Semi-Supervised Sentiment Classification and Emotion Distribution Learning Across Domains

CHEN, Yufu; RAO, Yanghui; CHEN, Shurui; LEI, Zhiqi; XIE, Haoran; LAU, Raymond Y. K.; YIN, Jian

Published in:
ACM Transactions on Knowledge Discovery from Data

Published: 27/02/2023

Document Version:
Post-print, also known as Accepted Author Manuscript, Peer-reviewed or Author Final version

Published version (DOI):
10.1145/3571736

Publication details:
CHEN, Y., RAO, Y., CHEN, S., LEI, Z., XIE, H., LAU, R. Y. K., & YIN, J. (2023). Semi-Supervised Sentiment Classification and Emotion Distribution Learning Across Domains. *ACM Transactions on Knowledge Discovery from Data*, *17*(5), Article 74. Advance online publication. https://doi.org/10.1145/3571736

Semi-supervised Sentiment Classification and Emotion Distribution Learning Across Domains

YUFU CHEN, School of Computer Science and Engineering, Sun Yat-sen University, China
YANGHUI RAO, School of Computer Science and Engineering, Sun Yat-sen University, China
SHURUI CHEN, School of Computer Science and Engineering, Sun Yat-sen University, China
ZHIQI LEI, School of Computer Science and Engineering, Sun Yat-sen University, China
HAORAN XIE, Department of Computing and Decision Sciences, Lingnan University, China
RAYMOND Y. K. LAU, Department of Information Systems, College of Business, City University of Hong Kong, China
JIAN YIN, School of Artificial Intelligence, Guangdong Key Laboratory of Big Data Analysis and Processing, Sun Yat-sen University, China

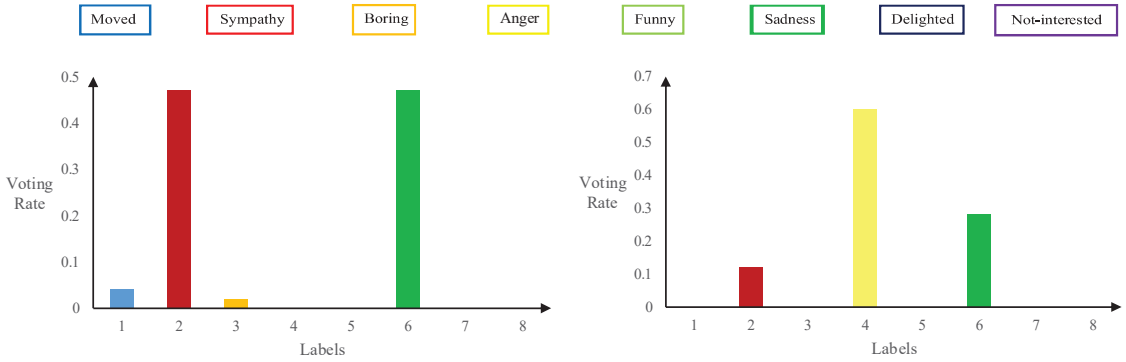
In this study, sentiment classification and emotion distribution learning across domains are both formulated as a semi-supervised domain adaptation problem, which utilizes a small amount of labeled documents in the target domain for model training. By introducing a shared matrix that captures the stable association between document clusters and word clusters, non-negative matrix tri-factorization (NMTF) is robust to the labeled target domain data and has shown remarkable performance in cross-domain text classification. However, the existing NMTF-based models ignore the incompatible relationship of sentiment polarities and the relatedness among emotions. Besides, their applications on large-scale datasets are limited by the high computation complexity. To address these issues, we propose a semi-supervised NMTF framework for sentiment classification and emotion distribution learning across domains. Based on a many-to-many mapping between document clusters and sentiment polarities (or emotions), we first incorporate the prior information of label dependency to improve the model performance. Then, we develop a parallel algorithm based on message passing interface (MPI) to further enhance the model scalability. Extensive experiments on real-world datasets validate the effectiveness of our method.

CCS Concepts: • **Information systems** → **Sentiment analysis**; • **Computing methodologies** → **Semi-supervised learning settings**.

Additional Key Words and Phrases: Semi-supervised learning, Sentiment classification, Emotion distribution learning, Non-negative matrix tri-factorization, Label dependency

The corresponding author: Yanghui Rao.
Authors' addresses: Yufu Chen, chenyf66@mail2.sysu.edu.cn, School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China; Yanghui Rao, School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, raoyangh@mail.sysu.edu.cn; Shurui Chen, School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, chenshr8@mail2.sysu.edu.cn; Zhiqi Lei, School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, leizhq5@mail2.sysu.edu.cn; Haoran Xie, Department of Computing and Decision Sciences, Lingnan University, New Territories, Hong Kong SAR, China, hrxie2@gmail.com; Raymond Y. K. Lau, Department of Information Systems, College of Business, City University of Hong Kong, Kowloon, Hong Kong SAR, China, raylau@cityu.edu.hk; Jian Yin, School of Artificial Intelligence, Guangdong Key Laboratory of Big Data Analysis and Processing, Sun Yat-sen University, Guangzhou, China, issjyin@mail.sysu.edu.cn.

1556-4681/2022/11-ART111 $15.00
https://doi.org/10.1145/1122445.1122456



(a) The twins both suffer from serious diseases, rendering the poor family needs to decide who to treat first.

(b) A two-and-half-year old girl was suffocated to death in the kindergarten bus.

Fig. 1. The examples of emotion distribution labels. The color of each bar corresponds to the border-color of the above label tags, and the height of each bar indicates the voting rate for the corresponding emotion.

ACM Reference Format:
Yufu Chen, Yanghui Rao, Shurui Chen, Zhiqi Lei, Haoran Xie, Raymond Y. K. Lau, and Jian Yin. 2022. Semi-supervised Sentiment Classification and Emotion Distribution Learning Across Domains. *ACM Trans. Knowl. Discov. Data.* 37, 4, Article 111 (November 2022), 30 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

With the development of Web 2.0, more and more users share their feelings, opinions, and emotions on the Internet. For instance, many online shopping websites tend to collect customers' comments, reviews, and ratings of a product [2], and several news websites provide a kind of service that lets readers convey their emotions after browsing a news article [1]. The former data (e.g., product reviews and social texts) is often explored by sentiment classification or opinion mining tasks [11, 12], while the prediction of reader emotions is formulated as emotion distribution learning that aims to identify multiple emotion categories with their intensities for an instance [50]. Although sentiment classification and emotion distribution learning are similar in capturing and predicting users' attitudes, there are still some major differences between the two tasks. On the one hand, sentiment classification aims at labeling a new instance according to polarity categories (e.g., positive and negative) or rating scores (e.g., 1 to 5 stars), in which, sentiment polarities are mostly incompatible. On the other hand, emotion distribution learning focuses on annotating unlabeled instances with multiple emotion categories (e.g., joy, sadness, fear, and surprise) and their strengths. In addition, one instance may trigger several emotions simultaneously, such as "happily surprised" or "angrily surprised", which reveals the importance of label dependency in learning emotion distributions [33]. An example of emotion labels and user voting numbers is presented in Figure 1. It shows that people's feelings are often compound. For news related to poverty and diseases, most readers feel "sympathy" and "sadness", while people would feel "angry" and "sadness" for the news related to accidental death. In this task, the compound representation by emotion distribution is more suitable to express human emotions than single label classification.

The studies of sentiment classification mostly rely on training a classifier which is enhanced by exploiting emoticons [11] or sentiment lexicons [24, 46]. Research into emotion distribution learning began with the affective text task in SemEval-2007 [40], which focused on exploiting reader emotions with individual words by supervised learning algorithms. Another solution to emotion distribution learning attempts to associate emotions with topics [1, 33, 36]. However, these models trained on a source domain may not perform well on a target domain because of different

domain-specific words or topics. For example, in the domain of finance, the words "bull", "bear", and "market", and the topic "stock market", are important to trigger reader emotions. While in the sports domain, the words "goal", "score", and "rebound", and the topic "score a goal", are critical to capture emotion distributions. Due to this, the existing models need to be re-trained on data of each new domain from scratch, which is quite inefficient. Cross-domain sentiment classification and emotion distribution learning methods are thus developed to improve the generalization performance on the target domain by exploiting knowledge from a source domain.

For cross-domain sentiment classification, Blitzer et al. [3] first proposed a domain adaptive sentiment polarity prediction method through modeling the common information shared by different domains and transferring the learned knowledge between domains. Furthermore, Blitzer et al. [2] exploited domain common words and domain unique words to predict the sentiment polarity of each document from the target domain. However, these methods can only predict the sentiment polarity of documents. In other words, they are inapplicable for fine-grained emotion distribution learning. The cross-domain emotion distribution learning task was first formulated by Zhang et al. [51] as a semi-supervised domain adaptation problem, which utilizes a small amount of labeled data from a target domain and abundant labeled data from a source domain for model training. In their word-level method, the differences of word distributions between source and target domains were modeled by the Gaussian kernel, and the importance of each labeled document for prediction was further exploited. However, the generalization ability of such a word-level model may be limited, because different emotions can be evoked by the same word in different contexts. For instance, the emotions of "sadness" and "fear" may be associated with the same word "bear" in the contexts of "a widespread bear market" and "a bear slips out the zoo", respectively. To address this, a topic-level model was proposed to capture different senses of the same word by topic modeling, in which topics that are context-independent and those that are context-dependent are further distinguished explicitly [35]. Unfortunately, it generates topics without using any labeled data for guidance, which may result in the mismatching between emotions and topics.

As a flexible and mathematical elegance framework, non-negative matrix factorization (NMF) has made remarkable progress in a variety of applications. For emotion detection over text, Wang and Pal [44] proposed a constraint optimization model based on NMF and lexicons. Nevertheless, this model faces two major deficiencies when applied to cross-domain emotion distribution learning. First, this model is only designed for single-domain emotion detection, suffering from the difference of word-level distributions across domains. Second, this model employs the information from manually-developed emotion lexicons as a constraint, which heavily relies on the language, scale, and quality of the emotion lexicon. By introducing a shared matrix that captures the stable association between document clusters and word clusters, Zhuang et al. [56] first developed a non-negative matrix tri-factorization (NMTF) based method for cross-domain document classification. However, this method is limited by the assumption that there exists a one-to-one mapping between document clusters and labels, i.e., the number of document clusters must equal to that of labels. To address this issue, Qin et al. [30] constructed a many-to-many mapping between document clusters and labels, and incorporated a content-based constraint into NMTF for cross-domain emotion distribution learning. Nevertheless, it neglects the label dependency among emotions and exhibits a major limitation of high computational complexity.

In light of these considerations, this paper is concerned with the following research questions: i) Given incompatible sentiment polarities and correlated emotions, how to incorporate the prior information of label dependency into an NMTF based model effectively? ii) Considering the high computation complexity, is it possible to apply the NMTF based method to large-scale datasets? To address these research questions, we first incorporate the prior dependency between sentiment polarities (or emotions) into our framework named semi-supervised NMTF (sNMTF). To solve

the objective function of sNMTF, we propose an iterative algorithm and theoretically show its convergence and computational complexity. Then, we develop a parallel algorithm based on message passing interface (MPI) for sNMTF to further enhance the model scalability. Through empirical studies based on various domain adaptation tasks, our experimental results confirm that the proposed method outperforms state-of-the-art sentiment classification and emotion distribution learning models. The remainder of this paper is organized as follows. We describe related work in Section 2. We present the sNMTF framework for cross-domain sentiment classification and emotion distribution learning in Section 3. We detail the datasets, results, and discussions in Section 4. Finally, we draw the conclusions in Section 5.

2 RELATED WORK

We here summarize related studies on sentiment classification and emotion distribution learning from the perspectives of single-domain and cross-domain, respectively.

2.1 Sentiment Classification

Sentiment classification, which deals with the computational tasks of annotating opinions and sentiments in text [26], is similar to emotion classification in natural language processing areas. Preliminary works on sentiment classification have mainly focused on classifying the sentiment of documents by machine learning algorithms. Pang et al. [27] employed three standard supervised learning methods, naïve Bayes, maximum entropy, and support vector machines, to classify movie reviews as positive and negative sentiments. Their study indicated that the three algorithms did not perform as well on sentiment classification as on traditional text classification. Go et al. [11] presented the accuracy of sentiment classification over Twitter messages using distant supervision. In their study, naïve Bayes, maximum entropy, and support vector machines were trained with emoticon data. Sentiment lexicons have also been explored to classify the sentiment of documents. For instance, the SentiWordNet lexical resource was exploited by Ohana et al. [24] to conduct sentiment classification on movie reviews. SentiWordNet is an opinion lexicon derived from the WordNet database, where each term is associated with numerical scores indicating positive and negative sentiment information. Some other sentiment or opinion lexicons were employed in sentiment classification, including the MPQA Subjectivity Lexicon [46], SemEval-2015 English Twitter Sentiment Lexicon [37], Hu & Liu's Lexicon [14], Sentiment Composition Lexicon of Negators, Modals, and Adverbs (SCL-NMA) (aka the SemEval-2016 General English Sentiment Modifiers Lexicon) [17], and NRC Hashtag Sentiment Lexicon [18].

Transfer learning focuses on improving the generalization ability of a model that is trained and evaluated on different domains [23]. In a preliminary study, Blizer et al. [3] introduced a structural correspondence learning (SCL) model to capture correspondences among features from different domains, which shed light on the research of cross-domain sentiment classification. To alleviate the word distribution gap between source and target domains, feature ensemble and instance adaptation were widely used in cross-domain sentiment classification. Blitzer et al. [2] investigated domain adaptation for sentiment classifiers by extending the aforementioned SCL algorithm. Pan et al. [25] firstly distinguished words in different domains as domain-specific and domain-independent words, and then grouped domain-specific words by using the domain-independent word as a bridge. Glorot et al. [10] proposed a deep learning approach that learned to extract a high-level representation for each review in an unsupervised fashion. Xia et al. [47] presented a feature ensemble plus sample selection approach for sentiment classification by taking both labeling adaptation and instance adaptation into account. Sharma et al. [39] showed that a weighted ensemble of the classifiers enhanced the performance of cross-domain sentiment classification. Li et al. [19] proposed a hierarchical attention network that transferred attentions for sentiments across domains using

domain-shared sentiment words as the bridge. Peng et al. [28] developed a boosting method to co-train two sentiment classifiers using the domain-specific information. He et al. [12] proposed a deep learning approach to minimize the distance between the source and target domain data in the embedded feature space. Zhang et al. [49] presented an interactive attention transfer mechanism, which could better transfer sentiments across domains by incorporating information of both sentences and aspects. Qu et al. [32] introduced a mechanism of adversarial learning which enabled the generator networks to generate more discriminative features according to the differences between the source domain and the target domain. Xue et al. [48] developed an adversarial mutual learning method which involved two groups of feature extractors, domain discriminators, sentiment classifiers, and label probers, to obtain domain-invariant features and explore sentiment polarities in each group for prediction. Although experimental results of the aforementioned models show convincing performances on sentiment classification, these methods were inapplicable to emotion distribution learning since they can only assign a polarity or a score to each document.

2.2 Emotion Distribution Learning

Emotion distribution learning aims at predicting both the category and the strength of emotional responses shared by various readers, and a benchmarking platform for related researches was introduced in SemEval-2007 [40]. Among the existing methods to learn emotion distributions from text, machine learning algorithms are used more than others (e.g., lexicon-based methods) because the underlying assumption is that all words, even neutral ones from news documents, can effectively induce a corresponding pleasant or painful reaction in readers [1].

The SWAT system was first designed to learn emotion distributions of news headlines [16]. In the SWAT system, a word-emotion mapping lexicon was first constructed, in which, each word was scored according to multiple emotion labels by the Bayes theorem. Then, the lexicon was used to predict emotion distributions of unlabeled news headlines. Considering the connection between emotions and a specific topic, Bao et al. [1] proposed the emotion-topic model (ETM) to associate text emotions with latent topics. By extending the labeled latent Dirichlet allocation [34], ETM incorporated supervision by constraining the model to use those topics that only correspond to the observed label set of a document. Rao et al. [36] exploited an additional topic layer between emotions and texts by one-to-one thematic mapping of labels and words, and developed two supervised topic models for emotion distribution learning. Quan et al. [33] proposed a logistic regression model with emotion dependency for emotion detection, where latent variables were introduced to model the latent document structure. With the development of deep learning, several works employed neural networks to learn emotion distributions. Zhang et al. [50] proposed a multi-task end-to-end framework to learn emotion distributions based on convolutional neural network. Wang et al. [43] proposed a dependency embedded recursive neural network by taking word dependency relations and the document topical information into consideration. However, the above methods are all designed for a single domain, which may not perform well on a new domain.

For cross-domain emotion distribution learning, Zhang et al. [51] proposed a method based on the logistic regression model and introduced a regularization term to prevent overfitting on the source domain. Unfortunately, the above model might not guarantee a good performance on the target domain. By distinguishing between topics that were context-independent and those that were context-dependent, a topic-level model was proposed for adaptive emotion distribution learning [35]. However, the topics extracted by the above model might not match the emotions for lack of the label guidance in generating topics.

Another stream of work focused on fine-tuning off-the-shelf systems (e.g., BERT) to solve downstream tasks [4, 13]. Although these methods had achieved notable performance on many

applications, they still suffered from some limitations in our research problem. Firstly, it is challenging to fine-tune most off-the-shelf systems for specific tasks due to the inaccessibility of the model parameters [6]. Secondly, deep neural network models, most of which are built on top of word embeddings, are more sensitive to the difference of features across domain, and thus the model performance is heavily relied on the labeled data in the target domain for fine-tuning [13]. Finally, a relevant work to ours can only assign an intensity to each instance, and it is sensitive to the difference between negations, downtoners, and amplifiers [4].

3 PROPOSED MODEL

In this section, we first formulate the problems of sentiment classification and emotion distribution learning across domains, and denote the frequently-used notations. We then describe how to capture the intrinsic association between document clusters and word clusters across domains based on a joint NMTF framework, and introduce several constraints to incorporate the supervision and the label dependency from both domains. By deriving from the objective function, we propose an inference algorithm and theoretically prove its convergence.

3.1 Formulation and Challenges

Given a source domain and a target domain, we assume that all documents in the source domain D_{src} and a small proportion of documents in the target domain D_{tar} are labeled. Furthermore, the collections of labeled and unlabeled documents in D_{tar} are denoted by D_{ltar} and D_{utar}, respectively. In sentiment classification, each labeled document is tagged with a positive or negative polarity. For emotion distribution learning, each labeled document is assigned with multiple emotional categories and their strengths. The task of cross-domain sentiment classification is to learn a model from labeled documents in D_{src} and D_{ltar}, so as to predict the sentiment polarity of each unlabeled document in D_{utar}. Similarly, the goal of cross-domain emotion distribution learning is to utilize the information in D_{src} and D_{ltar} to predict the strengths over all emotions as closer as the truth scores for each unlabeled document in D_{utar}. We list the frequently-used notations in Table 1, where $\mathbb{R}_+$ represents the field of non-negative real numbers.

In the context of text processing, the NMTF model seeks a decomposition of the document-word matrix into three non-negative latent factor matrices, i.e., the matrix of document clusters, the matrix of word clusters, and the association between document clusters and word clusters. An important merit of using NMTF for transfer learning is that the documents from the source and target domains share the same space of word features and the same set of document labels [56]. Moreover, although the instances in both document clusters and word clusters can be different across domains, the association between document clusters and word clusters remains stable [56]. However, the existing NMTF models lead to the following deficiencies. First, the number of document clusters is often enforced to be equal to the number of categories, and there exists a one-to-one mapping between document clusters and labels [56]. Second, text emotions are not independent but correlated with each other, and the dependency of them is demonstrated to be useful for boosting the model performance [33], yet the incorporation of emotional dependencies in an NMTF model is mathematically arduous. This is impractical due to that each document cluster can trigger multiple sentiment polarities or emotions, and one sentiment polarity or emotion can be associated with many document clusters.

3.2 Our Solution

3.2.1 Objective Function. The traditional two-factor NMF is restrictive as it requires the cluster numbers of documents and words to be equal. In light of this consideration, NMTF introduces a latent matrix $\boldsymbol{S}$ which provides increased degrees of freedom to absorb the different scales of

Table 1. The frequently-used notations.

Notation	Description
D_{src}	All documents in the source domain
D_{tar}	All documents in the target domain
D_{ltar}	Labeled documents in the target domain
D_{utar}	Unlabeled documents in the target domain
n_s	The number of documents in the source domain
n_t	The number of documents in the target domain
d_s	The number of non-repetitive words in the source domain
d_t	The number of non-repetitive words in the target domain
l	The number of labels
c	The number of document clusters
m	The number of word clusters
$\boldsymbol{X_s} \in \mathbb{R}_+^{n_s \times d_s}$	The document-word matrix of the source domain
$\boldsymbol{X_t} \in \mathbb{R}_+^{n_t \times d_t}$	The document-word matrix of the target domain
$\boldsymbol{E_s} \in \mathbb{R}_+^{n_s \times l}$	The label indicator matrix of the source domain
$\boldsymbol{E_t} \in \mathbb{R}_+^{n_t \times l}$	The label indicator matrix of the target domain
$\boldsymbol{F_s} \in \mathbb{R}_+^{n_s \times c}$	The document-cluster indicator matrix of the source domain
$\boldsymbol{F_t} \in \mathbb{R}_+^{n_t \times c}$	The document-cluster indicator matrix of the target domain
$\boldsymbol{G_s} \in \mathbb{R}_+^{d_s \times m}$	The word-cluster indicator matrix of the source domain
$\boldsymbol{G_t} \in \mathbb{R}_+^{d_t \times m}$	The word-cluster indicator matrix of the target domain
$\boldsymbol{C_t} \in \mathbb{R}_+^{n_t \times n_t}$	The label masking matrix for the target domain
$\boldsymbol{S} \in \mathbb{R}_+^{c \times m}$	The association matrix between document clusters and word clusters
$\boldsymbol{M} \in \mathbb{R}_+^{c \times l}$	The association matrix between document clusters and labels
$\boldsymbol{Q} \in \mathbb{R}_+^{l \times l}$	The label dependency matrix

the matrix of document clusters and the matrix of word clusters. Furthermore, the latent matrix $\boldsymbol{S}$ captures the intrinsic association between document clusters and word clusters. Such an association remains stable across domains. Following this motivation, the factorization of document-word matrices from both source and target domains is given by:

$$\underset{\boldsymbol{F_s},\boldsymbol{G_s},\boldsymbol{S},\boldsymbol{F_t},\boldsymbol{G_t}}{\operatorname{argmin}} \left\|\boldsymbol{X_s} - \boldsymbol{F_s}\boldsymbol{S}\boldsymbol{G_s}^T\right\|_F^2 + \lambda_t \left\|\boldsymbol{X_t} - \boldsymbol{F_t}\boldsymbol{S}\boldsymbol{G_t}^T\right\|_F^2. \tag{1}$$

In the above, $||\cdot||_F$ is the Frobenius norm of a matrix, and λ_t ($\lambda_t \geq 0$) is a trade-off parameter. The larger the value of the trade-off parameter, the more important the corresponding term is for the objective function. If the value equals to 0, the corresponding term is ignored. $\boldsymbol{X_s} \in \mathbb{R}_+^{n_s \times d_s}$ is the document-word matrix of D_{src}, where n_s and d_s are the numbers of documents and non-repetition words in D_{src}. $\boldsymbol{X_t} \in \mathbb{R}_+^{n_t \times d_t}$ represents the document-word matrix of D_{tar}, where n_t and d_t are the numbers of documents and non-repetition words in D_{tar}. We use two indicator matrices $\boldsymbol{F_s} \in \mathbb{R}_+^{n_s \times c}$ and $\boldsymbol{F_t} \in \mathbb{R}_+^{n_t \times c}$ to represent the potential confidence of a document belonging to each cluster in D_{src} and D_{tar}, where c is the number of document clusters. The reason of setting the same number of document clusters for D_{src} and D_{tar} is as follows: When computing the co-occurrence matrices of the document clusters for the source and target domains respectively, it can be observed that these two matrices are the same [56]. Similarly, the other two indicator matrices $\boldsymbol{G_s} \in \mathbb{R}_+^{d_s \times m}$ and $\boldsymbol{G_t} \in \mathbb{R}_+^{d_t \times m}$ represent clustering membership values of words in D_{src} and D_{tar}, where m is the number of word clusters. $\boldsymbol{S} \in \mathbb{R}_+^{c \times m}$ is a latent matrix that provides increased degrees of freedom

to ensure the accuracy of the low-rank matrix representation. It is noteworthy that S captures the intra-relationships between document clusters and word clusters, and acts as the bridge of knowledge transformation by being shared in the factorization processes of X_s and X_t.

As a semi-supervised domain adaptation task, the key problem is how to effectively utilize a small amount of labeled data from D_{tar} and abundant labeled data from D_{src} for training. In our proposed sNMTF, the labeled data is utilized by two regularization terms, as follows:

$$\underset{F_s,F_t,M}{\operatorname{argmin}} \quad \lambda_{E_s} ||F_sM - E_s||_F^2 + \lambda_{E_t} \operatorname{Tr}\left[(F_tM - E_t)^T C_t (F_tM - E_t)\right], \tag{2}$$

where $E_s \in \mathbb{R}_+^{n_s \times l}$ and $E_t \in \mathbb{R}_+^{n_t \times l}$ contain the sentiment polarity or emotion distribution of each labeled document in D_{src} and D_{tar}, in which, the former is fully used to supervise the decomposition process, while the latter is partially used. $M \in \mathbb{R}_+^{c \times l}$ is a many-to-many mapping matrix that connects document clusters to emotion labels. $\operatorname{Tr}[\cdot]$ is the trace of a matrix. λ_{E_s} ($\lambda_{E_s} \geq 0$) and λ_{E_t} ($\lambda_{E_t} \geq 0$) are trade-off parameters. To exploit labeled documents in D_{tar} for supervision, a masking matrix $C_t \in \mathbb{R}_+^{n_t \times n_t}$ is introduced and pre-defined as a diagonal matrix, where $C_{t(ii)} = 1$ if the label of the i-th document in D_{tar} is observable for training and $C_{t(ii)} = 0$ otherwise. The information of D_{ltar} can then be encoded through the constraint $\operatorname{Tr}\left[(F_tM - E_t)^T C_t (F_tM - E_t)\right]$. To this end, we can collect all documents in the target domain for training while some existing co-clustering methods only consider a small proportion of labeled documents in the target domain (i.e., D_{ltar}). Specifically, all unlabeled documents (i.e., D_{utar}) are used to generate F_t by simply setting the corresponding rows of E_t to zero, thus endowing the model with strong clustering abilities.

As aforementioned, text emotions can be highly correlated. In our model, we use Q to capture the label dependency from the ground truth, and it is used to guide the update of M as follows:

$$\underset{M}{\operatorname{argmin}} \left\|M^T M - Q\right\|_F^2. \tag{3}$$

In the above, Q denotes the pairwise label dependency matrix. For sentiment classification, we use an identity matrix to model the dependency of different sentiment polarities. This is corresponding to the human intuition that people seldom express opposite sentiments at the meantime. In the situation of emotion distribution learning, Q is generated by calculating the pairwise cosine similarity of emotions for items in the union set of E_s and C_tE_t. Particularly, let E and e denote $\begin{bmatrix} E_s \\ C_tE_t \end{bmatrix}$ and $\left[||e_1||_2, ||e_2||_2, \cdots ||e_l||_2\right]$, where $||e_i||_2$ represents the 2-norm of the i-th column of E. Then, $Q = \frac{E^T E}{e \otimes e}$, where $a \otimes b$ denotes the outer product of vectors a and b. The element of the i-th row and the j-th column in Q represents the cosine similarity of the i-th and the j-th emotions. We use the cosine similarity rather than other metrics (e.g., Pearson correlation coefficient) to calculate Q, because the cosine similarity of non-negative emotion strengths must be non-negative, and this ensures that Q is in accordance with the non-negative value of $M^T M$. Since the column in M represents the latent contribution from each document cluster to the corresponding sentiment polarity (or emotion), $M^T M$ reflects the relationship between text labels. $\left\|M^T M - Q\right\|_F^2$ enforces that the text label dependency $M^T M$ should be close to the ground truth Q.

By formulating Eqs. (1-3) into a unified function, the final optimization objective of our sNMTF is given below:

$$\begin{aligned} \Phi = &\left\|X_s - F_sSG_s^T\right\|_F^2 + \lambda_t \left\|X_t - F_tSG_t^T\right\|_F^2 \\ &+ \lambda_{E_s} ||F_sM - E_s||_F^2 + \lambda_Q \left\|M^T M - Q\right\|_F^2 \\ &+ \lambda_{E_t} \operatorname{Tr}\left[(F_tM - E_t)^T C_t (F_tM - E_t)\right], \end{aligned} \tag{4}$$

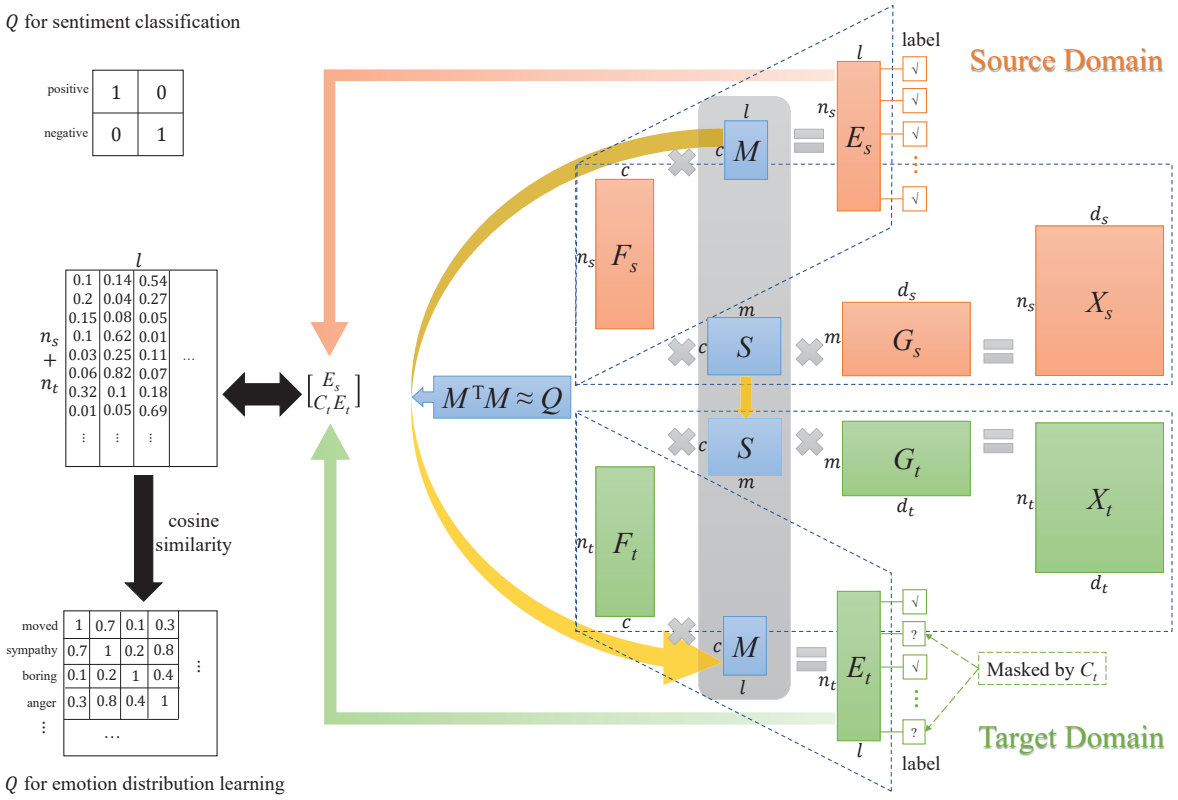


Fig. 2. The modeling process of our method.

where λ_Q ($\lambda_Q \geq 0$) is a trade-off parameter.

After training, we can simply use the cluster-emotion mapping matrix $\boldsymbol{M}$ to predict the sentiment polarity or emotion distribution for every document in D_{utar} by $\boldsymbol{F_t M}$. This is because all documents share the same cluster association matrix, i.e., $\boldsymbol{F_t}$ has the same document cluster index as $\boldsymbol{F_s}$. Our modeling process is illustrated in Figure 2, where the notation "✓" connected to $\boldsymbol{E_s}$ and $\boldsymbol{E_t}$ stands for labeled documents and the notation "?" connected to $\boldsymbol{E_t}$ stands for unlabeled documents.

3.2.2 Convex Dual-Problem. Since the emotion constraint term in Eq. (4) is of fourth-order, the objective function Φ is non-convex for $\boldsymbol{M}$ and is likely to produce sub-optimal results. However, the fundamental step to perform an exact blockwise coordinate descents or other classical inference methods rely on the partition of variables with convexity preserved [45]. To address this issue, we derive a convex dual-problem by variable substitution, as follows:

$$\begin{aligned}\Psi = \Phi &- \lambda_Q \left\| \boldsymbol{M}^T \boldsymbol{M} - \boldsymbol{Q} \right\|_F^2 \\ &+ \lambda_Q \left\| \boldsymbol{M}^T \boldsymbol{A} - \boldsymbol{Q} \right\|_F^2 + \lambda_Q \left\| \boldsymbol{M} - \boldsymbol{A} \right\|_F^2 .\end{aligned} \tag{5}$$

Using such a substitution, the new objective function Ψ is separately convex in each variable, including the introduced auxiliary matrix $\boldsymbol{A}$. Note that $\Phi = \Psi$ ($\boldsymbol{M} = \boldsymbol{A}$).

Lemma 3.1. *The problem* $\Psi^\star = \min \Psi$ *provides a lower bound for the problem* $\Phi^\star = \min \Phi$.

Proof. Let $\boldsymbol{M}^\star$ be the solution to $\Phi^\star$. As for the minimization problem, we have

$$\begin{aligned}\Psi^\star &\leq \min \Psi \left(\boldsymbol{M} = \boldsymbol{M}^\star \right) \\ &\leq \min \Psi \left(\boldsymbol{M} = \boldsymbol{M}^\star, \boldsymbol{A} = \boldsymbol{M}^\star \right) = \Phi^\star .\end{aligned} \tag{6}$$

Since Ψ provides a lower bound of Φ, solving Ψ ensures a parsimonious fit to Φ. □

3.2.3 Inference Algorithm. To solve the new objective function Ψ, we develop an alternately iterative algorithm that provides a good compromise between speed and ease of implementation.

Firstly, the derivative of Ψ on $\boldsymbol{F}_s$ is derived by:

$$\frac{\partial \Psi}{\partial \boldsymbol{F}_s} = -2\boldsymbol{X}_s\boldsymbol{G}_s\boldsymbol{S}^T + 2\boldsymbol{F}_s\boldsymbol{S}\boldsymbol{G}_s^T\boldsymbol{G}_s\boldsymbol{S}^T + 2\lambda_{E_s}\boldsymbol{F}_s\boldsymbol{M}\boldsymbol{M}^T - 2\lambda_{E_s}\boldsymbol{E}_s\boldsymbol{M}^T. \tag{7}$$

Secondly, the update rule of $\boldsymbol{F}_s$ is constructed by placing the negative part of the derivative ($\nabla\Psi$) in the numerator and the positive part in the denominator, as follows:

$$\boldsymbol{F}_s \leftarrow \boldsymbol{F}_s \odot \sqrt{\frac{\boldsymbol{X}_s\boldsymbol{G}_s\boldsymbol{S}^T + \lambda_{E_s}\boldsymbol{E}_s\boldsymbol{M}^T}{\boldsymbol{F}_s\boldsymbol{S}\boldsymbol{G}_s^T\boldsymbol{G}_s\boldsymbol{S}^T + \lambda_{E_s}\boldsymbol{F}_s\boldsymbol{M}\boldsymbol{M}^T}}, \tag{8}$$

where $\odot$ is the element-wise (Hadamard) product.

Thirdly, $\boldsymbol{F}_s$ is normalized to reduce the numerical difficulties like numerical instabilities or ill-conditioning [45], as follows:

$$F_{s(i\cdot)} \leftarrow \frac{F_{s(i\cdot)}}{\sum_{j=1}^{n_s} F_{s(ij)}}, \tag{9}$$

where $F_{s(ij)}$ denotes the item of the i-th row and the j-th column in $\boldsymbol{F}_s$.

A similar process is used to construct the update rules for $\boldsymbol{G}_s$, $\boldsymbol{F}_t$, $\boldsymbol{G}_t$, $\boldsymbol{M}$, and $\boldsymbol{A}$.

Finally, the latent matrix $\boldsymbol{S}$ is updated by:

$$\boldsymbol{S} \leftarrow \boldsymbol{S} \odot \sqrt{\frac{\boldsymbol{F}_s^T\boldsymbol{X}_s\boldsymbol{G}_s + \lambda_t\boldsymbol{F}_t^T\boldsymbol{X}_t\boldsymbol{G}_t}{\boldsymbol{F}_s^T\boldsymbol{F}_s\boldsymbol{S}\boldsymbol{G}_s^T\boldsymbol{G}_s + \lambda_t\boldsymbol{F}_t^T\boldsymbol{F}_t\boldsymbol{S}\boldsymbol{G}_t^T\boldsymbol{G}_t}}. \tag{10}$$

Since all latent factors are updated in the multiplicative form, non-negativity is always satisfied. We summarize our inference method in Algorithm 1.

Algorithm 1: Inference algorithm for the proposed sNMTF

Input: $\boldsymbol{X}_s$, $\boldsymbol{X}_t$, $\boldsymbol{E}_s$, $\boldsymbol{E}_t$, $\boldsymbol{Q}$, and $\boldsymbol{C}_t$;
Output: $\boldsymbol{F}_t$, $\boldsymbol{F}_s$, $\boldsymbol{G}_s$, $\boldsymbol{G}_t$, $\boldsymbol{S}$, and $\boldsymbol{M}$.

```
Predefine the value of threshold ϵ and the maximum number of iterations maxIteration;
Initialize F_s^(0), F_t^(0), G_s^(0), G_t^(0), M^(0), and S^(0) randomly, with each element's value ranging
  between 0 and 1;
A^(0) = M^(0);
i = 0;
repeat
    i = i + 1;
    Compute F_s^(i), F_t^(i), G_s^(i), G_t^(i), M^(i), and A^(i) using the update rule, e.g., Eq. (8);
    Normalize F_s^(i), F_t^(i), G_s^(i), G_t^(i), M^(i), and A^(i);
    Compute S^(i) using the multiplicative rule according to Eq. (10);
until Ψ^(i−1) − Ψ^(i) ≤ ϵ or i ≥ maxIteration;
```

Theorem 3.2. *Algorithm 1 is guaranteed to converge to a locally-optimal solution.*

We demonstrate the convergence of our algorithm below. As an illustration, we prove the convergence of $\boldsymbol{F_s}$ when $\boldsymbol{G_s}$, $\boldsymbol{S}$, $\boldsymbol{F_t}$, $\boldsymbol{G_t}$, $\boldsymbol{M}$, and $\boldsymbol{A}$ are fixed. For the optimization function Ψ with the equality constraints, we formulate the following Lagrangian function on $\boldsymbol{F_s}$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{F_s}) = &\left\|\boldsymbol{X_s} - \boldsymbol{F_s}\boldsymbol{S}\boldsymbol{G_s}^T\right\|_F^2 + \lambda_{E_s}\left\|\boldsymbol{F_s}\boldsymbol{M} - \boldsymbol{E_s}\right\|_F^2 \\ &+ Tr\left[\lambda\left(\boldsymbol{F_s}\boldsymbol{u}^T - \boldsymbol{v}^T\right)\left(\boldsymbol{F_s}\boldsymbol{u}^T - \boldsymbol{v}^T\right)^T\right],\end{aligned} \tag{11}$$

where the Lagrange multiplier $\lambda \in \mathbb{R}_+^{n_s \times n_s}$ is to impose the solution to satisfy the probability representation constraint that $\sum_{j=1}^{c} F_{s_{(ij)}} = 1$. The entry values of $\boldsymbol{u} \in \mathbb{R}_+^{1\times c}$ and $\boldsymbol{v} \in \mathbb{R}_+^{1\times n_s}$ are all equal to 1. To prove the convergence of the update rule Eq. (8), we define an auxiliary function in a way analogous to the EM algorithm [7].

Definition 3.3. $G(x, x')$ is called an auxiliary function for $F(x)$ if the following conditions are satisfied:

$$G(x, x') \geq F(x), G(x, x) = F(x).$$

Lemma 3.4. *If G is an auxiliary function, then F is non-increasing under the following update rule:*

$$x^{t+1} = arg\min_x G\left(x, x^t\right). \tag{12}$$

Proof. $F\left(x^{t+1}\right) \leq G\left(x^{t+1}, x^t\right) \leq G\left(x^t, x^t\right) = F\left(x^t\right)$. □

Clearly, if G is lower bounded and iteratively updated until $F\left(x^{t+1}\right) = F\left(x^t\right)$, x^t becomes a local minimum. The key is to find an appropriate $G(x, x')$. Since all the variables are sequentially updated, it is sufficient to show that the update rule for $\boldsymbol{F_s}$ confirms Eq. (12) for an appropriate auxiliary function.

Lemma 3.5. *Let $x = F_{s(ij)} > 0$ and $x' = F'_{s(ij)} > 0$. The function $G\left(x, x'\right)$ is an auxiliary function for $F(x) = \Psi\left(F_{s(ij)} = x\right)$, as follows:*

$$\begin{aligned}G\left(x, x'\right) = &\left\|\boldsymbol{X_s}\right\|_F^2 + \sum_{ij}\left(\boldsymbol{F'_s}\boldsymbol{S}\boldsymbol{G_s}^T\boldsymbol{G_s}\boldsymbol{S}^T\right)_{ij}\frac{x^2}{x'} \\ &- 2\sum_{ij}\left(\boldsymbol{X_s}\boldsymbol{G_s}\boldsymbol{S}^T\right)_{ij}x'\left(1 + log\frac{x}{x'}\right) \\ &+ \lambda_{E_s}\sum_{ij}\left(\boldsymbol{F'_s}\boldsymbol{M}\boldsymbol{M}^T\right)_{ij}\frac{x^2}{x'} \\ &- 2\lambda_{E_s}\sum_{ij}\left(\boldsymbol{E_s}\boldsymbol{M}^T\right)_{ij}x'\left(1 + log\frac{x}{x'}\right) \\ &+ \left\|\boldsymbol{E_s}\right\|_F^2 + \sum_{ij}\left(\lambda\boldsymbol{F'_s}\boldsymbol{u}^T\boldsymbol{u}\right)_{ij}\frac{x^2}{x'} \\ &- 2\sum_{ij}\left(\lambda\boldsymbol{v}^T\boldsymbol{u}\right)_{ij}x'\left(1 + log\frac{x}{x'}\right).\end{aligned} \tag{13}$$

Proof. Obviously, the equality $G\left(x, x'\right) = \Psi\left(F_{s(ij)} = x\right)$ holds when $\boldsymbol{F'_s} = \boldsymbol{F_s}$. When $\boldsymbol{F'_s} \neq \boldsymbol{F_s}$, the inequality $G\left(x, x'\right) \geq \Psi\left(F_{s(ij)} = x\right)$ also holds and can be proved using the similar proof procedure in [9]. This establishes that G is an auxiliary function for Ψ. □

Lemma 3.6. *Using the update rule given in the following equation, Ψ will monotonously decrease:*

$$\boldsymbol{F_s} \leftarrow \boldsymbol{F_s} \odot \sqrt{\frac{\boldsymbol{X_s}\boldsymbol{G_s}\boldsymbol{S}^T + \lambda_{E_s}\boldsymbol{E_s}\boldsymbol{M}^T + \lambda\boldsymbol{v}^T\boldsymbol{u}}{\boldsymbol{F_s}\boldsymbol{S}\boldsymbol{G_s}^T\boldsymbol{G_s}\boldsymbol{S}^T + \lambda_{E_s}\boldsymbol{F_s}\boldsymbol{M}\boldsymbol{M}^T + \lambda\boldsymbol{F_s}\boldsymbol{u}^T\boldsymbol{u}}}, \tag{14}$$

Proof. To prove that Algorithm 1 converges, we need to show that the update rule of $\boldsymbol{F_s}$ follows Eq. (12). By fixing $\boldsymbol{F_s'}$, we minimize $G\left(x, x^{'}\right)$ as follows:

$$\begin{aligned}\frac{\partial G\left(x, x^{'}\right)}{\partial x} =& -2\left(\boldsymbol{X_s G_s S}^T + \lambda_{E_s}\boldsymbol{E_s M}^T + \lambda \boldsymbol{v}^T\boldsymbol{u}\right)_{ij}\frac{x^{'}}{x} \\ &+2\left(\lambda_{E_s}\boldsymbol{F_s'MM}^T + \lambda \boldsymbol{F_s'u}^T\boldsymbol{u}\right)_{ij}\frac{x}{x^{'}} \\ &+2\left(\boldsymbol{F_s'SG_s}^T\boldsymbol{G_s S}^T\right)_{ij}\frac{x}{x^{'}}.\end{aligned} \tag{15}$$

Solving $\frac{\partial G\left(x,x^{'}\right)}{\partial x} = 0$ for x, we get the update rule in Eq. (14). Since G is the auxiliary function for Ψ, this update rule of $\boldsymbol{F_s}$ monotonically decreases the value of Ψ. □

Note that the Lagrange multiplier λ exists in Eq. (14), so as to make the solution satisfy the probability constraint that the sum of values in each row of $\boldsymbol{F_s}$ is 1. To omit the terms which depend on λ, we propose a normalization technique as follows. In each iteration, we use Eq. (9) to normalize each row in $\boldsymbol{F_s}$. After such a normalization process, term $\boldsymbol{v}^T\boldsymbol{u}$ will be equal to term $\boldsymbol{F_s'u}^T\boldsymbol{u}$. Thus, when we only consider the state of convergence, the effect of Eq. (8) and Eq. (9) can be approximately equivalent to Eq. (14). According to the convergence analysis, using the update rule of Eq. (8) and the normalization process in Eq. (9) will not increase the objective function Ψ. Since Ψ is a second-order convex function in each variable, we can use a similar method to prove the convergence of update rules for $\boldsymbol{G_s}$, $\boldsymbol{F_t}$, $\boldsymbol{G_t}$, $\boldsymbol{S}$, $\boldsymbol{M}$, and $\boldsymbol{A}$. Thus, each update step in Algorithm 1 monotonically decreases Ψ, which guarantees the convergence.

3.3 MPI Implementation

The multiplicative update algorithms mentioned in Section 3.2.3 are implemented for serial computing running on a single machine, and such implementations face two bottlenecks with the growth of text data [5]. On the one hand, the multiplicative update method requires a large number of matrix multiplication operations. From Algorithm 1, we can observe that the update formulas of the latent factor matrices involve multiple matrix multiplication operations on the document-word matrix and other latent factor matrices. When the scale of dataset grows, the dimension of these matrices will increase correspondingly. Performing continuous matrix multiplication operations on these high-dimensional matrices will greatly increase the computational cost, thus limiting the performance of the proposed model on large-scale corpora. On the other hand, the serial computing algorithm which runs on a single machine requires all involved data to be stored in this compute node for the updating process. However, the memory of a single compute node is limited. For a large-scale dataset, the node's memory space may fail to support the storage requirements of all data matrices in Algorithm 1, which further limits the model scalability. To address these issues, we here propose a novel distributed algorithmic framework for Algorithm 1.

For a distributed environment, we assume that it contains p available compute nodes or processors. Each node only stores a few blocks of the high-dimensional data matrix and is only responsible for the computation involving its own data matrix blocks. Given any data matrix $\boldsymbol{X} \in \mathbb{R}_+^{n\times d}$, we divide it into p row blocks by processing rows of the matrix and denote the row blocks as $\boldsymbol{X_{row}} \in \mathbb{R}_+^{n_p\times d}$, where n_p stands for the size of row blocks in each compute node. Note that every compute node has its ID, ranging from 0 to $p-1$. We define $n_p = \left\lceil \frac{n}{p} \right\rceil$ if the ID of the node is smaller than $(n \mod p)$; else $n_p = \left\lfloor \frac{n}{p} \right\rfloor$, where $\lceil x \rceil$ and $\lfloor x \rfloor$ represent rounding up and rounding down x to an integer, respectively. Similarly, we divide the data matrix into p column blocks by processing

columns of the matrix and denote the column blocks as $\boldsymbol{X_{col}} \in \mathbb{R}_+^{n \times d_p}$ with d_p calculated in the similar way.

Aiming at the task of sentiment analysis and emotion distribution learning across domains, the numbers of documents and non-repetitive words in the source and target domains, i.e., n_s, n_t, d_s, and d_t, are far larger than other parameters such as the number of document clusters c, the number of word clusters m, and the number of labels l. Thus the space complexity of Algorithm 1 is mainly attributed to the high-dimensional matrices with one dimension being n_s, n_t, d_s, or d_t, and we mainly consider the distribution of these matrices. Specifically, each compute node only stores the row blocks $\boldsymbol{X_{srow}} \in \mathbb{R}_+^{n_{sp} \times d_s}$, $\boldsymbol{X_{trow}} \in \mathbb{R}_+^{n_{tp} \times d_t}$, $\boldsymbol{E_{srow}} \in \mathbb{R}_+^{n_{sp} \times l}$, and $\boldsymbol{E_{trow}} \in \mathbb{R}_+^{n_{tp} \times l}$ of the raw high-dimensional matrices. Similarly, each compute node stores row blocks of matrices $\boldsymbol{F_{srow}} \in \mathbb{R}_+^{n_{sp} \times c}$, $\boldsymbol{F_{trow}} \in \mathbb{R}_+^{n_{tp} \times c}$, $\boldsymbol{G_{srow}} \in \mathbb{R}_+^{d_{sp} \times m}$, $\boldsymbol{G_{trow}} \in \mathbb{R}_+^{d_{tp} \times m}$. In particular, we need the extra column blocks $\boldsymbol{X_{scol}} \in \mathbb{R}_+^{n_s \times d_{sp}}$ and $\boldsymbol{X_{tcol}} \in \mathbb{R}_+^{n_t \times d_{tp}}$. The diagonal matrix $\boldsymbol{C_t}$ can be stored as a diagonal vector and each compute node owns a diagonal vector of length n_{tp} to represent the data of blocks $\boldsymbol{C_{tp}} \in \mathbb{R}_+^{n_{tp} \times n_{tp}}$. On the contrary, the whole copies of the matrices $\boldsymbol{S} \in \mathbb{R}_+^{c \times m}$ and $\boldsymbol{M} \in \mathbb{R}_+^{c \times l}$ are stored by each compute node because the values of c, m, and l are much smaller than other matrix dimensions. Our distributed algorithm could keep the copies of $\boldsymbol{S}$ and $\boldsymbol{M}$ consistent on every compute node during the update procedure.

We observe that the update rule of the latent factor matrices, e.g., Eq. (8), involves element-wise product and element-wise division. Therefore, for any update rule in the form of Eq. (8), the update of row blocks could be formulated as follows:

$$\boldsymbol{X_{row}} \leftarrow \boldsymbol{X_{row}} \odot \sqrt{\frac{\boldsymbol{N_{row}}}{\boldsymbol{D_{row}}}}, \tag{16}$$

where $\boldsymbol{N_{row}}$ and $\boldsymbol{D_{row}}$ stand for row blocks of numerator matrix $\boldsymbol{N}$ and row blocks of denominator matrix $\boldsymbol{D}$, respectively.

Take Eq. (8) as an example, $\boldsymbol{N}$ is obtained by $\boldsymbol{X_s G_s S^T} + \lambda_{E_s} \boldsymbol{E_s M^T}$, and $\boldsymbol{D}$ is obtained by $\boldsymbol{F_s S G_s^T G_s S^T} + \lambda_{E_s} \boldsymbol{F_s M M^T}$. After obtaining the row blocks of $\boldsymbol{N}$ and $\boldsymbol{D}$, we can update corresponding row blocks of the parameter matrix according to Eq. (16), i.e., $\boldsymbol{F_{srow}}$. In this way, we develop a distributed update procedure of Eq. (8), as shown in Algorithm 2.

Algorithm 2: Distributed update procedure for the document-cluster indicator matrix of the source domain $\boldsymbol{F_s}$

Input: $\boldsymbol{F_{srow}}$, $\boldsymbol{X_{scol}}$, $\boldsymbol{E_{srow}}$, $\boldsymbol{G_{srow}}$, $\boldsymbol{S}$, and $\boldsymbol{M}$.
Output: $\boldsymbol{F_{srow}}$.

1 **foreach** *compute node* **do**
2 $\quad \widetilde{\boldsymbol{G_{srow}}} \leftarrow \boldsymbol{G_{srow} S^T}$;
3 $\quad \boldsymbol{N_{row}} \leftarrow \text{Reduce\_Scatter}\left(\boldsymbol{X_{scol}} \widetilde{\boldsymbol{G_{srow}}}, \text{sum}\right)$;
4 $\quad \boldsymbol{N_{row}} \leftarrow \boldsymbol{N_{row}} + \lambda_{E_s} \boldsymbol{C_{sp} E_{srow} M^T}$;
5 $\quad \boldsymbol{T} \leftarrow \text{Allreduce}\left(\widetilde{\boldsymbol{G_{srow}}}^T \widetilde{\boldsymbol{G_{srow}}}, \text{sum}\right)$;
6 $\quad \boldsymbol{D_{row}} \leftarrow \boldsymbol{F_{srow} T} + \lambda_{E_s} \boldsymbol{C_{sp} F_{srow} M M^T}$;
7 $\quad \boldsymbol{F_{srow}} \leftarrow \boldsymbol{F_{srow}} \circ \sqrt{\frac{\boldsymbol{N_{row}}}{\boldsymbol{D_{row}}}}$;
8 **end**

In Algorithm 2, the matrix $\boldsymbol{T}$ is a temporary matrix with size of $c \times c$. Particularly, the function Reduce_Scatter$(\boldsymbol{X}, \text{sum})$ represents the operation consisting of adding up all blocks of $\boldsymbol{X}$ on

every compute node and then scattering corresponding block of $\boldsymbol{X}$ to each node. The function Allreduce($\boldsymbol{X}$, sum) represents the operation consisting of adding up all blocks of $\boldsymbol{X}$ on every compute node and then scattering the copy of $\boldsymbol{X}$ to each node. Utilizing these operations provided by MPI, the proposed Algorithm 2 can achieve a distributed update procedure for the document-cluster indicator matrix of the source domain $\boldsymbol{F_s}$, in which, each compute node only needs to store the small block of high-dimensional matrix rather than the whole matrix. Similarly, we develop distributed update procedures for $\boldsymbol{G_s}$, $\boldsymbol{F_t}$, $\boldsymbol{G_t}$ in Algorithms 3, 4, and 5, respectively. The matrix $\boldsymbol{U}$ is also a temporary matrix with size of $m \times m$.

Algorithm 3: Distributed update procedure for the word-cluster indicator matrix of the source domain $\boldsymbol{G_s}$

Input: $\boldsymbol{G_{srow}}$, $\boldsymbol{X_{srow}}$, $\boldsymbol{F_{srow}}$, and $\boldsymbol{S}$.
Output: $\boldsymbol{G_{srow}}$.
foreach *compute node* **do**
  $\widetilde{\boldsymbol{F_{srow}}} \leftarrow \boldsymbol{F_{srow}}\boldsymbol{S}$;
  $\boldsymbol{N_{row}} \leftarrow \text{Reduce\_Scatter}\left(\boldsymbol{X_{srow}^T}\widetilde{\boldsymbol{F_{srow}}}, \text{sum}\right)$;
  $\boldsymbol{U} \leftarrow \text{Allreduce}\left(\widetilde{\boldsymbol{F_{srow}}}^T\widetilde{\boldsymbol{F_{srow}}}, \text{sum}\right)$;
  $\boldsymbol{D_{row}} \leftarrow \boldsymbol{G_{srow}}\boldsymbol{U}$;
  $\boldsymbol{G_{srow}} \leftarrow \boldsymbol{G_{srow}} \circ \sqrt{\frac{\boldsymbol{N_{row}}}{\boldsymbol{D_{row}}}}$;
end

Algorithm 4: Distributed update procedure for the document-cluster indicator matrix of the target domain $\boldsymbol{F_t}$

Input: $\boldsymbol{F_{trow}}$, $\boldsymbol{X_{tcol}}$, $\boldsymbol{E_{trow}}$, $\boldsymbol{G_{trow}}$, $\boldsymbol{S}$, and $\boldsymbol{M}$.
Output: $\boldsymbol{F_{trow}}$.
foreach *compute node* **do**
  $\widetilde{\boldsymbol{G_{trow}}} \leftarrow \boldsymbol{G_{trow}}\boldsymbol{S}^T$;
  $\boldsymbol{N_{row}} \leftarrow \text{Reduce\_Scatter}\left(\boldsymbol{X_{tcol}}\widetilde{\boldsymbol{G_{trow}}}, \text{sum}\right)$;
  $\boldsymbol{N_{row}} \leftarrow \boldsymbol{N_{row}} + \lambda_{E_s}\boldsymbol{C_{sp}}\boldsymbol{E_{trow}}\boldsymbol{M}^T$;
  $\boldsymbol{T} \leftarrow \text{Allreduce}\left(\widetilde{\boldsymbol{G_{trow}}}^T\widetilde{\boldsymbol{G_{trow}}}, \text{sum}\right)$;
  $\boldsymbol{D_{row}} \leftarrow \boldsymbol{F_{trow}}\boldsymbol{T} + \lambda_{E_s}\boldsymbol{C_{sp}}\boldsymbol{F_{trow}}\boldsymbol{M}\boldsymbol{M}^T$;
  $\boldsymbol{F_{trow}} \leftarrow \boldsymbol{F_{trow}} \circ \sqrt{\frac{\boldsymbol{N_{row}}}{\boldsymbol{D_{row}}}}$;
end

With respect to the distributed update procedure of matrix $\boldsymbol{S}$, we reduce $\boldsymbol{F_s^T}\boldsymbol{X_s}\boldsymbol{G_s}$, $\boldsymbol{F_t^T}\boldsymbol{X_t}\boldsymbol{G_t}$, $\boldsymbol{F_s^T}\boldsymbol{F_s}$, $\boldsymbol{G_s^T}\boldsymbol{G_s}$, $\boldsymbol{F_t^T}\boldsymbol{F_t}$, and $\boldsymbol{G_s^T}\boldsymbol{G_t}$ to every compute node by utilizing the Reduce_Scatter operation and the Allreduce operation. As a consequence, every compute node keeps consistent with each other. The update procedure is presented in Algorithm 6. For the association matrix between document clusters and labels $\boldsymbol{M}$, we have introduced an auxiliary matrix $\boldsymbol{A}$ to preserve convexity for each variable. The distributed update procedures of $\boldsymbol{M}$ and $\boldsymbol{A}$ are presented in Algorithms 7 and 8. Each compute node maintains a copy of $\boldsymbol{M}$ or $\boldsymbol{A}$ during their own update procedures.

Algorithm 5: Distributed update procedure for the word-cluster indicator matrix of the target domain G_t

Input: G_{trow}, X_{trow}, F_{trow}, and S.
Output: G_{trow}.

foreach *compute node* **do**
- $\widetilde{F_{trow}} \leftarrow F_{trow}S$;
- $N_{row} \leftarrow \text{Reduce\_Scatter}\left(X_{trow}^T \widetilde{F_{trow}}, \text{sum}\right)$;
- $U \leftarrow \text{Allreduce}\left(\widetilde{F_{trow}}^T \widetilde{F_{trow}}, \text{sum}\right)$;
- $D_{row} \leftarrow G_{trow}U$;
- $G_{trow} \leftarrow G_{trow} \circ \sqrt{\frac{N_{row}}{D_{row}}}$;

end

Algorithm 6: Distributed update procedure for the association matrix between document clusters and word clusters S

Input: F_{srow}, G_{srow}, X_{srow}, F_{trow}, G_{trow}, X_{trow}, and S.
Output: S.

foreach *compute node* **do**
- $K_s \leftarrow \text{Reduce\_Scatter}\left(X_{srow}^T F_{srow}, \text{sum}\right)$;
- $A \leftarrow \text{Allreduce}\left(K_s^T G_{srow}, \text{sum}\right)$;
- $K_t \leftarrow \text{Reduce\_Scatter}\left(X_{trow}^T F_{trow}, \text{sum}\right)$;
- $A \leftarrow A + \lambda_t \text{Allreduce}\left(K_t^T G_{trow}, \text{sum}\right)$;
- $F_s^T F_s \leftarrow \text{Allreduce}\left(F_{srow}^T F_{srow}, \text{sum}\right)$;
- $G_s^T G_s \leftarrow \text{Allreduce}\left(G_{srow}^T G_{srow}, \text{sum}\right)$;
- $D \leftarrow F_s^T F_s S G_s^T G_s$;
- $F_t^T F_t \leftarrow \text{Allreduce}\left(F_{trow}^T F_{trow}, \text{sum}\right)$;
- $G_t^T G_t \leftarrow \text{Allreduce}\left(G_{trow}^T G_{trow}, \text{sum}\right)$;
- $D \leftarrow D + \lambda_t F_t^T F_t S G_t^T G_t$;
- $S \leftarrow S \circ \sqrt{\frac{A}{D}}$;

end

Algorithm 7: Distributed update procedure for the association matrix between document clusters and labels M

Input: F_{srow}, G_{srow}, X_{srow}, F_{trow}, G_{trow}, X_{trow}, S, and A.
Output: S.

foreach *compute node* **do**
- $N \leftarrow \text{Allreduce}\left(\lambda_{E_s} F_{srow}^T C_{sp} E_{srow} + \lambda_{E_t} F_{trow}^T C_{tp} E_{trow}, \text{sum}\right)$;
- $N \leftarrow N + \lambda_Q A Q^T + \lambda_Q A$;
- $T \leftarrow \text{Allreduce}\left(\lambda_{E_s} F_{srow}^T C_{sp} F_{srow} + \lambda_{E_t} F_{trow}^T C_{tp} F_{trow}, \text{sum}\right)$;
- $D \leftarrow TM + \lambda_Q A A^T M + \lambda_Q M$;
- $M \leftarrow M \circ \sqrt{\frac{N}{D}}$;

end

Algorithm 8: Distributed update procedure for the auxiliary matrix A

Input: F_{srow}, G_{srow}, X_{srow}, F_{trow}, G_{trow}, X_{trow}, S, and A
Output: S
foreach *compute node* **do**
  $N \leftarrow MQ + M$;
  $D \leftarrow MM^T A + A$;
  $A \leftarrow A \circ \sqrt{\frac{N}{D}}$;
end

The normalization operation in our serial Algorithm 1 can be extended naturally to a distributed procedure because each compute node contains the row block of the required matrices and could normalize row blocks directly after updating factor matrices. Our final distributed algorithm based on MPI is summarized in Algorithm 9.

Algorithm 9: MPI based distributed algorithm for the proposed sNMTF

Input: X_s, X_t, E_s, E_t, C_s, and C_t.
Output: F_s, F_t, G_s, G_t, S, and M.
if *The current node is the master node* **then**
  Initialize S and M randomly;
  Initialize $A = M$;
  Broadcast S, M, and A to each compute node;
end
foreach *compute node* **do**
  Read X_{srow}, X_{trow}, X_{scol}, X_{tcol}, E_{srow}, E_{trow}, C_{sp}, and C_{tp} ;
  Initialize F_{srow}, F_{trow}, G_{srow}, and G_{trow} randomly ;
end
repeat
  foreach *compute node* **do**
    Update F_s according to Algorithm 2 ;
    Update G_s according to Algorithm 3 ;
    Update F_t according to Algorithm 4 ;
    Update G_t according to Algorithm 5 ;
    Update M according to Algorithm 7 ;
    Update A according to Algorithm 8 ;
    Normalize F_s, F_t, G_s, G_t, M, and A ;
    Update S according to Algorithm 6 ;
  end
until *terminate criterion is met*;

3.4 Computational Complexity

To systematically estimate the computational complexity of our distributed algorithm, we compare the space complexity and time complexity between the serial algorithm (i.e., Algorithm 1) and the distributed algorithm (i.e., Algorithm 9). Furthermore, we introduce the $\alpha - \beta - \gamma$ model [15] to

Table 2. Complexity comparison on a compute node per iteration between distributed and serial algorithms, i.e., Algorithm 9 and Algorithm 1.

Algorithm		Complexity
Distributed	Space	$O\left[\frac{n_s d_s+n_t d_t+(n_s+n_t+d_s+d_t)(c+m)+(n_s+n_t)l}{p}+ml+cl\right]$
	Time	$O\left[\frac{(n_s d_s+n_t d_t)(c+m)+(n_s+d_s+n_t+d_t)\left(c^2+m^2+cm\right)+n_s lc}{p}+lc^2+c^2m+cm^2\right]$
	Communication	$26\alpha\log p+\beta\frac{p-1}{p}\left[(n_s+n_t+d_s+d_t)\,c+(d_s+d_t)\,m+6c^2+6m^2+4cm\right]$
Serial	Space	$O\left[n_s d_s+n_t d_t+(n_s+n_t+d_s+d_t)\,(c+m)+(n_s+n_t)\,l+ml+cl\right]$
	Time	$O\Big[(n_s d_s+n_t d_t)\,(c+m)+(n_s+d_s+n_t+d_t)\left(c^2+m^2+cm\right)+n_s lc+lc^2+c^2m+cm^2\Big]$

analyze the communication complexity of the distributed Algorithm 9. In the $\alpha-\beta-\gamma$ model, every node adopts a two-way message passing strategy to communicate with each other. The cost of passing a message with n words is defined as $\alpha+n\beta$, where α stands for the latency cost per message and β stands for the bandwidth cost per word. Each node executes floating point operations (flop) on data in the local memory and the computation cost is γ per flop. Accordingly, we focus on the times of flop computation on each node, the times of communication between nodes, and the size of message delivered. For data of size n, the function Reduce_Scatter provided by MPI results in the communication complexity of $\left[\alpha\cdot\log p+(\beta+\gamma)\cdot\frac{p-1}{p}n\right]$ and the function Allreduce results in the communication complexity of $\left[2\alpha\cdot\log p+(2\beta+\gamma)\cdot\frac{p-1}{p}n\right]$. We summarize the comparison of computational complexity on Algorithm 1 and Algorithm 9 in Table 2, where all values refer to the complexity on a single compute node per iteration. It indicates that the space complexity of the update algorithm mainly comes from the storage of factor matrices and text data matrices, because the numbers of document clusters and word clusters, i.e., c and m, are much smaller than the numbers of documents and words in most cases. Besides, the time complexity mainly comes from the matrix multiplication involving the matrices with one of the dimensions being n_s, n_t, d_s, or d_t. For our distributed algorithm, although each node redundantly retains the whole matrices $\boldsymbol{S}$ and $\boldsymbol{M}$, it can greatly reduce the main complexity of each node by the distributed storage and parallel computation of large matrices.

4 EXPERIMENTS

In this section, we conduct experimental evaluations to answer the following questions.

- Q1. Can our models achieve better performances than the existing models? (Section 4.3)
- Q2. Is a many-to-many mapping between document clusters and labels eligible for sentiment classification and emotion distribution learning across domains? (Section 4.4)
- Q3. How does the label dependency matrix influence the model performance? (Section 4.5)
- Q4. How does the number of document clusters influence the model performance? (Section 4.6)
- Q5. Can our distributed algorithm scale the model to larger corpora effectively? (Section 4.7)

4.1 Datasets and Settings

For cross-domain sentiment classification, we employ the widely used Amazon product review dataset [2]. This dataset includes four domains: Books (B), DVD (D), Electronics (EL), and Kitchen (K). For each domain, there are 1000 positive reviews and 1000 negative reviews. Furthermore, each review is tagged as one sentiment polarity (i.e., positive or negative) only. It is noteworthy that the mapping between document clusters and sentiment polarities might be many-to-many. By checking manually, there is a document cluster with approximately 50 reviews about "CD" in the EL domain. Within this cluster, 27 reviews are positive (e.g., "These mini CD-Rs work great in our Sony Mavica CD Cameras at a cost of only .42 cents apiece") and 23 reviews are negative (e.g., "I just recently bought this and when I installed it Norton caught 2 viruses on the install CD"). Meanwhile, each sentiment polarity may be associated with multiple document clusters. For instance, the negative reviews could be covered in document clusters of "CD" and "Mouse". The dataset is preprocessed as follows: Firstly, we remove the numeric characters, punctuations, stop words, and low frequency words which occur in 5 or less reviews for each domain. Then, we construct bag-of-words features and represent each pre-processed document by the $tf - idf$ weighting scheme. The dataset statistics are listed in Table 3, including the numbers of positive reviews, negative reviews, and non-repetitive features.

For cross-domain emotion distribution learning, we employ the real-world ChinaNews dataset with news articles and emotion voting numbers of readers [30]. Each news article was published by a specific channel which could be referred as a domain (e.g., Economics and Culture), and was voted by online readers over 8 basic emotions, including "moved", "sympathy", "boring", "anger", "funny", "sadness", "delighted", and "not-interested". Following [30], we use four domains which contain top numbers of documents for evaluation: Economics (EC), Culture (C), Law (L), and Society (S). The dataset is preprocessed as follows: Firstly, articles that have won only a few number of votes are not popular and tend to be unimportant for our study, because the limited votes do not represent the attitudes of the public. For this reason, we discard those articles with less than 5 votes, i.e., articles with the rating number over all emotions less than 5 are filtered. Secondly, the title and main body of each article are extracted and represented as a single document. Then, each document is preprocessed following the steps of tokenization[1], filtering out stop words and non-Chinese characters, and discarding rare words that occur in 10 or less documents to reduce feature dimensions. Thirdly, we use the raw bag-of-words features and characterize the document in each domain by the $tf - idf$ weighting scheme. For reader ratings, we follow [53] to obtain the emotion distribution of every instance since each news article only contains the voting numbers over those emotions. Specifically, we use n_d^j to represent the voting number of the j-th emotion and r_d^j to represent the rating of the j-th emotion for document d, then we have $r_d^j = \frac{n_d^j}{\sum_k n_d^k}$. Note that r_d^j is the proportion that emotion j accounts for document d, rather than the probability that j correctly labels d. To constitute emotion distributions, the emotion rating is normalized to $r_d^j \in [0, 1]$ and $\sum_j r_d^j = 1$. The dataset statistics are shown in Table 4, including the numbers of news articles, reader ratings, and non-repetitive features.

To make effective comparisons, we construct 12 domain adaptation tasks for each of the two datasets. For example, B→D is one of the cross-domain sentiment classification tasks, where B denotes the source domain of Books and D corresponds to the target domain of DVD. EC→C is one of the cross-domain emotion distribution learning tasks, where EC denotes the source domain of Economics, and C corresponds to the target domain of Culture. For each method, all documents in the source domain and 10 percent documents in the target domain are used for training. Furthermore,

[1] https://github.com/fxsjy/jieba/

Table 3. Statistics of the Amazon dataset for sentiment classification.

Domains	#Positive	#Negative	#Features
Books (B)	1,000	1,000	3,893
Electronics (EL)	1,000	1,000	2,280
DVD (D)	1,000	1,000	3,977
Kitchen (K)	1,000	1,000	2,094

Table 4. Statistics of the ChinaNews dataset for emotion distribution learning.

Domains	#News	#Ratings	#Features
Economics (EC)	1,554	58,111	4,835
Culture (C)	1,241	39,889	5,974
Law (L)	1,963	250,657	7,064
Society (S)	1,602	206,260	5,719

we use another 10 percent documents and the remaining 80 percent documents in the target domain as the validation set and the testing set, respectively.

4.2 Models and Metrics

For sentiment classification, we list our models and baselines as follows:

- Semi-supervised Non-negative Matrix Tri-Factorization (sNMTF) is the proposed framework for cross-domain sentiment classification, which uses an identity matrix as the pairwise label dependency matrix. In this method, all latent matrices are initialized randomly.
- msNMTF is a degraded version of sNMTF. By setting $\lambda_Q = 0$, msNMTF ignores the constraint of pairwise label dependency.
- Bidirectional Encoder Representation from Transformers with Support Vector Machine (BERT-SVM) firstly represents each document from the pre-trained BERT [8]. Then, it utilizes SVM for cross-domain sentiment classification. This method is developed in a similar way by following a baseline adopted in [52], and it is originally used for single-domain tasks. For fair comparison, we train the above method on the combination of training instances from the source domain and 10 percent labeled instances from the target domain, and evaluate it on the same testing set as other semi-supervised cross-domain models.
- Domain Adaptive Semi-supervised Learning (DAS)[2] [12] minimizes the KL divergence of the extracted features between source and target domains and utilizes additional label information in the target domain for improving the performance of cross-domain sentiment classification. It uses pre-trained Glove word embeddings [29] through an external large corpus.
- Deep Adversarial Mutual Learning (DAML)[3] [48] adopts a deep adversarial mutual learning method which involves two groups that consist of feature extractors and classifiers, and the two groups could improve the effectiveness by teaching each other. Similar to DAS, it also uses pre-trained Glove word embeddings [29] as the input.

We select trade-off hyperparameters of the model loss, i.e., λ_t, λ_{E_s}, λ_{E_t}, and λ_Q in our sNMTF and msNMTF, λ_1, λ_2, and λ_3 in DAS, in addition to η, λ_D, and λ_M in DAML, by grid searching

[2]https://github.com/ruidan/DAS
[3]https://github.com/SleepyBag/DAML

on the validation set under the range of $\{0.01, 1, 5, 20\}$. The numbers of document clusters and word clusters in our models are searched via $\{20, 50, 100, 150, 200, 300\}$. Note that most of the above models are initialized randomly, so we run each model on every domain pair 10 times independently.

For emotion distribution learning, we group baselines and our models into three categories, i.e., single-domain, unsupervised cross-domain, and semi-supervised cross-domain.

Single-domain Models. The following models are originally developed for single-domain emotion distribution learning. For fair comparison, all instances from the source domain and 10 percent labeled instances from the target domain are used as the training set in our experiment, which is the same as the aforementioned training set used by semi-supervised cross-domain models. For these models, the other 10 percent and 80 percent instances from the target domain are employed as the validation set and the testing set, respectively.

- Multi-Task Convolutional Neural Network (MTCNN) [50] establishes a multi-task end-to-end learning framework based on convolution neural network by conducting text classification and emotion distribution learning simultaneously. It uses the publicly available word embeddings from word2vec [22] trained on the external Chinese Wikipedia corpus.
- Dependency Embedded Recursive Neural Network (DERNN) [43] integrates the syntactic dependency and document topic information to learn the emotion distribution. Similar to MTCNN, the word embeddings are also obtained by word2vec [22].
- Bidirectional Encoder Representation from Transformers with Linear Regression (BERT-LR) is a strong baseline adopted in [52]. This model firstly represents documents by the pre-trained BERT [8]. Then, it utilizes the linear regression model for emotion distribution learning.
- Topic Enhanced Self Attention Network (TESAN) [42] firstly extracts the embedding representation of topics by a neural topic model and the embedding representation of documents by a self-attention network, respectively. Then, it combines the topic information and the document feature through a fusion gate. This method utilizes word2vec [22] to train word embeddings from the Chinese Wikipedia corpus.

Unsupervised Cross-domain Models. The following models are proposed for unsupervised cross-domain emotion distribution learning, i.e., they do not exploit any label information from the target domain originally. For these models, we combine 10 percent labeled instances from the target domain with all instances from the source domain for training, so as to be consistent with semi-supervised cross-domain models.

- Contextual Sentiment Topic Model (CSTM) [35] firstly extracts background topics, context specific topics, and context independent topics across domains. Then, the context independent topics are used to learn the document's emotion distributions in the target domain.
- Cross-Domain Text Classification (CDTC) model [56] learns the joint probability distribution of text features and emotion categories to model the difference between domains based on NMTF. By introducing a shared matrix, the knowledge learned in the source domain could be transferred to the target domain. To apply CDTC which was originally developed for cross-domain text classification to our tasks, we use $\boldsymbol{G_t}$ as the prediction results for this model. Furthermore, the labeled data in the target domain are incorporated into CDTC under the control of $\boldsymbol{C_t}$ for fair comparisons.
- Triplex Transfer Learning (TriTL) [55] models the domain sharing concepts and domain specific concepts among different domains, so as to improve the performance of cross-domain emotion distribution learning based on NMTF.

- Constrained NMTF (cNMTF) [30] groups similar documents into the same clusters and learns the many-to-many mapping relationship between document clusters and labels.

Semi-supervised Cross-domain Models. The following emotion distribution learning models are developed for semi-supervised cross-domain tasks. By exploiting all labeled instances in the source domain and 10 percent labeled instances in the target domain for training, these models could effectively utilize the limited labeled instances from the target domain to boost performance.

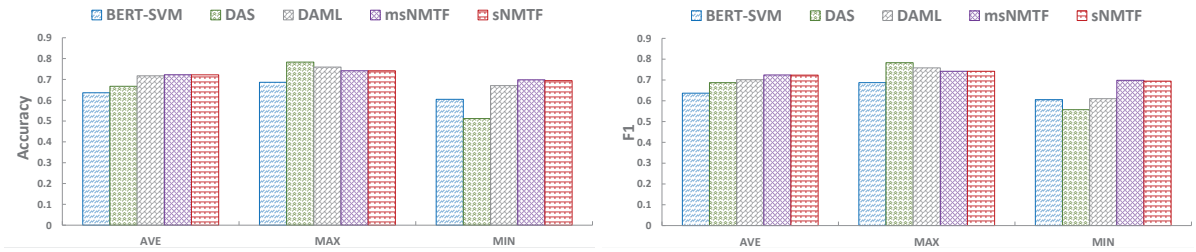
- Semi-supervised Non-negative Matrix Tri-Factorization with Source and Target domain labels (sNMTF-st) is the proposed framework for emotion distribution learning. It utilizes all labels from the source domain and available labels from the target domain, i.e., the union set of $\boldsymbol{E_s}$ and $\boldsymbol{C_t E_t}$, to generate the pairwise label dependency matrix $\boldsymbol{Q}$.
- sNMTF-s denotes the proposed framework which only utilizes the label information from the source domain to generate $\boldsymbol{Q}$.
- sNMTF-t denotes the proposed framework which only utilizes the available label information from the target domain to generate $\boldsymbol{Q}$.
- msNMTF is a degraded version of our method by setting $\lambda_Q = 0$, which is the same as the degraded method in sentiment classification.
- Cross-Domain Emotion Tagging (CDET) method [51] predicts emotion distributions in the target domain by modifying the regularization term in a logistic regression model to control the training process.
- Transitive Transfer Learning (TTL) [41] simulates the process of human activities when learning the new domain knowledge, which utilizes part of labeled data in the target domain as the bridge to transfer the knowledge across domains based on NMTF.
- Cross-domain Cluster-level Emotion Pattern Matching (CDEPM) method [54] firstly performs K-means clustering algorithm on the source and target domains, respectively. Then, the document clusters are used to train a logistic regression model across domains.

The source codes of DERNN, TESAN, CSTM, cNMTF, and CDEPM are provided by the authors, and we reproduce other baselines of MTCNN, CDTC, TriTL, CDET, and TTL. All the aforementioned baselines and our methods adopt the same hyperparameter configuration by grid searching on the validation set. For each baseline, the hyperparameter values for grid searching are set according to the original description. In our methods, we adopt the same hyperparameter values for grid searching as in CDTC [56], and the remaining trade-off parameter is set by $\lambda_Q \in \{0.01, 0.05, 0.1, 0.5, 1.5, 10\}$.

With respect to the evaluation metric, we use two classical metrics (i.e., Accuracy and F1) to validate the effectiveness of our methods and baselines on sentiment classification. For emotion distribution learning, there are many metrics that can be applied to measure the similarity or distance between the predicted and actual emotion distributions. As suggested in [53] and [50], we employ six fine-grained metrics for distribution prediction evaluation, *i.e.*, Euclidean, Sϕrensen, Squared$\mathcal{X}^2$, Cosine, Fidelity, and Intersection. As the label distribution learning can be treated as a classification learning by simply sorting the label distribution, we also use two coarse-grained metrics to evaluate the classification performance: $Accu@1$ [51] and $NDCG@1$ [20].

4.3 Comparison with Baselines

Figure 3 and Table 5 show the averaged results of different methods over the 12 domain adaptation tasks on cross-domain sentiment classification and emotion distribution learning, respectively. To test whether the performance differences between our sNMTF-st and others are statistically significant, we first employ an effective measure of normality for small samples ($n < 20$), i.e., the Shapiro-Wilk test [38] on the 10 results of each model to determine the hypothesis testing method. The results indicate normal distributions for the 10 independent model performances, with



(a) Accuracy (↑) (b) F1 (↑)

Fig. 3. Performance comparison of all methods on sentiment classification in terms of Accuracy and F1 scores, where AVE, MAX and MIN denote the average, maximum, and minimum value of the results on 12 domain adaption tasks. For each metric, "↑" means the larger the value is, the better performance the model achieves.

Table 5. Performance comparison of all models on emotion distribution learning, where the best performance on each measure is highlighted by boldface. For the 12 domain adaptation tasks, the result is the average value with • indicating a significant difference according to the two-tailed t-tests with 5% significance level. Besides, "↓" after the metrics indicates smaller is better while "↑" indicates larger is better.

Category	Method	Fine-grained Metrics						Coarse-grained Metrics	
		Euclidean↓	Sørensen↓	Squaredχ^2 ↓	Intersection↑	Fidelity↑	Cosine↑	Accu@1↑	NDCG@1↑
Single Domain	MTCNN	0.4336•	0.4259•	0.5618•	0.5641•	0.7784•	0.7053•	0.4645•	0.6433•
	DERNN	0.4560•	0.4729•	0.6314•	0.5275•	0.7606•	0.6649•	0.3795•	0.5300•
	BERT-LR	0.4668•	0.4985•	0.6766•	0.5015•	0.7384•	0.6604•	0.4392•	0.6175•
	TESAN	0.4653•	0.4973•	0.6730•	0.5010•	0.7397•	0.6646•	0.4884•	0.6893•
Cross Domain	CDTC	0.7847•	0.7326•	1.2702•	0.2674•	0.4544•	0.3258•	0.1490•	0.2675•
	TriTL	0.5545•	0.5877•	0.8794•	0.4123•	0.6324•	0.4908•	0.1197•	0.2331•
	cNMTF	0.4200•	0.4287•	0.5520•	0.5714•	0.7880•	0.7193•	0.4835•	0.6610•
	TTL	0.5585•	0.5911•	0.9114•	0.4089•	0.6288•	0.4854•	0.1208•	0.2342•
	CDEPM	0.4736•	0.4566•	0.6193•	0.5434•	0.7745•	0.6895•	0.4925•	0.6693•
	CDET	0.4428•	0.4814•	0.6812•	0.5279•	0.7736•	0.6857•	0.4027•	0.5526•
	CSTM	0.4290•	0.4425•	0.5792•	0.5450•	0.7332•	0.6834•	0.4235•	0.5911•
Our Model	msNMTF	0.4197•	0.4388•	0.5606•	0.5618•	0.7977•	0.6871•	0.5156	0.7409•
	sNMTF-s	0.3746	**0.3808**	**0.5178**	**0.6068**	0.8267	**0.7504**	0.5211	**0.7681**
	sNMTF-t	**0.3740**	0.3842	0.5194	0.6048	**0.8338**	0.7466	**0.5216**	0.7669
	sNMTF-st	0.3777	0.3822	0.5194	0.6064	0.8305	0.7490	0.5206	0.7671

p-values ranging between 0.13 and 0.58. Then, the two-tailed t-tests with 5% significance level are performed, which validate the effectiveness of our models on both tasks. We can also observe that BERT-SVM and BERT-LR achieve undesirable performance, because it is hard for these two models to effectively learn the characteristics of different domains using a small amount of labeled samples in the target domain. To make in-depth discussions, we detail the results of each cross-domain sentiment classification task. The performance comparison of all sentiment classification models on each source-target domain pair is presented in Figures 4 and 5, in which, the source and target domains are listed in the vertical-axis and the horizontal-axis, respectively. For each cell, the value represents the performance of the corresponding source-target domain pair, and a darker color level indicates a better result for a certain method.

For cross-domain sentiment classification, our sNMTF obtains the best results on the minimum values of 12 domain adaption tasks and achieves remarkable performance on the average and maximum results. It is noteworthy that the DAS model extremely relies on the KL divergence of the extracted features between source and target domains, thus the minimum accuracy of DAS on those 12 domain adaption tasks is close to 0.5 (random prediction), which means that the above model is ineffective for two domains with large differences in domain characteristics. From Figures 4 and 5, we can also observe that Kitchen (K)↔Electronics (EL) is easier to be adapted across domains than other domain pairs. Furthermore, the DAML model is seriously affected by the random initialization

Fig. 4. Performance comparison of all models on 12 cross-domain sentiment classification tasks in Accuracy.

BERT-SVM	B	EL	D	K
B		0.63	0.64	0.65
EL	0.61		0.60	0.69
D	0.64	0.62		0.67
K	0.61	0.65	0.63	
			AVG:	0.64

DAS	B	EL	D	K
B		0.68	0.74	0.72
EL	0.54		0.58	0.78
D	0.72	0.68		0.72
K	0.51	0.78	0.53	
			AVG:	0.67

DAML	B	EL	D	K
B		0.67	0.71	0.70
EL	0.69		0.75	0.74
D	0.76	0.70		0.70
K	0.71	0.75	0.72	
			AVG:	0.72

sNMTF	B	EL	D	K
B		0.74	0.71	0.73
EL	0.71		0.70	0.74
D	0.71	0.73		0.74
K	0.71	0.74	0.69	
			AVG:	0.72

Fig. 5. Performance comparison of all models on 12 cross-domain sentiment classification tasks in F1.

BERT-SVM	B	EL	D	K
B		0.63	0.64	0.65
EL	0.61		0.60	0.69
D	0.64	0.62		0.67
K	0.61	0.65	0.63	
			AVG:	0.64

DAS	B	EL	D	K
B		0.68	0.75	0.72
EL	0.61		0.62	0.78
D	0.74	0.69		0.72
K	0.56	0.78	0.59	
			AVG:	0.67

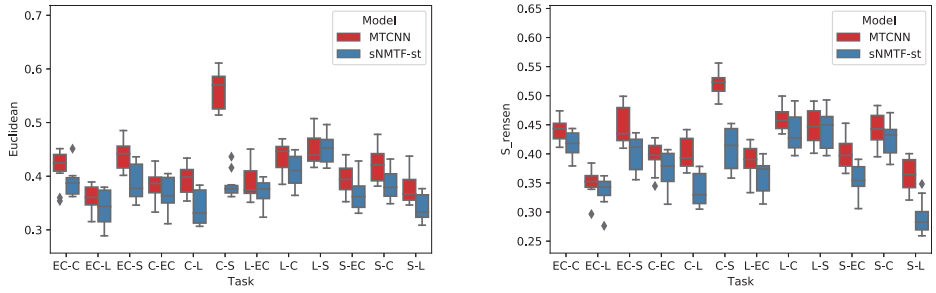
DAML	B	EL	D	K
B		0.61	0.71	0.67
EL	0.67		0.74	0.74
D	0.76	0.66		0.68
K	0.72	0.74	0.70	
			AVG:	0.70

sNMTF	B	EL	D	K
B		0.74	0.71	0.74
EL	0.72		0.71	0.74
D	0.72	0.73		0.74
K	0.71	0.74	0.69	
			AVG:	0.72

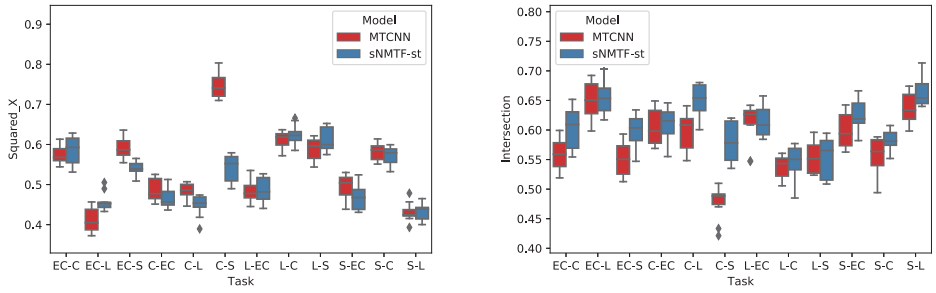
of parameters. By comparison, our sNMTF is convenient to train and performs quite stable on different domain adaption tasks.

For cross-domain emotion distribution learning, our models (i.e., sNMTF-s, sNMTF-t, and sNMTF-st) achieve nearly the same results and significantly outperform baseline models on both fine-grained and coarse-grained metrics. Among those baselines, it is interesting to observe that a neural network based method developed for single-domain emotion distribution learning (i.e., MTCNN) performs the best. The possible reason is that MTCNN uses the word-embedding pre-trained on Wikipedia through a multi-task framework, which incorporates additional information provided by the external corpus. Such information may reduce the influence of the difference of word-level distributions across domains. The performance comparison of MTCNN and our sNMTF-st is presented in Figure 6, in which, each cross-domain task is listed in the horizontal axis while the vertical axis presents the distribution of each metric on 10 independent random runs. When compared our sNMTF-st to the top-performing baseline of MTCNN, the results indicate that our method outperforms MTCNN consistently. It is worth noting that MTCNN performs quite limited on the Culture (C)→Society (S) task for fine-grained metrics. The possible reasons are two-fold. First, the emotion distribution learning tasks in which Society is chosen as the target domain are more difficult to learn than using other domains as the target domain, because the topics in Society may be quite varied. Second, the single-domain end-to-end model (i.e., MTCNN) relies on abundant training samples to achieve convincing performance. However, the scale of the Culture domain is limited. By modeling the intrinsic structure and the emotion distribution pattern across domains, our sNMTF-st can utilize the information of domain commonality to alleviate the above problems. Furthermore, the efficiency of sNMTF-st is much higher than that of MTCNN. Particularly, we implemented MTCNN by *Tensorflow* and ran it on *GeForce GTX 1080 Ti* GPUs. For each task, the averaged running time of MTCNN is 350 seconds. On the other hand, the averaged running time of our sNMTF-st is only 110 seconds on an *Intel I7-7700* CPU.

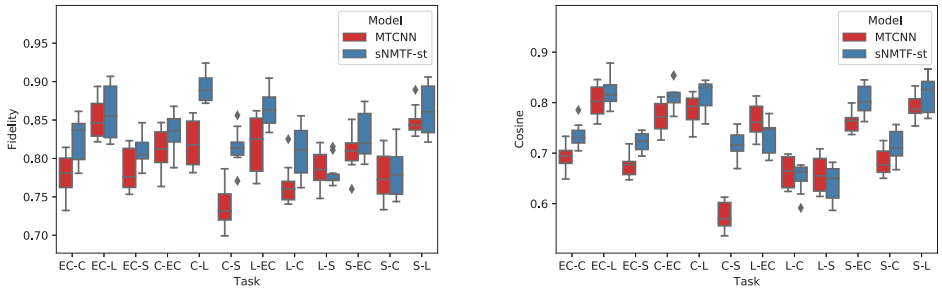
For different domain adaptation tasks, we observe that Economics (EC)→Culture (C) and Law (L)→Culture (C) are more challenging than Society (S)→Culture (C). The result is consistent to the diversity of feature distributions among these domains. Particularly, there is some overlap of words (or topics) between the domains of Society and Culture, while the domain-specific words (or topics) used in Economics and Laws are often much less used in Culture.



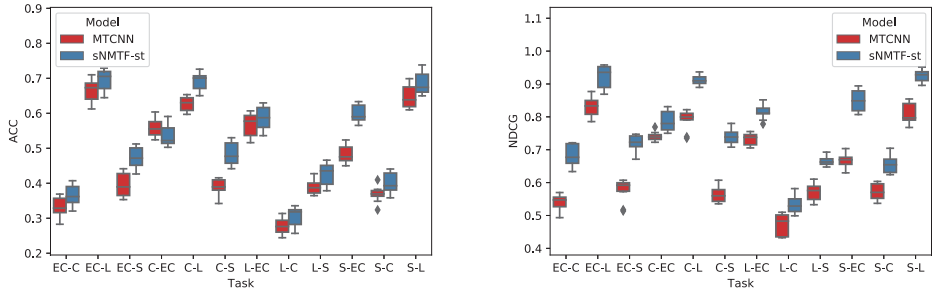
(a) Euclidean(↓) (b) Sϕrensen(↓)



(c) Squared$\mathcal{X}^2$(↓) (d) Intersection(↑)



(e) Fidelity(↑) (f) Cosine(↑)



(g) Accu@1(↑) (h) NDCG@1(↑)

Fig. 6. Performance comparison of MTCNN and our sNMTF-st on 12 cross-domain emotion distribution learning tasks. For each metric, "↓" means smaller is better and "↑" indicates larger is better.

4.4 Evaluation on Different Mapping Methods

For emotion distribution learning across domains, we observe that the baseline of CDTC performs poorly on all the metrics. The reason of obtaining quite limited performance for CDTC may be that it assumes a one-to-one mapping relationship between document clusters and labels, and such an assumption is inapplicable to emotion distribution learning. To further evaluate the effectiveness of constructing a many-to-many mapping between document clusters and emotions, we compare CDTC with our msNMTF (i.e., a degraded version which ignores the constraint of pairwise label dependency). During the training process, both models utilize the labeled data in the target domain for guidance. The only difference between them is that CDTC constructs a one-to-one mapping while msNMTF constructs a many-to-many mapping between document clusters and emotions. As we can see, our msNMTF significantly outperforms CDTC on all metrics. This validates that there exists a many-to-many mapping between document clusters and emotions. By incorporating the factor matrix $\boldsymbol{M}$, our many-to-many NMTF based model significantly improves the capability of the model to learn the inherent structure information among all factors.

4.5 Influence of the Label Dependency Matrix

To evaluate the effectiveness and robustness of generating the label dependency matrix $\boldsymbol{Q}$, we present the performance of the proposed many-to-many NMTF model without the label dependency constraint (i.e., msNMTF), and two models named sNMTF-s and sNMTF-t. For sNMTF-s, $\boldsymbol{Q}$ is generated based on $\boldsymbol{E_s}$ only, while for sNMTF-t, only the label distribution in $\boldsymbol{C_t E_t}$ are used to calculate $\boldsymbol{Q}$. As aforementioned, we denote our method which generates $\boldsymbol{Q}$ based on the union set of $\boldsymbol{E_s}$ and $\boldsymbol{C_t E_t}$ as sNMTF-st. Note that the sentiment polarity is opposite to each other for the Amazon dataset, thus we set $\boldsymbol{Q}$ as an identity matrix and the corresponding model is denoted as sNMTF. Compared to sNMTF for sentiment classification, as well as sNMTF-s, sNMTF-t, and sNMTF-st for emotion distribution learning, the performance of msNMTF slightly decreases on four coarse-grained metrics (i.e., Accuracy, F1, $Accu$@1, and $NDCG$@1), and significantly decreases on the other six fine-grained metrics. These results indicate that the label dependency constraint is valuable to the prediction of sentiment polarities and emotion distributions, especially for the latter task. As for sNMTF-s, sNMTF-t, and sNMTF-st, the performance is nearly the same, which indicates that the emotion dependency across domains is quite stable for the ChinaNews dataset.

For clarity, we further show the heatmaps of the prior sentiment polarity (or emotion) dependency $\boldsymbol{Q}$ and the learned sentiment polarity (or emotion) dependency $\boldsymbol{M}^T\boldsymbol{M}$ in Figures 7 and 8. The learned sentiment polarity dependency is from our sNMTF on the Books (B)→DVD (D) task, and the learned emotion emotion dependency is from our sNMTF-st on the Economics (EC)→Culture (C) task. Note that except for the diagonal elements, the sum value of each row in $\boldsymbol{M}^T\boldsymbol{M}$ is normalized to 1. These results show a good consistency between $\boldsymbol{Q}$ and $\boldsymbol{M}^T\boldsymbol{M}$, indicating that the prior information of label dependency $\boldsymbol{Q}$ refines the training of $\boldsymbol{M}$ successfully. Take the emotion "sympathy" as an example, its most related emotion is "sadness" in both $\boldsymbol{Q}$ and $\boldsymbol{M}^T\boldsymbol{M}$.

4.6 Influence of the Number of Document Clusters

In the previous experiments, we select the number of document clusters c and other parameter values in our models by grid searching. Considering that document clusters are directly associated with labels, we here evaluate the influence of c on our model performance. As an illustration, Figure 9 presents the accuracy and F1 scores of our sNMTF on sentiment classification when setting c in the range of {20, 50, 100, 150, 200, 300}. The results indicate that except for the extreme case of 20 document clusters, our sNMTF performs relatively stable under other settings.

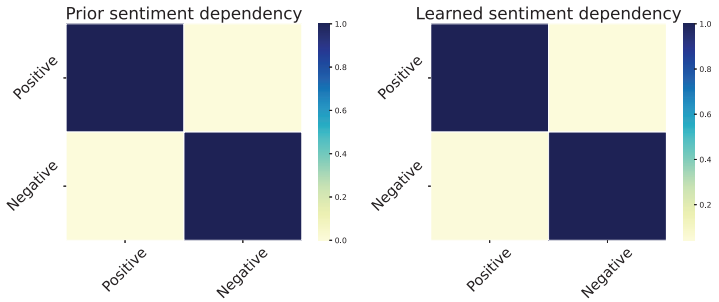


Fig. 7. Heatmaps of the prior sentiment polarity dependency Q (left) and the learned sentiment polarity dependency $M^T M$ from our sNMTF on the Books (B)→DVD (D) task (right).

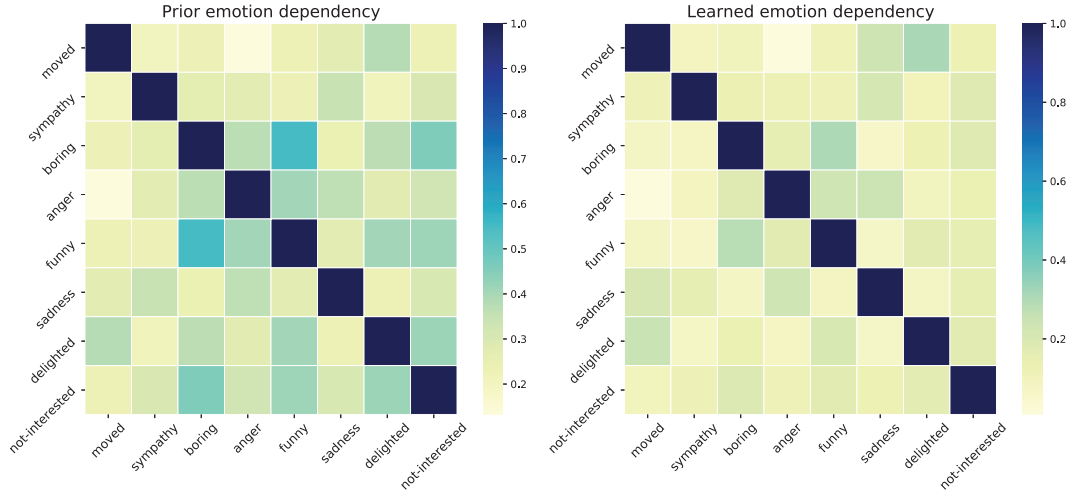


Fig. 8. Heatmaps of the prior emotion dependency Q (left) and the learned emotion dependency $M^T M$ from our sNMTF-st on the Economics (EC)→Culture (C) task (right).

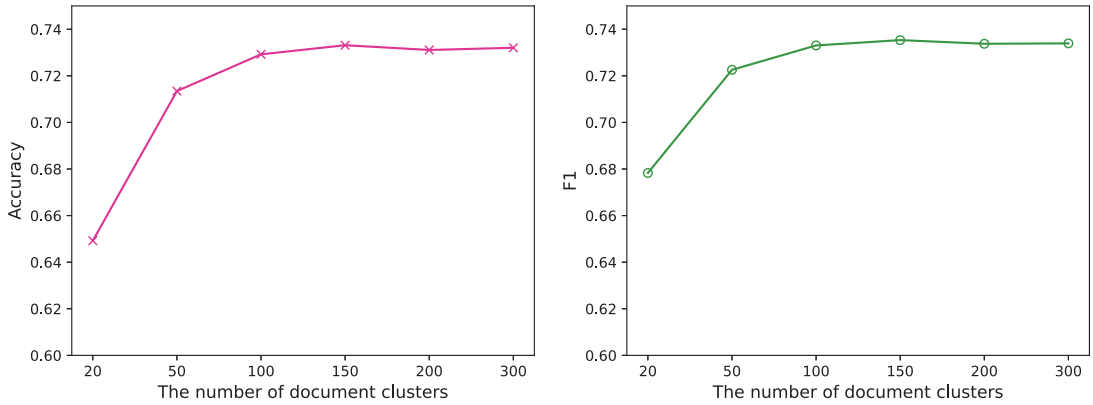


Fig. 9. Performance of our sNMTF on sentiment classification with different numbers of document clusters.

Table 6. The effect of unlabeled documents in the target domain.

#Unlabeled documents	0	2000	5000	8000	10000
Accuracy	0.69	0.70	0.68	0.69	0.71
F1	0.69	0.69	0.70	0.72	0.72

Table 7. The speedup and efficiency of our distributed algorithm.

#Processes	Time consuming per iteration (ms)	Speedup	Efficiency
1	7171	1.0000	1.0000
2	3740	1.9174	0.9587
4	2028	3.5360	0.8840
8	1190	6.0261	0.7533
16	725	9.8910	0.6182
20	656	10.9314	0.5466
24	610	11.7557	0.4898

4.7 Evaluation on the Distributed Algorithm

For the Amazon dataset, there exists an extra amount of unlabeled documents in every domain [2]. Our semi-supervised method could utilize unlabeled documents in the target domain for training. In Figures 4 and 5, we observe that our method achieves the lowest performance on the Kitchen (K)→DVD (D) domain pair among all tasks. Thus, we first test whether our model's performance can be enhanced by randomly integrating 2000, 5000, 8000, and 10000 unlabeled documents from the DVD domain into the training set. As shown in Table 6, our model achieves better performance generally with larger numbers of unlabeled documents. However, the high computation complexity of Algorithm 1 may hinder the model scalability. Considering this issue, we here evaluate our distributed algorithm based on the union of original training, validation, and testing sets, in addition to 10000 unlabeled documents.

We validate the efficiency of our distributed algorithm on the aforementioned dataset by applying two metrics, i.e., speedup and efficiency. Speedup is the ratio of time consumed by respectively running serial and distributed algorithms, which is formulated as $\text{Speedup}(p) = \frac{T_1}{T_p}$, where T_p is the running time on a machine with p processes and T_1 is the running time on a machine with only one process. Efficiency is the ratio of the actual speedup and the theoretically optimal speedup (i.e., the number of processes), which is formulated as $\text{Efficiency}(p) = \frac{\text{Speedup}(p)}{p}$. The parallel experiments are performed on "Tianhe-2", a supercomputer located in the National Super Computer Center at Guangzhou. Tianhe-2 consists of 17,920 compute nodes, where each node contains two 12-core Intel Xeon E5-2692 v2 processors and three 57-core Intel Xeon Phi 31S1P coprocessors. Table 7 shows the speedup and efficiency of our method with different numbers of processes. The results indicate that for the proposed distributed algorithm, the average consuming time per iteration decreases with the increasing of the number of processes.

5 CONCLUSIONS

In this paper, we propose a semi-supervised NMTF (sNMTF) framework for sentiment classification and emotion distribution learning across domains. By introducing a many-to-many mapping between document clusters and labels, our framework can capture the inherent structure information

among latent factors. The dependency of document labels (i.e., label dependency) is also used to improve the model performance. Furthermore, we enhance the scalability of our method by deploying the serial algorithm to a parallel one. We employ two real-world datasets on multiple domains for sentiment classification and emotion distribution learning, respectively. Extensive experiments validate the effectiveness of the proposed framework.

In the future, we would like to incorporate transitive transfer learning [41] into our method to deal with the scenarios where the source and target domains share few factors directly. We also plan to investigate our semi-supervised NMTF framework on category-name guided topic modeling [21] across domains, especially on the challenging task of lifelong learning [31]. In the context of topic discovery, NMTF or NMF not only enjoys sound interpretability of parts-based representation with sparseness, but also is able to flexibly introduce prior knowledge via regularization.

ACKNOWLEDGMENTS

The research described in this paper was supported by the National Natural Science Foundation of China (61972426). The work of Haoran Xie was supported by Lam Woo Research Fund (LWP20019), and the Faculty Research Grants (DB22B4 and DB22B7) of Lingnan University, Hong Kong. The work of Raymond Y. K. Lau was supported by a grant from the Research Grants Council of the HKSAR, China (Project: CityU 11507219), and a grant from the City University of Hong Kong SRG (Project: 7005780). The work of Jian Yin was supported by the National Natural Science Foundation of China under Grants U1811264, U1811262, U1811261, U1911203, U2001211, and U22B2060. We are grateful to Chang Wang for providing the source codes of their works [42] and [43].

REFERENCES

[1] S. Bao, S. Xu, L. Zhang, R. Yan, Z. Su, D. Han, and Y. Yu. 2012. Mining social emotions from affective text. *IEEE Transactions on Knowledge and Data Engineering* 24, 9 (2012), 1658–1670.

[2] J. Blitzer, M. Dredze, and F. Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. 440–447.

[3] J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. 120–128.

[4] L. A. M. Bostan and R. Klinger. 2019. Exploring fine-tuned embeddings that model intensifiers for emotion analysis. In *Proceedings of the 10th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@NAACL-HLT*. 25–34.

[5] Yufu Chen, Zhiqi Lei, Yanghui Rao, Haoran Xie, Fu Lee Wang, Jian Yin, and Qing Li. 2022. Parallel non-negative matrix tri-factorization for text data co-clustering. *IEEE Transactions on Knowledge and Data Engineering* (2022), 1–15. https://doi.org/10.1109/TKDE.2022.3145489

[6] S. Chhabra, P. Majumdar, M. Vatsa, and R. Singh. 2019. Data fine-tuning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. 8223–8230.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39, 1 (1977), 1–38.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.

[9] C. H. Q. Ding, T. Li, W. Peng, and H. Park. 2006. Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining*. 126–135.

[10] X. Glorot, A. Bordes, and Y. Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning*. 513–520.

[11] A. Go, R. Bhayani, and L. Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford* (2009).

[12] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier. 2018. Adaptive semi-supervised learning for cross-domain sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3467–3476.

[13] J. Howard and S. Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 328–339.

[14] M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining*. 168–177.
[15] R. Kannan, G. Ballard, and H. Park. 2018. MPI-FAUN: An MPI-based framework for alternating-updating nonnegative matrix factorization. *IEEE Trans. Knowl. Data Eng.* 30, 3 (2018), 544–558.
[16] P. Katz, M. Singleton, and R. Wicentowski. 2007. Swat-mp: The semeval-2007 systems for task 5 and task 14. In *Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval@ACL*. 308–313.
[17] S. Kiritchenko and S. M. Mohammad. 2016. The effect of negators, modals, and degree adverbs on sentiment composition. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@NAACL-HLT*. 43–52.
[18] S. Kiritchenko, X. Zhu, and S. M. Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research* 50 (2014), 723–762.
[19] Z. Li, Y. Wei, Y. Zhang, and Q. Yang. 2018. Hierarchical attention transfer network for cross-domain sentiment classification. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 5852–5859.
[20] K. H.-Y. Lin and H.-H. Chen. 2008. Ranking reader emotions using pairwise loss minimization and emotional distribution regression. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. 136–144.
[21] Yu Meng, Jiaxin Huang, Guangyuan Wang, Zihan Wang, Chao Zhang, Yu Zhang, and Jiawei Han. 2020. Discriminative topic mining via category-name guided text embedding. In *Proceedings of the Web Conference*. 2121–2132.
[22] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. 3111–3119.
[23] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning* 39, 2–3 (2000), 103–134.
[24] B. Ohana and B. Tierney. 2009. Sentiment classification of reviews using SentiWordNet. In *Proceedings of the 9th. IT & T Conference*. 13.
[25] S. J. Pan, X. Ni, J. T. Sun, Q. Yang, and Z. Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th International Conference on World Wide Web*. 751–760.
[26] B. Pang, L. Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2, 1–2 (2008), 1–135.
[27] B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. 79–86.
[28] M. Peng, Q. Zhang, Y.-G. Jiang, and X. Huang. 2018. Cross-domain sentiment classification with target domain specific information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 2505–2513.
[29] J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1532–1543.
[30] X. Qin, Y. Chen, Y. Rao, H. Xie, M. L. Wong, and F. L. Wang. 2021. A constrained optimization approach for cross-domain emotion distribution learning. *Knowledge-Based Systems* (2021), 107160.
[31] Xiaorui Qin, Yuyin Lu, Yufu Chen, and Yanghui Rao. 2021. Lifelong learning of topics and domain-specific word embeddings. In *Findings of the Association for Computational Linguistics*. 2294–2309.
[32] X. Qu, Z. Zou, Y. Cheng, Y. Yang, and P. Zhou. 2019. Adversarial category alignment network for cross-domain sentiment classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2496–2508.
[33] X. Quan, Q. Wang, Y. Zhang, L. Si, and W. Liu. 2015. Latent discriminative models for social emotion detection with emotional dependency. *ACM Transactions on Information Systems* 34, 1 (2015), 2:1–2:19.
[34] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-label corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. 248–256.
[35] Y. Rao. 2016. Contextual sentiment topic model for adaptive social emotion classification. *IEEE Intelligent Systems* 31, 1 (2016), 41–47.
[36] Y. Rao, Q. Li, X. Mao, and L. Wenyin. 2014. Sentiment topic models for social emotion mining. *Information Sciences* 266 (2014), 90–100.
[37] S. Rosenthal, P. Nakov, S. Kiritchenko, S. Mohammad, A. Ritter, and V. Stoyanov. 2015. SemEval-2015 task 10: Sentiment analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT*. 451–463.
[38] S. S. Shapiro and M. B. Wilk. 1965. An analysis of variance test for normality (complete samples). *Biometrika* 52, 3/4 (1965), 591–611.
[39] R. Sharma, P. Bhattacharyya, S. Dandapat, and H. S. Bhatt. 2018. Identifying transferable information across domains for cross-domain sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational*

Linguistics. 968–978.

[40] C. Strapparava and R. Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval@ACL*. 70–74.

[41] B. Tan, Y. Song, E. Zhong, and Q. Yang. 2015. Transitive transfer learning. In *Proceedings of the 21st International Conference on Knowledge Discovery and Data Mining*. 1155–1164.

[42] C. Wang and B. Wang. 2020. An end-to-end topic-enhanced self-attention network for social emotion classification. In *The Web Conference*. 2210–2219.

[43] C. Wang, B. Wang, W. Xiang, and M. Xu. 2019. Encoding syntactic dependency and topical information for social emotion classification. In *Proceedings of the 42nd International Conference on Research & Development in Information Retrieval*. 881–884.

[44] Y. Wang and A. Pal. 2015. Detecting emotions in social media: A constrained optimization approach. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. 996–1002.

[45] Y.-X. Wang and Y.-J. Zhang. 2013. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering* 25, 6 (2013), 1336–1353.

[46] T. Wilson, J. Wiebe, and P. Hoffmann. 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational Linguistics* 35, 3 (2009), 399–433.

[47] R. Xia, C. Zong, X. Hu, and E. Cambria. 2013. Feature ensemble plus sample selection: Domain adaptation for sentiment analysis. *IEEE Intelligent Systems* 28, 3 (2013), 10–18.

[48] Q. Xue, W. Zhang, and H. Zha. 2020. Improving domain-adapted sentiment classification by deep adversarial mutual learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. 9362–9369.

[49] K. Zhang, H. Zhang, Q. Liu, H. Zhao, H. Zhu, and E. Chen. 2019. Interactive attention transfer network for cross-domain sentiment classification. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. 5773–5780.

[50] Y. Zhang, J. Fu, D. She, Y. Zhang, S. Wang, and J. Yang. 2018. Text emotion distribution learning via multi-task convolutional neural network. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 4595–4601.

[51] Y. Zhang, N. Zhang, L. Si, Y. Lu, Q. Wang, and X. Yuan. 2014. Cross-domain and cross-category emotion tagging for comments of online news. In *Proceedings of the 37th International Conference on Research & Development in Information Retrieval*. 627–636.

[52] Z. Zhao and X. Ma. 2019. Text emotion distribution learning from small sample: A meta-learning approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 3955–3965.

[53] D. Zhou, X. Zhang, Y. Zhou, Q. Zhao, and X. Geng. 2016. Emotion distribution learning from texts. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 638–647.

[54] E. Zhu, Y. Rao, H. Xie, Y. Liu, J. Yin, and F. L. Wang. 2017. Cluster-level emotion pattern matching for cross-domain social emotion classification. In *Proceedings of the 2017 Conference on Information and Knowledge Management*. 2435–2438.

[55] F. Zhuang, P. Luo, C. Du, Q. He, and Z. Shi. 2013. Triplex transfer learning: Exploiting both shared and distinct concepts for text classification. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*. 425–434.

[56] F. Zhuang, P. Luo, H. Xiong, Q. He, Y. Xiong, and Z. Shi. 2010. Exploiting associations between word clusters and document classes for cross-domain text categorization. In *Proceedings of the SIAM International Conference on Data Mining*. 13–24.