zhang, 2016).

CNNs also train faster than recurrent neural networks (Hochreiter & Schmidhuber, 1997), making large-n MATL scenarios more feasible. Figure 1 shows 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. The model has two parts: an *encoder* part and a *classifier* part. Hence each model $\mathrm{M}_i = \{\mathrm{M}_i^{enc}, \mathrm{M}_i^{cls}\}$. The above broad definitions encompasses other classification tasks (e.g. image classification) and other classification models (e.g. LSTMs (Hochreiter & Schmidhuber, 1997)).

We propose a task-clustering framework for both multi-task learning (MTL) and few-shot learning (FSL) settings. In this framework, we have the MTL and FSL algorithms summarized in Section 3.3 & 3.4, where our task-clustering framework serves as the initial step in both algorithms. Figure 2 gives an overview of our idea and an example on how our task-clustering algorithm helps MTL.

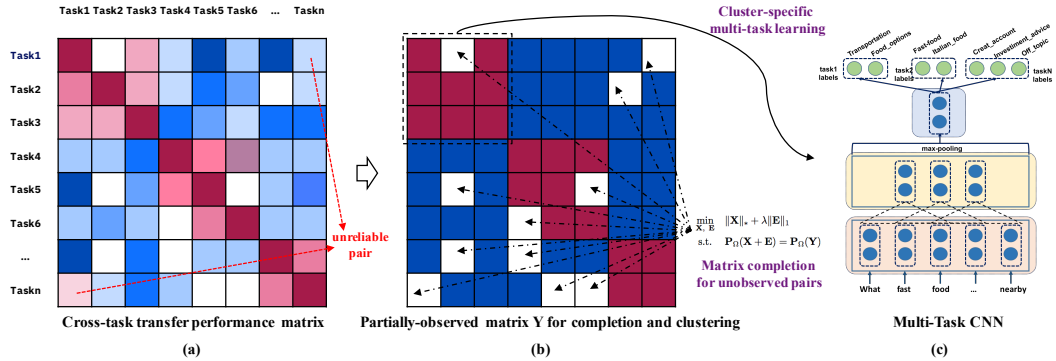

Figure 2: Overview of the idea of our Robust Clustering method with multi-task learning as an example application. (a) an illustration of the sparse cross-tasks transfer-performance matrix with unobserved entries (white blocks) and unreliable values (top-right and bottom-left corners), where red colors 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 3.2 for the detailed mathematics); (c) a multi-task Convolutional Neural Network (MTL-CNN) architecture trained on a task cluster (tasks 1, 2 and 3 in the example).

3.1 Cross-Task Transfer-Performance Matrix Estimation

Using single-task models, we can compute performance scores s_{ij} by adapting each M_i to each task $T_j (j \neq i)$. 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 $\mathbf{S}_{ij} \neq \mathbf{S}_{ji}$.

When **all tasks have identical label sets**, we can directly evaluate the model M_i on the training set of task j, D_j^{train}, and use the accuracy as the cross-task transfer score $\mathbf{S}_{ij}$.

When **tasks have different label sets**, we freeze the encoder M_i^{enc} from M_i, on top of which we use D_j^{train} to train a classifier layer. 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.

Task Pair Sampling: When the number of tasks n is very large, the evaluation of $O(n^2)$ entries is time-consuming. Thus we sample n' pairs of tasks $\{i, j\}$ $(i \neq j)$, with $n' \ll n$. Then we set $\mathbf{S}_{ij}$ and $\mathbf{S}_{ji}$ as the transfer performance defined above when $\{i, j\}$ is in the n' samples, otherwise the entry is marked as "*unobserved*" [1].

3.2 Robust Task Clustering by Matrix Completion

As discussed in the introduction, directly generating the full matrix $\mathbf{S}$ and partitioning tasks based on it has the following disadvantages: (i) there are too many entries to evaluate when the number of tasks is large; (ii) some task pairs are uncertain, thus can mislead the clustering algorithm to

[1] We set all $\mathbf{S}_{ii} = 1$.

Algorithm 1: 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 M_i for each task T_i
2 **Evaluation of Transfer-Performance Matrix**: get performance matrix **S** (Section 3.1)
3 **Score Filtering**: Filter the uncertain scores in **S** and construct the symmetric matrix **Y** using Eq. (1)
4 **Matrix Completion**: Complete the similar matrix **X** from **Y** using Eq. (2)
5 **Task Clustering**: $C_{1:K} = SpectralClustering(\mathbf{X}, K)$

output an incorrect task-partition; and (iii) **S** is asymmetric, thus cannot be directly analyzed by many conventional clustering methods. We address the first issue by randomly sample some task pairs to evaluate, as described in Section 3.1. Besides, we address the other issues by constructing a symmetric similarity matrix and only consider the reliable task relationships, as will be introduced in Eq. (1). Below, we describe our method (summarized in Algorithm 1) in detail.

First, we use only reliable task pairs to generate a *partially-observed* similarity matrix **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 **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 will have no influence to our clustering method. This leads to a partially-observed symmetric matrix **Y**, i.e.,

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

Given the partially observed matrix **Y**, we then reconstruct the full similarity matrix $\mathbf{X} \in \mathbb{R}^{n\times n}$. We first note that the similarity matrix **X** should be of low-rank (proof deferred to appendix). Additionally, since the observed entries of **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 **E** to capture the observed incorrect entries in **Y**. Combining the two observations, **Y** can be decomposed into the sum of two matrices **X** and **E**, where **X** is a low rank matrix storing similarities between task pairs, and **E** is a sparse matrix that captures the errors in **Y**. The matrix completion problem can be cast as the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{X},\,\mathbf{E}} \quad & \|\mathbf{X}\|_* + \lambda\|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{P}_\Omega(\mathbf{X} + \mathbf{E}) = \mathbf{P}_\Omega(\mathbf{Y}), \end{aligned} \tag{2}$$

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

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

The following theorem shows the perfect recovery guarantee for the problem (2). The proof is deferred to Appendix.

Theorem 3.1. *Let $\boldsymbol{X}^* \in \mathbb{R}^{n\times n}$ be a rank k matrix with a singular value decomposition $\boldsymbol{X}^* = \boldsymbol{U}\Sigma\boldsymbol{V}^\top$, where $\boldsymbol{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_k) \in \mathbb{R}^{n\times k}$ and $\boldsymbol{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_k) \in \mathbb{R}^{n\times k}$ are the left and right singular vectors of $\boldsymbol{X}^*$, respectively. Similar to many related works of matrix completion, we assume that the following two assumptions are satisfied:*

1. *The row and column spaces of $\boldsymbol{X}$ have coherence bounded above by a positive number μ_0.*
2. *Max absolute value in matrix $\boldsymbol{U}\boldsymbol{V}^\top$ is bounded above by $\mu_1\sqrt{r}/n$ for a positive number μ_1.*

Suppose that m_1 entries of $\boldsymbol{X}^*$ are observed with their locations sampled uniformly at random, and among the m_1 observed entries, m_2 randomly sampled entries are corrupted. Using the resulting partially observed matrix as the input to the problem (2), then with a probability at least $1 - n^{-3}$, the underlying matrix $\boldsymbol{X}^*$ can be perfectly recovered, given

1. $\mu(\boldsymbol{E})\xi(\boldsymbol{X}) \leq \frac{1}{4k+5}$,
2. $\frac{\xi(\boldsymbol{X})-(2k-1)\mu(\boldsymbol{E})\xi^2(\boldsymbol{X})}{1-2(k+1)\mu(\boldsymbol{E})\xi(\boldsymbol{X})} < \lambda < \frac{1-(4k+5)\mu(\boldsymbol{E})\xi(\boldsymbol{X})}{(k+2)\mu(\boldsymbol{E})}$,
3. $m_1 - m_2 \geq C[\max(\mu_0, \mu_1)]^4 n \log^2 n$,

where C is a positive constant; $\xi(\circ)$ and $\mu(\circ)$ denotes the low-rank and sparsity incoherence (Chandrasekaran et al., 2011).

Theorem 3.1 implies that even if some of the observed entries computed by (1) are incorrect, problem (2) can still perfectly recover the underlying similarity matrix $\mathbf{X}^*$ if the number of observed correct entries is at least $O(n \log^2 n)$. For MATL with large n, this implies that only a tiny fraction of all task pairs is needed to reliably infer similarities over all task pairs. Moreover, the completed similarity matrix $\mathbf{X}$ is symmetric, due to symmetry of the input matrix $\mathbf{Y}$. This enables analysis by similarity-based clustering algorithms, such as spectral clustering.

3.3 Multi-Task Learning Based on Tasks Clusters

Algorithm 2: ROBUSTTC-MTL: Multi-Task Learning based on Task Clustering

Input : A set of n tasks $\mathcal{T} = \{\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_n\}$; number of clusters K
Output : K task clusters $C_{1:K}$ and cluster-models $\mathbf{\Lambda} = \{\Lambda_1, \Lambda_2, \cdots, \Lambda_K\}$

1 **Robust Task Clustering**: $C_{1:K}$ = ROBUSTTC($\mathcal{T}$,K) (Algorithm 1)
2 **Cluster-Model Training**: Train one multi-task model Λ_i on each task cluster C_i (Section 3.3)

For each cluster C_k, we train a model Λ_k with all tasks in that cluster to encourage parameter sharing. We call Λ_k the **cluster-model**. When evaluated on the MTL setting, with sufficient data to train a task-specific classifier, we only share the encoder part and have distinct task-specific classifiers (Figure 1(b)). These task-specific classifiers provide flexibility to handle varying number of labels.

3.4 Few-Shot Learning Based on Tasks Clusters

Algorithm 3: ROBUSTTC-FSL: Task Clustering for Few-Shot Learning

Input : N training tasks $\mathcal{T} = \{\mathrm{T}_1, \mathrm{T}_2, \cdots, \mathrm{T}_n\}$; number of clusters K; target few-shot learning task T_{trg}
Output : A classification model for the target task M_{trg}, K task clusters $C_{1:K}$ and cluster-models $\mathbf{\Lambda} = \{\Lambda_1, \Lambda_2, \cdots, \Lambda_K\}$

1 **Learning Cluster-Models on Training Tasks**: $C_{1:K}$, $\mathbf{\Lambda}$ = ROBUSTTC-MTL($\mathcal{T}$,K) (Algorithm 2)
2 **Few-Shot Learning on Cluster-models**: Train a model M_{trg} on task T_{trg} with the method in Section 3.4.

We only have access to a limited number of training samples in few-shot learning setting, so it is impractical to train well-performing task-specific classifiers as in the multi-task learning setting. Instead, we make the prediction of a new task by linearly combining prediction from learned clusters.

$$p(y|x) = \sum_k \alpha_k P(y|x; \Lambda_k). \tag{3}$$

where Λ_k is the learned (and frozen) model of the k-th cluster, $\{\alpha_k\}_{k=1}^K$ are adaptable parameters.

We use some alternatives to train cluster-models Λ_k, which could better suit (and is more consistent to) the above FSL method.[2] When **all tasks have identical label sets**, we train a single classification

[2]We also tried these alternatives under the MTL settings, which perform worse than MTL-CNN.

model on all the tasks like in previous work (Barzilai & Crammer, 2015), the predictor $P(y|x;\Lambda_k)$ is directly derived from this cluster-model. When **tasks have different label sets** , we train a metric-learning model like (Vinyals et al., 2016) among all the tasks in C_k, which consist a shared encoding function Λ_k^{enc} aiming to make each example closer to examples with the same label compared to the ones with different labels.

Then we use the encoding function to derive the predictor by

$$P(y=y_l|x;\Lambda_k)=\frac{\exp\left\{\Lambda_k^{enc}(x_l)^\top\Lambda_k^{enc}(x)\right\}}{\sum_{l'}\exp\left\{\Lambda_k^{enc}(x_{l'})^\top\Lambda_k^{enc}(x)\right\}} \tag{4}$$

where x_l is the corresponding training sample for label y_l.

4 EXPERIMENTS

4.1 EXPERIMENT SETUP

Data Sets We test our methods by conducting experiments on three text classification data sets. In the data-preprocessing step we used NLTK toolkit[3] for tokenization. For MTL setting, all tasks are used for clustering and model training. For FSL setting, the task are divided into training tasks and testing tasks (target tasks), where the training tasks are used for clustering and model training, the testing tasks are few-shot learning ones used to for evaluating the method in Eq. (3).

1. **Amazon Review Sentiment Classification** First, following Barzilai & Crammer (2015), we construct a multi-task learning setting 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 3 for the details). For each 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 review-buckets then form the basis of the task-setup for MATL, giving us $23\times 3=69$ tasks in total. For each domain we distribute the reviews uniformly to the three tasks. For evaluation, we select tasks from 4 domains (*Books, DVD, Electronics, Kitchen*) as the target tasks (12 tasks) out of all 23 domains. For FSL evaluation, we create five-shot learning tasks on the selected target tasks. The cluster-models for this evaluation are standard CNNs shown in Figure 1(a), and we share the same output layer to evaluate the probability in Eq. (3) as all tasks have the same number of labels.
2. **Diverse Real-World Tasks: User Intent Classification for Dialog System** The second dataset is from an on-line 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 a customer-service request scenarios. During classification, intent-labels[5] are assigned to user utterances (usually sentences). We use a total of 175 tasks from different clients, and randomly sample 10 tasks from them as our target tasks. For each task, we randomly sample 64% data into a training set, 16% into a validation set, and use the rest as the test set (see Appendix 3 for details). The number of labels for these tasks vary from 2 to 100. Hence, to adapt this to a FSL scenario, we keep one example for each label (one-shot), plus 20 randomly picked labeled examples to create our training data. We believe this is a fairly realistic estimate of labeled examples one client could provide easily. Since we deal with various number of labels in the FSL setting, we chose matching networks (Vinyals et al., 2016) as the cluster-models.
3. **Extra-Large Number of Real-World Tasks** Similar to the second dataset, we further collect 1,491 intent classification tasks from the on-line service. This setting is mainly used to verify the robustness of our task clustering method, since it is difficult to estimate the full transfer-performance matrix **S** in this setting ($1{,}491^2$=2.2M entries). Therefore, in

[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.

order to extract task clusters, we randomly sample task pairs from the data set to obtain 100,000 entries in $\mathbf{S}$, which means that only about $100\text{K}/2.2\text{M} \approx 4.5\%$ of the entries in $\mathbf{S}$ are observed. The number of 100,000 is chosen to be close to $n \log^2 n$ in our theoretical bound in Theorem 3.1, so that we could also verify the tightness of the bound empirically. To make the best use of the sampled pairs, in this setting we modified the Eq. 1, so that each entry $\mathbf{Y}_{ij} = \mathbf{Y}_{ji} = 1$ if $\mathbf{S}_{ij} \geq \mu_j$ or $\mathbf{S}_{ji} \geq \mu_i$ and $\mathbf{Y}_{ij} = 0$ otherwise. In this way we could have determined number of entries in $\mathbf{Y}$ as well, since all the sampled pairs will correspond to observed (but noisy) entries in $\mathbf{Y}$. We only run MTL setting on this data set.

Baselines For MTL setting, we compare our method to the following baselines: (1) **single-task CNN**: training a CNN model for each task individually; (2) **holistic MTL-CNN**: training one MTL-CNN model (Figure 1b) on all tasks; (3) **holistic MTL-CNN (target only)**: training one MTL-CNN model on all the target tasks. For FSL setting, the baselines consist of: (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**: fine-tuning the classifier layer on each target task after training initial MTL-CNN model on all training tasks; (4) **Matching Network**: a metric-learning based few-shot learning model trained on all training tasks. We initialize all models with pre-trained 100-dim Glove embeddings (trained on 6B corpus) (Pennington et al., 2014).

As the intent classification tasks usually have various numbers of labels, to our best knowledge the proposed method is the only one supporting task clustering in this setting; hence we only compare with the above baselines. Since sentiment classification involves binary labels, we compare our method with the state-of-the-art logistic regression based task clustering method (**ASAP-MT-LR**) (Barzilai & Crammer, 2015). We also try another approach where we run our MTL/FSL methods on top of the (**ASAP-Clus-MTL/FSL**) clusters (as their entire formulation is only applicable to convex models).

Hyper-Parameter Tuning In all experiments, we set both p_1 and p_2 parameters in (1) 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. For MTL settings, we tune parameters like the window size and hidden layer size of CNN, learning rate and the initialization of embeddings (random or pre-trained) based on average accuracy on the union of all tasks' dev sets, in order to find the best identical setting for all tasks. Finally we have the CNN with window size of 5 and 200 hidden units. The learning rate is selected as 0.001; and all MTL models use random initialized word embeddings on sentiment classification and use Glove embeddings as initialization on intent classification, which is likely because the training sets of the intent tasks are usually small. We also used the early stopping criterion based on the previous condition.

For the FSL setting, hyper-parameter selection is difficult since there is no validation data (which is a necessary condition to qualify as a k-shot learning). So, in this case we preselect a subset of training tasks as validation tasks and tune the learning rate and training epochs (for the rest we follow the best setting from the MTL experiments) on the validation tasks. During the testing phase (i.e. model training on the target FSL tasks), we fix the selected hyper-parameter values for all the algorithms.

Out-of-Vocabulary in Transfer-Performance Evaluation In text classification tasks, transferring an encoder with fine-tuned word embeddings from one task to another may not work as there can be a significant difference between the vocabularies. Hence, while learning the single-task models (line 1 of Algorithm 1) we always use the CNNs with fixed set of pre-trained embeddings.

4.2 Sentiment Classification on Amazon Product Reviews

Improving Observed Tasks (MTL Setting) Table 1 shows the results of the 12 target tasks when all 69 tasks are used for training. Since most of the tasks have a significant amount of training data, the single-task baselines achieve good results. Because the conflicts among some tasks (e.g. the 2-star bucket tasks and 5-star bucket tasks require opposite labels on 4-star examples), the holistic MTL-CNN does not show accuracy improvements compared to the single-task methods. It also lags behind the holistic MTL-CNN model trained only on 12 target domains, which indicates that the holistic MTL-CNN cannot leverage large number of background tasks. Our RobustTC-MTL method based on task clustering achieves a significant improvement over all the baselines.

Table 1: Accuracy on the 12 target sentiment classification tasks.

(a) MTL setting (i.e. training on all 69 tasks).

Model	Avg Acc
Single-task CNN	85.51
ASAP-MTLR (Barzilai & Crammer, 2015)	85.17
Holistic MTL-CNN	85.23
Holistic MTL-CNN (target only)	85.71
ROBUSTTC-MTL	
clus=5	86.13
clus=10	**86.73**
ASAP-Clus-MTL	
clus=5	86.07
clus=10	85.60

(b) Few-shot learning setting (five-shot).

Model	Avg Acc
Single-task CNN w/ pre-trained emb	65.92
Single-task FastText w/ pre-trained emb	63.05
Fine-tuned the holistic MTL-CNN	76.56
Matching Network (Vinyals et al., 2016)	65.73
ROBUSTTC-FSL	
clus=5	**83.12**
clus=10	81.96
clus=num_of_tasks (no clustering)	78.85
ASAP-Clus-FSL	
clus=5	82.65
clus=10	81.44

The ASAP-MTLR (best score achieved with five clusters) could improve single-task linear models with similar merit of our method. However, it is restricted by the representative strength of linear models so the overall result is lower than the deep learning baselines.

Adaptation to New Tasks (FSL Setting) Table 1(b) shows the results on the 12 five-shot tasks by leveraging the learned knowledge from the 57 previously observed tasks. Due to the limited training resources, all the baselines perform poorly. Our ROBUSTTC-FSL gives far better results compared to all baselines ($>6\%$). It is also significantly better than applying Eq. (3) without clustering (78.85%), i.e. using single-task model from each task instead of cluster-models for $P(y|x;\cdot)$.

Comparison to the ASAP Clusters Our clustering-based MTL and FSL approaches also work for the ASAP clusters, in which we replace our task clusters with the task clusters generated by ASAP-MTLR. In this setting we get a slightly lower performance compared to the ROBUSTTC-based ones on both MTL and FSL settings, but overall it performs better than the baseline models. This result shows that, apart from the ability to handle varying number of class labels, our ROBUSTTC model can also generate better clusters for MTL/FSL of deep networks, even under the setting where all tasks have the same number of labels.

It is worth to note that from Table 1(a), training CNNs on the ASAP clusters gives better results compared to training logistic regression models on the same 5 clusters (86.07 vs. 85.17), despite that the clusters are not optimized for CNNs. Such result further emphasizes the importance of task clustering for deep models, when better performance could be achieved with such models.

4.3 USER INTENT CLASSIFICATION FROM DIVERSE REAL-WORLD ONLINE SERVICES

Table 2: Accuracy on the 10 dialog intent classification tasks.

(a) MTL setting (i.e. training on all 175 tasks).

Model	Avg Acc
Single-task CNN	58.47
Holistic MTL-CNN	62.42
Holistic MTL-CNN (target only)	62.45
ROBUSTTC-MTL	
clus=10	64.41
clus=20	**68.11**
clus=30	66.74

(b) FSL setting (one-shot + 20 examples).

Model	Avg Acc
Single-task CNN w/pre-trained emb	34.46
Single-task FastText w/pre-trained emb	23.87
Fine-tuned holistic MTL-CNN	30.36
Matching Network (Vinyals et al., 2016)	25.14
ROBUSTTC-FSL	
clus=10	34.64
clus=20	**37.59**
clus=30	36.82
clus=num_of_tasks (no clustering)	34.43
Adaptive ROBUSTTC-FSL (clus=20)	**42.97**

Table 2(a) & (b) show the MTL & FSL results on dialog intent classification, which demonstrates trends similar to the sentiment classification tasks. Note that the holistic MTL methods achieve much better results compared to single-task CNNs. This is because the tasks usually have smaller training and development sets, and both the model parameters learned on training set and the hyper-parameters selected on development set can easily lead to over-fitting. ROBUSTTC-MTL achieves

large improvement (5.5%) over the best MTL baseline, because the tasks here are more diverse than the sentiment classification tasks and task-clustering greatly reduces conflicts from irrelevant tasks.

Although our ROBUSTTC-FSL improves over baselines under the FSL setting, the margin is smaller. This is because of the huge diversity among tasks – by looking at the training accuracy, we found several tasks failed because none of the clusters could provide a metric that suits the training examples. To deal with this problem, we hope that the algorithm can automatically decide whether the new task belongs to any of the task-clusters. If the task doesn't belong to any of the clusters, it would not benefit from any previous knowledge, so it should fall back to single-task CNN. The new task is treated as "out-of-cluster" when none of the clusters could achieve higher than 20% accuracy (selected on dev tasks) on its training data. We call this method **Adaptive ROBUSTTC-FSL**, and it gives more than 5% performance boost over the best ROBUSTTC-FSL result.

Discussion on Clustering-Based FSL The single metric based FSL method (Matching Network) achieved success on homogeneous few-shot tasks like Omniglot and miniImageNet (Vinyals et al., 2016) but performs poorly in both of our experiments. This indicates that it is important to maintain multiple metrics for few-shot learning problems with more diverse tasks, similar to the few-shot NLP problems investigated in this paper. Our clustering-based FSL approach maintains diverse metrics while keeping the model simple with only K parameters to estimate. It is worthwhile to study how and why the NLP problems make few-shot learning more difficult/heterogeneous; and how well our method can generalize to non-NLP problems like miniImageNet. We will leave these topics for future work.

4.4 LARGE-SCALE USER INTENT CLASSIFICATION WITH TASK-PAIR SAMPLING

Table 3 shows the MTL results on the extra-large dialog intent classification dataset. Compared to the results on the 175 tasks, the holistic MTL-CNN achieves larger improvement (6%) over the single-task CNNs, which is a stronger baseline. Similar as the observation on the 175 tasks, here the main reason for its improvement is the consistent development and test performance due to holistic multi-task training approach: both the single-task and holistic multi-task model achieve around 66% average accuracy on development sets. Unlike the experiments in Section 4.3, we did not evaluate the full transfer-performance matrix **S** due to time considerations. Instead, we only use the information of $\sim 4.5\%$ of all the task-pairs, and our algorithm still achieves a significant improvement over the baselines. Note that this result is obtained by only sampling about $n \log^2 n$ task pairs, it not only confirms the empirical advantage of our multi-task learning algorithm, but also verifies the correctness of our theoretical bound in Theorem 3.1.

Table 3: Accuracy of Multi-Task Learning on the 1,491 dialog intent classification tasks.

Model	**Avg Acc**
Single-task CNN	60.49
Holistic MTL-CNN	66.42
ROBUSTTC-MTL	
clus=30	69.62
clus=40	**70.50**
clus=50	69.87

5 CONCLUSION

In this paper, we propose a robust task-clustering method that not only has strong theoretical guarantees but also demonstrates significantly empirical improvements when equipped by our MTL and FSL algorithms. Our empirical studies verify that (i) the proposed task clustering approach is very effective in the many-task learning setting especially when tasks are diverse; (ii) our approach could efficiently handle large number of tasks as suggested by our theory; and (iii) cross-task transfer performance can serve as a powerful task similarity measure. Our work opens up many future research directions, such as supporting online many-task learning with incremental computation on task similarities, and combining our clustering approach with the recent learning-to-learn methods (e.g. (Ravi & Larochelle, 2017)), to enhance our MTL and FSL methods.

REFERENCES

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

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

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

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

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

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

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

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

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

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

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

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

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

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

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

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

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

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

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

Harm van Seijen, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang. Hybrid reward architecture for reinforcement learning. *arXiv preprint arXiv:1706.04208*, 2017.

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

Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. *arXiv preprint arXiv:1605.06391*, 2016.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

APPENDIX A: PROOF OF LOW-RANKNESS OF MATRIX X

We first prove that the full similarity matrix $\mathbf{X} \in \mathbb{R}^{n\times n}$ is of low-rank. To see this, let $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_k)$ be the underlying perfect clustering result, where k is the number of clusters and $\mathbf{a}_i \in \{0,1\}^n$ is the membership vector for the i-th cluster. Given $\mathbf{A}$, the similarity matrix $\mathbf{X}$ is computed as

$$\mathbf{X} = \sum_{i=1}^{k} \mathbf{a}_i \mathbf{a}_i^\top = \sum_{i=1}^{k} \mathbf{B}_i$$

where $\mathbf{B}_i = \mathbf{a}_i\mathbf{a}_i^\top$ is a rank one matrix. Using the fact that $\text{rank}(\mathbf{X}) \leq \sum_{i=1}^{k} \text{rank}(\mathbf{B}_i)$ and $\text{rank}(\mathbf{B}_i) = 1$, we have $\text{rank}(\mathbf{X}) \leq k$, i.e., the rank of the similarity matrix $\mathbf{X}$ is upper bounded by the number of clusters. Since the number of clusters is usually small, the similarity matrix $\mathbf{X}$ should be of low rank.

APPENDIX B: PROOF OF THEOREM 4.1

We then prove our main theorem. First, we define several notations that are used throughout the proof. Let $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$ be the singular value decomposition of matrix $\mathbf{X}$, where $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_k) \in \mathbb{R}^{n\times k}$ and $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_k) \in \mathbb{R}^{n\times k}$ are the left and right singular vectors of matrix $\mathbf{X}$, respectively. Similar to many related works of matrix completion, we assume that the following two assumptions are satisfied:

1. **A1**: the row and column spaces of $\mathbf{X}$ have coherence bounded above by a positive number μ_0, i.e., $\sqrt{n/r}\max_i \|\mathbf{P}_\mathbf{U}(\mathbf{e}_i)\| \leq \mu_0$ and $\sqrt{n/r}\max_i \|\mathbf{P}_\mathbf{V}(\mathbf{e}_i)\| \leq \mu_0$, where $\mathbf{P}_\mathbf{U} = \mathbf{U}\mathbf{U}^\top$, $\mathbf{P}_\mathbf{V} = \mathbf{V}\mathbf{V}^\top$, and $\mathbf{e}_i$ is the standard basis vector, and
2. **A2**: the matrix $\mathbf{U}\mathbf{V}^\top$ has a maximum entry bounded by $\mu_1\sqrt{r}/n$ in absolute value for a positive number μ_1.

Let T be the space spanned by the elements of the form $\mathbf{u}_i\mathbf{y}^\top$ and $\mathbf{x}\mathbf{v}_i^\top$, for $1 \leq i \leq k$, where $\mathbf{x}$ and $\mathbf{y}$ are arbitrary n-dimensional vectors. Let $T^\perp$ be the orthogonal complement to the space T, and let $\mathbf{P}_T$ be the orthogonal projection onto the subspace T given by

$$\mathbf{P}_T(\mathbf{Z}) = \mathbf{P}_\mathbf{U}\mathbf{Z} + \mathbf{Z}\mathbf{P}_\mathbf{V} - \mathbf{P}_\mathbf{U}\mathbf{Z}\mathbf{P}_\mathbf{V}.$$

The following proposition shows that for any matrix $\mathbf{Z} \in T$, it is a zero matrix if enough amount of its entries are zero.

Proposition 1. *Let Ω be a set of m entries sampled uniformly at random from $[1,\ldots,n]\times[1,\ldots,n]$, and $\mathbf{P}_\Omega(\mathbf{Z})$ projects matrix $\mathbf{Z}$ onto the subset Ω. If $m > m_0$, where $m_0 = C_R^2\mu_0 rn\beta\log n$ with $\beta > 1$ and C_R being a positive constant, then for any $\mathbf{Z} \in T$ with $\mathbf{P}_\Omega(\mathbf{Z}) = 0$, we have $\mathbf{Z} = 0$ with probability $1 - 3n^{-\beta}$.*

Proof. According to the Theorem 3.2 in Candès & Tao (2010), for any $\mathbf{Z} \in T$, with a probability at least $1 - 2n^{2-2\beta}$, we have

$$\|\mathbf{P}_T(\mathbf{Z})\|_F - \delta\|\mathbf{Z}\|_F \leq \frac{n^2}{m}\|\mathbf{P}_T\mathbf{P}_\Omega\mathbf{P}_T(\mathbf{Z})\|_F^2 = 0 \tag{5}$$

where $\delta = m_0/m < 1$. Since $\mathbf{Z} \in T$, we have $P_T(\mathbf{Z}) = \mathbf{Z}$. Then from (5), we have $\|\mathbf{Z}\|_F \leq 0$ and thus $\mathbf{Z} = 0$. □

In the following, we will develop a theorem for the dual certificate that guarantees the unique optimal solution to the following optimization problem

$$\begin{aligned} \min_{\mathbf{X},\,\mathbf{E}} \quad & \|\mathbf{X}\|_* + \lambda\|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{P}_\Omega(\mathbf{X}+\mathbf{E}) = \mathbf{P}_\Omega(\mathbf{Y}). \end{aligned} \tag{6}$$

Theorem 1. *Suppose we observe m_1 entries of $\boldsymbol{X}$ with locations sampled uniformly at random, denoted by Ω. We further assume that m_2 entries randomly sampled from m_1 observed entries are corrupted, denoted by Δ. Suppose that $\boldsymbol{P}_\Omega(\boldsymbol{Y}) = \boldsymbol{P}_\Omega(\boldsymbol{X}+\boldsymbol{E})$ and the number of observed correct entries $m_1 - m_2 > m_0 = C_R^2 \mu_0 r n \beta \log n$. Then, for any $\beta > 1$, with a probability at least $1-3n^{-\beta}$, the underlying true matrices $(\boldsymbol{X}, \boldsymbol{E})$ is the unique optimizer of (6) if both assumptions* **A1** *and* **A2** *are satisfied and there exists a dual $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ such that (a) $\boldsymbol{Q} = \boldsymbol{P}_\Omega(\boldsymbol{Q})$, (b) $\boldsymbol{P}_T(\boldsymbol{Q}) = \boldsymbol{U}\boldsymbol{V}^\top$, (c) $\|\boldsymbol{P}_{T^\top}(\boldsymbol{Q})\| < 1$, (d) $\boldsymbol{P}_\Delta(\boldsymbol{Q}) = \lambda\ sgn(\boldsymbol{E})$, and (e) $\|\boldsymbol{P}_{\Delta^c}(\boldsymbol{Q})\|_\infty < \lambda$.*

Proof. First, the existence of $\mathbf{Q}$ satisfying the conditions (a) to (e) ensures that $(\mathbf{X}, \mathbf{E})$ is an optimal solution. We only need to show its uniqueness and we prove it by contradiction. Assume there exists another optimal solution $(\mathbf{X}+\mathbf{N_X}, \mathbf{E}+\mathbf{N_E})$, where $\mathbf{P}_\Omega(\mathbf{N_X}+\mathbf{N_E}) = 0$. Then we have

$$\|\mathbf{X}+\mathbf{N_X}\|_* + \lambda\|\mathbf{E}+\mathbf{N_E}\|_1 \quad \geq \quad \|\mathbf{X}\|_* + \lambda\|\mathbf{E}\|_1 + \langle \mathbf{Q_E}, \mathbf{N_E}\rangle + \langle \mathbf{Q_X}, \mathbf{N_X}\rangle$$

where $\mathbf{Q_E}$ and $\mathbf{Q_X}$ satisfying $\mathbf{P}_\Delta(\mathbf{Q_E}) = \lambda\ \mathrm{sgn}(\mathbf{E})$, $\|\mathbf{P}_{\Delta^c}(\mathbf{Q_E})\|_\infty \leq \lambda$, $\mathbf{P}_T(\mathbf{Q_X}) = \mathbf{U}\mathbf{V}^\top$ and $\|\mathbf{P}_{T^\perp}(\mathbf{Q_X})\| \leq 1$. As a result, we have

$$\begin{aligned}
& \lambda\|\mathbf{E}+\mathbf{N_E}\|_1 + \|\mathbf{X}+\mathbf{N_X}\|_* \\
\geq\ & \lambda\|\mathbf{E}\|_1 + \|\mathbf{X}\|_* + \langle \mathbf{Q} + \mathbf{P}_{\Delta^c}(\mathbf{Q_E}) - \mathbf{P}_{\Delta^c}(\mathbf{Q}), \mathbf{N_E}\rangle + \langle \mathbf{Q} + \mathbf{P}_{T^\perp}(\mathbf{Q_X}) - \mathbf{P}_{T^\perp}(\mathbf{Q}), \mathbf{N_X}\rangle \\
=\ & \lambda\|\mathbf{E}\|_1 + \|\mathbf{X}\|_* + \langle \mathbf{Q}, \mathbf{N_E}+\mathbf{N_X}\rangle + \langle \mathbf{P}_{\Delta^c}(\mathbf{Q_E}) - \mathbf{P}_{\Delta^c}(\mathbf{Q}), \mathbf{N_E}\rangle + \langle \mathbf{P}_{T^\perp}(\mathbf{Q_X}) - \mathbf{P}_{T^\perp}(\mathbf{Q}), \mathbf{N_X}\rangle \\
=\ & \lambda\|\mathbf{E}\|_1 + \|\mathbf{X}\|_* + \langle \mathbf{P}_{\Delta^c}(\mathbf{Q_E}) - \mathbf{P}_{\Delta^c}(\mathbf{Q}), \mathbf{P}_{\Delta^c}(\mathbf{N_E})\rangle + \langle \mathbf{P}_{T^\perp}(\mathbf{Q_X}) - \mathbf{P}_{T^\perp}(\mathbf{Q}), \mathbf{P}_{T^\perp}(\mathbf{N_X})\rangle
\end{aligned}$$

We then choose $\mathbf{P}_{\Delta^c}(\mathbf{Q_E})$ and $\mathbf{P}_{T^\perp}(\mathbf{Q_X})$ to be such that $\langle \mathbf{P}_{\Delta^c}(\mathbf{Q_E}), \mathbf{P}_{\Delta^c}(\mathbf{N_E})\rangle = \lambda\|\mathbf{P}_{\Delta^c}(\mathbf{N_E})\|_1$ and $\langle \mathbf{P}_{T^\perp}(\mathbf{Q_X}), \mathbf{P}_{T^\perp}(\mathbf{N_X})\rangle = \|\mathbf{P}_{T^\perp}(\mathbf{N_X})\|_*$. We thus have

$$\begin{aligned}
& \lambda\|\mathbf{E}+\mathbf{N_E}\|_1 + \|\mathbf{X}+\mathbf{N_X}\|_* \\
\geq\ & \lambda\|\mathbf{E}\|_1 + \|\mathbf{X}\|_* + (\lambda - \|\mathbf{P}_{\Delta^c}(\mathbf{Q})\|_\infty)\|\mathbf{P}_{\Delta^c}(\mathbf{N_E})\|_1 + (1 - \|\mathbf{P}_{T^\perp}(\mathbf{Q})\|)\|\mathbf{P}_{T^\perp}(\mathbf{N_X})\|_*
\end{aligned}$$

Since $(\mathbf{X}+\mathbf{N_X}, \mathbf{E}+\mathbf{N_E})$ is also an optimal solution, we have $\|\mathbf{P}_{\Omega^c}(\mathbf{N}_E)\|_1 = \|\mathbf{P}_{T^\perp}(\mathbf{N_X})\|_*$, leading to $\mathbf{P}_{\Omega^c}(\mathbf{N_E}) = \mathbf{P}_{T^\perp}(\mathbf{N_X}) = 0$, or $\mathbf{N_X} \in T$. Since $\mathbf{P}_\Omega(\mathbf{N_X}+\mathbf{N_E}) = 0$, we have $\mathbf{N_X} = \mathbf{N_E} + \mathbf{Z}$, where $P_\Omega(\mathbf{Z}) = 0$ and $\mathbf{P}_{\Omega^c}(\mathbf{N_E}) = 0$. Hence, $\mathbf{P}_{\Omega^c\cap\Omega}(\mathbf{N_X}) = 0$, where $|\Omega^c \cap \Omega| = m_1 - m_2$. Since $m_1 - m_2 > m_0$, according to Proposition 1, we have, with a probability $1-3n^{-\beta}$, $\mathbf{N_X} = 0$. Besides, since $\mathbf{P}_\Omega(\mathbf{N_X}+\mathbf{N_E}) = \mathbf{P}_\Omega(\mathbf{N_E}) = 0$ and $\Delta \subset \Omega$, we have $\mathbf{P}_\Delta(\mathbf{N_E}) = 0$. Since $\mathbf{N_E} = \mathbf{P}_\Delta(\mathbf{N_E}) + \mathbf{P}_{\Delta^c}(\mathbf{N_E})$, we have $\mathbf{N_E} = 0$, which leads to the contradiction. □

Given Theorem 1, we are now ready to prove Theorem 3.1.

Proof. The key to the proof is to construct the matrix $\mathbf{Q}$ that satisfies the conditions (a)-(e) specified in Theorem 1. First, according to Theorem 1, when $m_1 - m_2 > m_0 = C_R^2 \mu_0 r n \beta \log n$, with a probability at least $1-3n^{-\beta}$, mapping $\mathbf{P}_T\mathbf{P}_\Omega\mathbf{P}_T(\mathbf{Z}) : T \mapsto T$ is an one to one mapping and therefore its inverse mapping, denoted by $(\mathbf{P}_T\mathbf{P}_\Omega\mathbf{P}_T)^{-1}$ is well defined. Similar to the proof of Theorem 2 in Chandrasekaran et al. (2011), we construct the dual certificate $\mathbf{Q}$ as follows

$$\mathbf{Q} = \lambda\ \mathrm{sgn}(\mathbf{E}) + \epsilon_\Delta + \mathbf{P}_\Delta\mathbf{P}_T(\mathbf{P}_T\mathbf{P}_\Omega\mathbf{P}_T)^{-1}(\mathbf{U}\mathbf{V}^\top + \epsilon_T)$$

where $\epsilon_T \in T$ and $\epsilon_\Delta = \mathbf{P}_\Delta(\epsilon_\Delta)$. We further define

$$\begin{aligned}
\mathbf{H} &= \mathbf{P}_\Omega\mathbf{P}_T(\mathbf{P}_T\mathbf{P}_\Omega\mathbf{P}_T)^{-1}(\mathbf{U}\mathbf{V}^\top) \\
\mathbf{F} &= \mathbf{P}_\Omega\mathbf{P}_T(\mathbf{P}_T\mathbf{P}_\Omega\mathbf{P}_T)^{-1}(\epsilon_T)
\end{aligned}$$

Evidently, we have $\mathbf{P}_\Omega(\mathbf{Q}) = \mathbf{Q}$ since $\Delta \subset \Omega$, and therefore the condition (a) is satisfied. To satisfy the conditions (b)-(e), we need

$$\mathbf{P}_T(\mathbf{Q}) = \mathbf{U}\mathbf{V}^\top \quad \rightarrow \quad \epsilon_T = -\mathbf{P}_T(\lambda\ \mathrm{sgn}(\mathbf{E}) + \epsilon_\Delta) \tag{7}$$

$$\|\mathbf{P}_{T^\perp}(\mathbf{Q})\| < 1 \quad \rightarrow \quad \mu(\mathbf{E})\,(\lambda + \|\epsilon_\Delta\|_\infty) + \|\mathbf{P}_{T^\perp}(\mathbf{H})\| + \|\mathbf{P}_{T^\perp}(\mathbf{F})\| < 1 \tag{8}$$

$$\mathbf{P}_\Delta(\mathbf{Q}) = \lambda\ \mathrm{sgn}(\mathbf{E}) \quad \rightarrow \quad \epsilon_\Delta = -\mathbf{P}_\Delta(\mathbf{H}+\mathbf{F}) \tag{9}$$

$$|\mathbf{P}_{\Delta^c}(\mathbf{Q})|_\infty < \lambda \quad \rightarrow \quad \xi(\mathbf{X})(1 + \|\epsilon_T\|) < \lambda \tag{10}$$

Below, we will first show that there exist solutions $\epsilon_T \in T$ and ϵ_Δ that satisfy conditions (7) and (9). We will then bound $\|\epsilon_\Omega\|_\infty$, $\|\epsilon_T\|$, $\|\mathbf{P}_{T^\perp}(\mathbf{H})\|$, and $\|\mathbf{P}_{T^\perp}(\mathbf{F})\|$ to show that with sufficiently small $\mu(\mathbf{E})$ and $\xi(\mathbf{X})$, and appropriately chosen λ, conditions (8) and (10) can be satisfied as well.

First, we show the existence of ϵ_Δ and ϵ_T that obey the relationships in (7) and (9). It is equivalent to show that there exists ϵ_T that satisfies the following relation

$$\epsilon_T = -\mathbf{P}_T(\lambda \operatorname{sgn}(\mathbf{E})) + \mathbf{P}_T\mathbf{P}_\Delta(\mathbf{H}) + \mathbf{P}_T\mathbf{P}_\Delta\mathbf{P}_T(\mathbf{P}_T\mathbf{P}_\Omega\mathbf{P}_T)^{-1}(\epsilon_T)$$

or

$$\mathbf{P}_T\mathbf{P}_{\Omega\setminus\Delta}\mathbf{P}_T(\mathbf{P}_T\mathbf{P}_\Omega\mathbf{P}_T)^{-1}(\epsilon_T) = -\mathbf{P}_T(\lambda \operatorname{sgn}(\mathbf{E})) + \mathbf{P}_T\mathbf{P}_\Delta(\mathbf{H}),$$

where $\Omega \setminus \Delta$ indicates the complement set of set Δ in Ω and $|\Omega \setminus \Delta|$ denotes its cardinality. Similar to the previous argument, when $|\Omega \setminus \Delta| = m_1 - m_2 > m_0$, with a probability $1 - 3n^{-\beta}$, $\mathbf{P}_T\mathbf{P}_{\Omega\setminus\Delta}\mathbf{P}_T(\mathbf{Z}) : T \mapsto T$ is an one to one mapping, and therefore $(\mathbf{P}_T\mathbf{P}_{\Omega\setminus\Delta}\mathbf{P}_T(\mathbf{Z}))^{-1}$ is well defined. Using this result, we have the following solution to the above equation

$$\epsilon_T \quad = \quad \mathbf{P}_T\mathbf{P}_\Omega\mathbf{P}_T(\mathbf{P}_T\mathbf{P}_{\Omega\setminus\Delta}\mathbf{P}_T)^{-1}\left(-\mathbf{P}_T(\lambda \operatorname{sgn}(\mathbf{E})) + \mathbf{P}_T\mathbf{P}_\Delta(\mathbf{H})\right)$$

We now bound $\|\epsilon_T\|$ and $\|\epsilon_\Delta\|_\infty$. Since $\|\epsilon_T\| \leq \|\epsilon_T\|_F$, we bound $\|\epsilon_T\|_F$ instead. First, according to Corollary 3.5 in Candès & Tao (2010), when $\beta = 4$, with a probability $1 - n^{-3}$, for any $\mathbf{Z} \in T$, we have

$$\left\|\mathbf{P}_{T^\perp}\mathbf{P}_\Omega\mathbf{P}_T(\mathbf{P}_T\mathbf{P}_\Omega\mathbf{P}_T)^{-1}(\mathbf{Z})\right\|_F \leq \|\mathbf{Z}\|_F.$$

Using this result, we have

$$\begin{aligned}
\|\epsilon_\Delta\|_\infty &\leq \xi(\mathbf{X})\left(\|\mathbf{H}\| + \|\mathbf{F}\|\right) \\
&\leq \xi(\mathbf{X})\left(1 + \|\mathbf{P}_{T^\perp}(\mathbf{H})\|_F + \|\epsilon_T\| + \|\mathbf{P}_{T^\perp}(\mathbf{F})\|_F\right) \\
&\leq \xi(\mathbf{X})\left(2 + \|\epsilon_T\| + \|\epsilon_T\|_F\right) \\
&\leq \xi(\mathbf{X})\left[2 + (2k+1)\|\epsilon_T\|\right]
\end{aligned}$$

In the last step, we use the fact that $\operatorname{rank}(\epsilon_T) \leq 2k$ if $\epsilon_T \in T$. We then proceed to bound $\|\epsilon_T\|$ as follows

$$\|\epsilon_T\| \quad \leq \quad \mu(\mathbf{E})\left(\lambda + \|\epsilon_\Delta\|_\infty\right)$$

Combining the above two inequalities together, we have

$$\begin{aligned}
\|\epsilon_T\| &\leq \xi(\mathbf{X})\mu(\mathbf{E})(2k+1)\|\epsilon_T\| + 2\xi(\mathbf{X})\mu(\mathbf{E}) + \lambda\mu(\mathbf{E}) \\
\|\epsilon_\Delta\|_\infty &\leq \xi(\mathbf{X})\left[2 + (2k+1)\mu(\mathbf{E})(\lambda + \|\epsilon_\Delta\|_\infty\right),
\end{aligned}$$

which lead to

$$\begin{aligned}
\|\epsilon_T\| &\leq \frac{\lambda\mu(\mathbf{E}) + 2\xi(\mathbf{X})\mu(\mathbf{E})}{1 - (2k+1)\xi(\mathbf{X})\mu(\mathbf{E})} \\
\|\epsilon_\Delta\|_\infty &\leq \frac{2\xi(\mathbf{X}) + (2k+1)\lambda\xi(\mathbf{X})\mu(\mathbf{E})}{1 - (2k+1)\xi(\mathbf{X})\mu(\mathbf{E})}
\end{aligned}$$

Using the bound for $\|\epsilon_\Delta\|_\infty$ and $\|\epsilon_T\|$, we now check the condition (8)

$$1 \quad > \quad \mu(\mathbf{E})\left(\lambda + |\epsilon_\Delta|_\infty\right) + \frac{1}{2} + \frac{k}{2}\|\epsilon_T\|$$

or

$$\lambda < \frac{1 - \xi(\mathbf{X})\mu(\mathbf{E})(4k+5)}{\mu(\mathbf{E})(k+2)}$$

For the condition (10), we have

$$\lambda > \xi(\mathbf{X}) + \xi(\mathbf{X})\|\epsilon_T\|$$

or

$$\lambda > \frac{\xi(\mathbf{X}) - (2k-1)\xi^2(\mathbf{X})\mu(\mathbf{E})}{1 - 2(k+1)\xi(\mathbf{X})\mu(\mathbf{E})}$$

To ensure that there exists $\lambda \geq 0$ satisfies the above two conditions, we have

$$1 - 5(k+1)\xi(\mathbf{X})\mu(\mathbf{E}) + (10k^2 + 21k + 8)[\xi(X)\mu(\mathbf{E})]^2 > 0$$

and

$$1 - \xi(\mathbf{X})\mu(\mathbf{E})(4k+5) \geq 0$$

Since the first condition is guaranteed to be satisfied for $k \geq 1$, we have

$$\xi(\mathbf{X})\mu(\mathbf{E}) \leq \frac{1}{4k+5}.$$

Thus we finish the proof. □

APPENDIX C: DATA STATISTICS

We listed the detailed domains of the sentiment analysis tasks in Table 4. We removed the *musical_instruments* and *tools_hardware* domains from the original data because they have too few labeled examples. The statistics for the 10 target tasks of intent classification in Table 5

Table 4: Statistics of the Multi-Domain Sentiment Classification Data.

Domains	**#train**	**#validation**	**#test**
apparel	7398	926	928
automotive	601	69	66
baby	3405	437	414
beauty	2305	280	299
books	19913	2436	2489
camera_photo	5915	744	749
cell_phones_service	816	109	98
computer_video_games	2201	274	296
dvd	19961	2624	2412
electronics	18431	2304	2274
gourmet_food	1227	182	166
grocery	2101	268	263
health_personal_care	5826	687	712
jewelry_watches	1597	188	196
kitchen_housewares	15888	1978	1990
magazines	3341	427	421
music	20103	2463	2510
office_products	337	54	40
outdoor_living	1321	143	135
software	1934	254	202
sports_outdoors	4582	566	580
toys_games	10634	1267	1246
video	19941	2519	2539

Table 5: Statistics of the User Intent Classification Data.

Dataset ID	**#labeled instances**	**#labels**
1	497	11
2	3071	14
3	305	21
4	122	7
5	110	11
6	126	12
7	218	45
8	297	10
9	424	4
10	110	17