Domain-Adversarial Graph Neural Networks for Text Classification

Man Wu [†], Shirui Pan [¶*], Xingquan Zhu [†], Chuan Zhou [‡§], Lei Pan [††]
[†]Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, USA
[¶]Faculty of Information Technology, Monash University, Melbourne, Australia
[‡]Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China
[§]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[††]School of Information Technology, Deakin University, Geelong, VIC, 3220, Australia
mwu2019@fau.edu, shirui.pan@monash.edu, xzhu3@fau.edu, zhouchuan@amss.ac.cn, l.pan@deakin.edu.au

***Abstract*—Text classification, in cross-domain setting, is a challenging task. On the one hand, data from other domains are often useful to improve the learning on the target domain; on the other hand, domain variance and hierarchical structure of documents from words, key phrases, sentences, paragraphs, *etc.* make it difficult to align domains for effective learning. To date, existing cross-domain text classification methods mainly strive to minimize feature distribution differences between domains, and they typically suffer from three major limitations — (1) difficult to capture semantics in non-consecutive phrases and long-distance word dependency because of treating texts as word sequences, (2) neglect of hierarchical coarse-grained structures of document for feature learning, and (3) narrow focus of the domains at instance levels, without using domains as supervisions to improve text classification. This paper proposes an end-to-end, domain-adversarial graph neural networks (DAGNN), for cross-domain text classification. Our motivation is to model documents as graphs and use a domain-adversarial training principle to lean features from each graph (as well as learning the separation of domains) for effective text classification. At the instance level, DAGNN uses a graph to model each document, so that it can capture non-consecutive and long-distance semantics. At the feature level, DAGNN uses graphs from different domains to jointly train hierarchical graph neural networks in order to learn good features. At the learning level, DAGNN proposes a domain-adversarial principle such that the learned features not only optimally classify documents but also separates domains. Experiments on benchmark datasets demonstrate the effectiveness of our method in cross-domain classification tasks.**

***Keywords*-Graph neural networks, cross-domain learning, text classification**

I. Introduction

Text classification, an important and classical challenge in natural language processing, has raised continuous attention over the past decades. Many algorithms have been developed to automatically organize texts using the classifiers well trained from a large number of training samples. In practice, it may be time-consuming and expensive to collect sufficiently labeled data in a specific domain of interest, whereas a large number of labeled data from a related but different domain might exist to support the learning [1]. It has motivated many cross-domain text classification research efforts in recent years, which aims to borrow knowledge from domains with abundant labeled data

*Corresponding author.

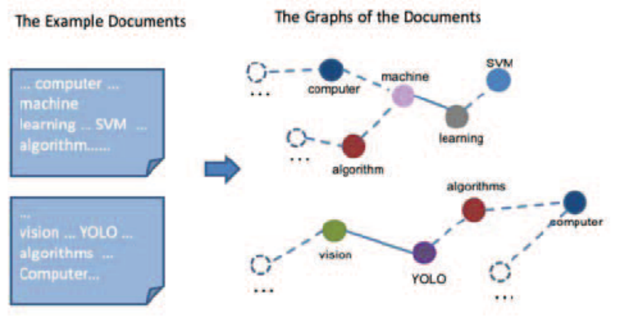

The Example Documents
The Graphs of the Documents
... computer ... machine learning ... SVM ... algorithm......
... vision ... YOLO ... algorithms ... Computer...
computer
machine
SVM
learning
algorithm
algorithms
computer
vision
YOLO

Fig. 1: Examples of converting documents to graphs. Nodes represent words, and edges denote co-occurrences between words within the same context of the document. Solid lines indicate direct relationships between two words, and dashed lines represent long range semantic relations.

to train a classifier for correct classification of documents from a target domain [2].

To date, the existing cross-domain text classification methods can be grouped into four categories — instance reweighting based approaches [3], co-training methods [4], kernel methods [5], and feature-representation based methods [6], [7], among which the feature representation based approaches are the most common ones. The key idea of feature-representation based methods is to find a latent feature space to contract the distributions of different domains. For example, Spectral Feature Alignment (SFA) [8] aims to model the relationship between word features to bridge domains for cross-domain sentiment classification; Topic Correlation Analysis (TCA) [1] extracts both shared and domain-specific features to facilitate the text classification knowledge transfer between domains. For all these traditional methods, they represent text using hand-craft features, which need to manually extract and identify the latent feature representations across domains.

Recently, deep neural networks, as a great tool to automatically learn feature representations, have been used for cross-domain learning. For example, marginalized SDA (mSDA) [9] uses auto-encoders to learn latent high-level feature representations for cross domain sentiment analysis. Meanwhile, a

2374-8486/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDM.2019.00075

$l_{1,2}$-norm stacked robust autoencoders ($l_{1,2}$-SRA) for domain adaptation [10] uses learned features to train a classifier, using samples from the source domain, and applies the trained classifier directly to the target domain. Furthermore, Recurrent Neural Networks (RNNs) [11] and convolutional neural networks (CNNs) [12] in particular, have been widely used for text representation and classification, thanks to their impressive ability to capture local semantic features in small sliding windows (short messages or word-level syntactics or semantics) and flexibility to deal with sequential information. However, the above methods cannot capture the long range semantic relations among words in feature representation, because they ignore the structural information of documents which is a very important aspect for cross-domain text classification.

In order to capture structural dependency between words, there are several proposals to model documents as graphs [13], where each document corresponds to a graph with nodes representing words and edges denoting the dependency between words, as shown in Fig. 1. Although graph represented documents can accurately capture word dependency, obtaining features for graph classification is a challenging task, and common approaches such as gSpan [14] often result in a large number of subgraph features while representing documents for classification.

Instead of relying on subgraph features, the recently developed graph neural networks (GCNs) [15] have shown to be a powerful tool to leverage the content and structure of graph objects and represent them as feature vectors for classification. The superb feature learning capability of GCNs provides an alternative solution to capture the long-distance dependency between word objects for document classification. Accordingly, a recent paper [16] uses information about the predicate-argument structure of source sentences and uses GCNs to inject a semantic bias into sentence encoders for machine translation. TextGCN [17] builds a single text graph for a corpus based on word co-occurrence and document-word relations, and it subsequently learns a text graph convolutional network for the corpus to obtain word and document embeddings. Although GNNs have been successfully used for structure data to learn feature representations, to the best of our knowledge, no existing work has explored using GNNs to handle texts originated from multiple domains.

Advancing graph neural networks for cross-domain classification is, however, a nontrivial task, and has the following three major challenges:

Instance level challenge: Existing RNNs or CNNs based text classification use sequence of words to represent a document, they only capture local semantic features in small sliding windows (short messages or word-level syntactics or semantics). We need to find a new way to represent document and preserve non-consecutive and long-distance dependency between words for classification.

Feature level challenge: Because each document has hierarchical structures from words, key phrases, paragraphs, *etc*., we need to learn the hierarchical features to accurately reflect such structures. Existing GNN methods only focus on word-level relationships, they can not obtain the hierarchical features, which may result in the degraded performance when source and target domains only have few overlapping content. Capturing hierarchical information is important for cross domain text classification, because documents from different domains often share coarse-grained structures.

Cross-domain learning challenge: At the learning level, we need to leverage information from different domains to jointly learn a classifier for the target domain. While traditional cross-domain learning methods mainly focus on reducing the feature distribution discrepancy across different domains, they essentially ignore to consider domain a valuable source of supervision. Recently, an adversarial adaptation method [18] was proposed to train domain classifier to discern whether a sample is from the source domain or the target domain. We propose to advance this adversarial learning paradigm to a domain-adversarial learning principle, where cross-domain learning aims to find features not only optimally classify documents into classes, but also differentiate domains (or minimize the domain confusion loss).

To address the above limitations, we propose an end-to-end Domain-Adversarial Graph Neural Networks (DAGNN) for cross-domain text classification by modeling documents as graphs data structure, and combining the hierarchical structure and domain labels into a unified deep model. (1) At the instance level, in order to accurately capture word relations, we use a graph to represent each document. A novel graph neural network is utilized to effectively deal with rich relational structure and catch long-distance semantics in texts; (2) At the feature level, we propose a novel hierarchical graph pooling layer to extract the high-level feature representations of input graph; (3) At the learning level, the domain classification loss is utilized to guide the feature extractor to learn domain-shared representations via the Gradient Reversal Layer for effective cross-domain text classification. Empirical results on two public real datasets demonstrate that DAGNN outperforms the state-of-the-art cross-domain text classification methods. Our contributions are summarized below:

- We propose an end-to-end Domain-Adversarial Graph Neural Network (DAGNN) for cross-domain text classification by jointly modeling word relations, hierarchical graph structure, and domain labels as a unified learning framework. To the best of our knowledge, this is the first work to model the three kinds of information jointly in a deep model for cross-domain text classification.
- We model documents as graphs in classification tasks, which can capture non-consecutive and long-distance semantics. Moreover, a well-designed hierarchical graph pooling network is proposed to capture the hierarchical high-level document representations of the source and target domains.
- We evaluate our method on real-world datasets, and the results demonstrate that the proposed model outperforms the baseline methods.

II. Related work

A. Cross-domain classification

Existing cross-domain classification algorithms can be roughly categorized into four groups: (1) Instance re-weighting approaches aim to identify the training samples in the source domain that are most relevant to the target domain by instance re-weighting and importance sampling. Then the re-weighted source instances are used for training a target domain model [3]. (2) Co-training methods bridge the gap between the source domain and the target domain by slowly adding target features and the most reliable examples of the current algorithm in the training set [4]. (3) Kernel methods explore multiple kernels to induce an optimal learning space and learn a kernel function and a robust classifier by minimizing the distribution mismatch between the labeled and unlabeled samples from the source and target domains [5]. (4) Feature representation based methods are designed to map different domains into a common shared space and contract their feature distributions as close as possible [6], [19].

Recently, deep learning has been regarded as a powerful way of learning feature representations for cross-domain classification. Stacked Denoising Auto-encoder (SDA) [20] is proposed to learn intermediate high-level feature representations for cross-domain sentiment analysis.

For cross-domain learning, many methods use an adversarial objective to reduce domain discrepancy [21], [22]. Among which, the domain adversarial neural network (DANN) [18] learns domain invariant features by a minimax game between the domain classifier and the feature extractor, using a gradient reversal layer to back-propagate the gradients computed from the domain classifier. Our proposed DAGNN model also adopts the adversarial adaptation. However, the difference is that our model can model word relations, hierarchical graph structure, and domain labels as a unified learning framework.

B. Graph Neural Networks (GNN)

Graph Neural Networks are designed to use deep learning architectures on graph-structured data. Many solutions are proposed to generalize well-established neural network models (such as CNN) that work on regular grid structure to deal with graphs with arbitrary structures [23]–[27]. A recent pioneer work [15] proposes a simplified graph convolutional networks (GCNs), which has been applied to many NLP tasks to learn text representations. A GCNs based framework [28] incorporates the syntactic structure into neural attention-based encoder-decoder models for machine translation.

In order to capture word correlations, TextGCN [17] built a single text graph for a corpus based on word co-occurrence and document word relations, then learned a Text Graph Convolutional Network for the corpus to learn word and document embeddings. Peng et al. [29] proposed a GCN-based deep learning model to first convert texts into graph-of-words, and then used graph convolution operations to convolve the word graph for text classification. In order to capture structural dependency between words, we model documents as graphs, which are are ubiquitous in the real world and widely used to model the inter-dependence among data in many applications [30]. Then, a novel Graph neural networks (GNNs) [15] is utilized to effectively deal with rich relational structure and catch long-distance semantics. Furthermore, a novel hierarchical graph pooling network is proposed to extract hierarchical high-level feature representations of input data, which can help capture coarse-grained structures of words of each document for complex scenes.

III. Problem Definition and Overall Framework

This section defines the problem to be addressed and then presents our overall framework for the problem.

A. Problem Statement

Text Classification in Target Domains In this paper, we focus on text classification. Given a set of documents from a target domain $\mathbb{D}_t = \{\mathcal{D}_t^1, ..., \mathcal{D}_t^{N^t}\}$, where $\mathcal{D}_t^i$ is a document consisting of a set of words. We need to learn a model $f_t : \mathbb{D}_t \rightarrow \mathcal{Y}$, to classify each document $\mathcal{D}_t^i$ into a number of predefined categories $\mathcal{Y} = \{C_1, \cdots, C_{|C|}\}$. Unlike traditional text classification, the class label for each document $\mathcal{D}_t^i$ in $\mathbb{D}_t$ is not available, therefore, we could not directly build a model f_t for text classification in this domain.

Source Domain Knowledge We are also given documents in the source domain $\mathbb{D}_s = \{(\mathcal{D}_s^1, y_s^1), ..., (\mathcal{D}_s^{N^s}, y_s^{N^s})\}$, where each document $\mathcal{D}_s^i$ is associated with a class label $y_s^i \in \mathcal{Y}$. As the class labels information is available in $\mathbb{D}_s$, we can easily build a supervised model $f_s : \mathbb{D}_s \rightarrow \mathcal{Y}$.

Cross-Domain Text Classification. Given a target domain $\mathbb{D}_t = \{\mathcal{D}_t^1, ..., \mathcal{D}_t^{N^t}\}$ and a source domain $\mathbb{D}_s = \{(\mathcal{D}_s^1, y_s^1), ..., (\mathcal{D}_s^{N^s}, y_s^{N^s})\}$, the cross-domain text classification aims to build a classifier $f : (\mathbb{D}_s, \mathbb{D}_t) \rightarrow \mathcal{Y}$ to predict the class labels of unlabeled examples in the target domain. However, this is a very challenging task due to the lack of labels for $\mathbb{D}_t$. In this paper, we advocate a domain-adversarial learning framework for this task, with the objective function being formulated as follows:

$$\min \mathcal{L}_C(f_s(Z_s), Y_s) + \lambda \mathcal{L}_{DC}(Z_s, Z_t) \tag{1}$$

where $\mathcal{L}_C(f_s(Z_s), Y_s) = \sum_{i=1}^{N_s} Loss(f_s(z_s^i), y_s^i)$ is the classification loss in the source domain with the classifier model f_s, and $z_s^i \in Z_s$ is the feature representation of document $\mathcal{D}_s^i$. $\mathcal{L}_{DC}(Z_s, Z_t)$ measures the similarity between the feature representation Z_s from documents from $\mathbb{D}_s$ and Z_t from $\mathbb{D}_t$. λ is the balance parameter during the learning process. Basically, our models will target three goals simultaneously: 1) learn a good feature representation z^i for each document $\mathcal{D}^i$, 2) build a robust (deep) model from the source domain by minimizing the classification loss $\mathcal{L}_C(Z_s, Y_s)$, and 3) align the representation from the source domain and the target domain to a similar space. By this way, the model f_s built from $\mathbb{D}_s$ can be used to predict the documents in the target domain.

Notations We develop a graph neural network based algorithm to enable the domain-adversarial training framework.

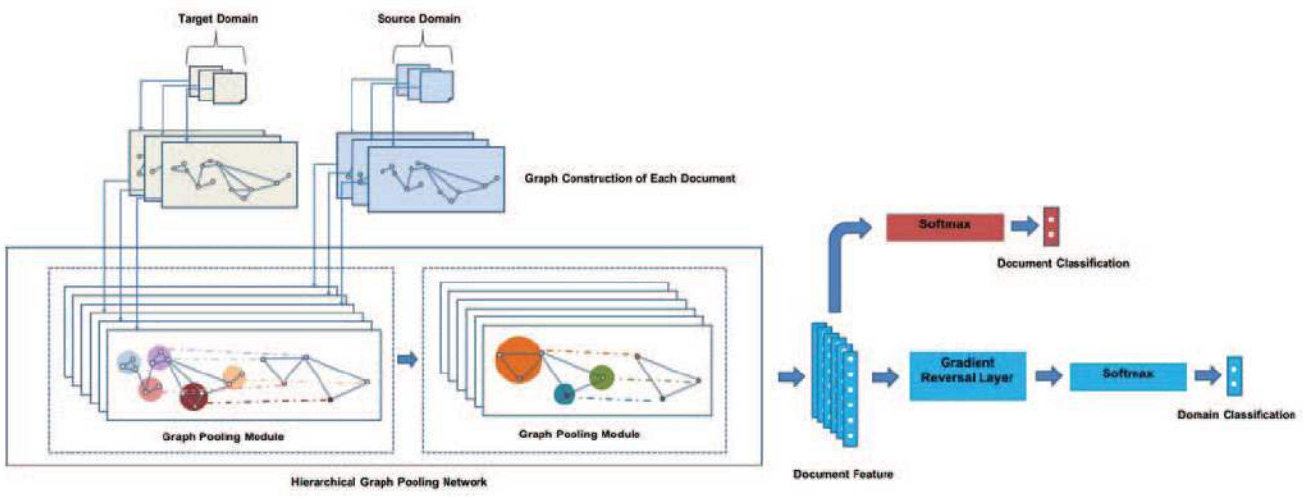


Fig. 2: The overall architecture of the proposed Domain-Adversarial Graph Neural Network (DAGNN) model for cross-domain text classification. The input consists of documents from source and target domains. (1) DAGNN first converts each document as a graph; (2) Then, a hierarchical graph pooling network is used to capture the hierarchical document-level representation based on training graphs; (3) Finally, after extracting document representations from the hierarchical Graph Pooling Network, we introduce two loss functions for training the label predictor of the source domain and the domain classifier of the source domain and the target domain, respectively. Please refer to the content in Section IV for detailed descriptions.

Formally, given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. We use $|V|$ to denote the number of vertices. Each vertex is associated with a feature vector which has the dimension of $d^{(0)}$. A feature matrix $X \in \mathbb{R}^{|V| \times d^{(0)}}$ is used to represent the features of all vertices, where the i-th row corresponds to the feature vector of the i-th vertex. The edge set E is commonly indicated by an adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$, in which A_{ij} is the weight of the edge between the i-th and the j-th vertex. The degree matrix $D \in \mathbb{R}^{|V| \times |V|}$ is an important concept in GCN models, where D is a diagonal matrix and $D_{ij} = \sum_j A_{ij}$.

B. Overall Framework

In order to leverage cross-domain documents to learn a classifier for text classification, we propose a domain-adversarial graph neural networks (DAGNN) to reduce the distribution gap and induce a low-dimensional feature representation shared across domains. Our framework, as shown in Figure 2, mainly consists of the following three components:

- **Graph Representation for Documents**. We will first represent each document as a graph, so that the non-consecutive and long distance dependency between words can be captured.
- **Hierarchical Graph Pooling Network**. Based on the graph representation, a hierarchical graph pooling network is developed to obtain hierarchical feature representations: (1) The network employs a special GCN model which combines graphs from source and target domains to train parameters-shared GCN model to learn representations; (2) With the GCN model, each graph pooling module in the network can learn latent document-based cluster representations at a document level; and (3) multiple modules are stacked to build a hierarchical document representation model.
- **Domain-Adversarial Learning for Cross-Domain Classification**. To enable cross-domain classification, we advocate a domain-adversarial learning to train two classifiers. The first one aims to minimize the classification loss on the source domain data, and the other one enforces the differentiation between the source and target domains. By doing so, the domain-adversarial learning can maximally utilize the domain information to train classifiers for cross-domain classification.

IV. Methodology

This section presents our domain adversarial graph neural network approach for text classification.

A. Graph Representation for Documents

Our model creates an undirected graph for each document to model its content information. Given a document, the content words are taken as the graph vertices. We employ the point-wise mutual information (PMI) to calculate the weights of edges, which preserves the global word co-occurrence information [17]. In detail, we employ a fixed-size sliding window on all documents in the source and target domains to collect word co-occurrence statistics. Note that statistics are based on global corpus rather than a specific document.

We calculate the PMI of word pairs as follows:

$$p(w_i) = \frac{W(w_i)}{|W|}, \tag{2}$$

$$p(w_i, w_j) = \frac{W(w_i, w_j)}{|W|}, \tag{3}$$

$$\mathsf{PMI}(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}, \quad (4)$$

where $W(w_i)$ is the number of sliding windows that contain the word w_i, $W(w_i, w_j)$ is the number of sliding windows that contain both words w_i and w_j, and $|W|$ is the total number of sliding windows. The PMI score can reflect the correlation between words, and a higher PMI score reflects stronger semantic correlation. Therefore, we only preserve the edges with positive PMI scores, while exclude the edges with non-positive PMI scores:

$$a_{ij} = \begin{cases} \mathsf{PMI}(w_i, w_j) & \mathsf{PMI}(w_i, w_j) > 0 \\ 0 & \mathsf{PMI}(w_i, w_j) \leq 0, \end{cases} \quad (5)$$

where a_{ij} is the relation between the word w_i and w_j. After this process, we obtain the word relations $\mathcal{A}$ over the global corpus, and the adjacent matrix A is the subset of $\mathcal{A}$ for each document $\mathcal{D}^i$.

To produce the feature matrix X for each document, we use the pre-trained fastText word embedding vectors [31], which contains more than 2 million pre-trained word vectors. Compared to other pre-trained word embedding vectors such as GloVe [32], using the fastText helps us to avoid massive unknown words.

B. Hierarchical Graph Pooling Network

With graph representation, we are able to capture more complex semantic and long distance relation between words. However, the challenges for extracting graph features are also increased. In this paper, we develop an effective hierarchical graph pooling network for graph representation learning. Our framework consists of two key components, 1) *node level graph embedding* which aims to embed each node into a single vector, and 2) *graph level embedding* which summarizes all embedding in a graph into a fixed length vector, so that graph level classification can be performed. All the embedding is done in an end-to-end fashion with graph neural networks.

1) Node Embedding with Graph Convolutional Network: GCN is an effective approach to embed the node in a graph into a vector. A GCN [15] is a multilayer neural model that takes an undirected graph as input and outputs embedding vectors for vertices based on vertex features and relations.

Based on the adjacency matrix and the degree matrix, each GCN layer transforms the input feature matrix $Z^{(j)} \in \mathbb{R}^{|V| \times l^{(j)}}$ (the raw feature matrix or the output of the previous GCN layer) and generates a higher-order feature matrix $Z^{(j+1)} \in \mathbb{R}^{|V| \times l^{(j+1)}}$ for vertices as follows:

$$Z^{(0)} = X, \quad (6)$$

$$Z^{(j+1)} = \sigma(D^{-\frac{1}{2}}(I + A)D^{-\frac{1}{2}}Z^{(j)}\theta), \quad (7)$$

where $\theta \in \mathbb{R}^{l^{(j)} \times l^{(j+1)}}$ is a transformation matrix and σ is an activation function which adds nonlinearity to the GCN layer.

One problem of the GCN model [15] is that it only considers the vertices that are a few propagation steps away and the size of this utilized neighborhood is hard to extend, since too many layers lead to oversmoothing. Motivated by [33], we employ an approximate personalized propagation based algorithm to address the issue. Our GCN model can achieve linear computational complexity by approximating topic-sensitive PageRank [34] via power iteration, as shown in Fig. 3 (a).

GCN Update Formula Our GCN module, defined as $\mathsf{GCN}(A, X)$, takes in a node feature matrix X and and adjacency matrix A, and then generates a new representation Z for all nodes. This can be formulated as follows :

$$Z^{(0)} = H = f_{\mathsf{MLP}}(X), \quad (8)$$

$$Z^{(j+1)} = (1-\alpha)D^{-\frac{1}{2}}(I + A)D^{-\frac{1}{2}}Z^{(j)} + \alpha H, \quad (9)$$

where f_{MLP} denotes a multilayer perceptron and α is the restart probability. By introducing the restart probability α, our GCN component can have many layers and leverage the information from far more propagation steps without leading to oversmoothing.

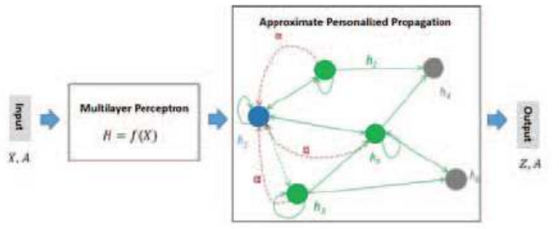
2) Graph Level Embedding via Pooling: As each graph has a different number of nodes with variable node embedding, we need to summarize them into a single vector to facilitate graph level classification. In our framework, we advance a differentiate pooling approach [35], which hierarchically clusters the nodes in a graph into a set of clusters. As the original graph is coarsened more and more during iterations, we finally get a single cluster, then we can learn a graph level embedding with a single vector. In this way, our model is able to capture the hierarchical information of any input graph after training.

The core of the our graph pooling (GP) module is to learn an assignment matrix to assign the words in each document to a set of clusters. We first describe how the module pools words in each document when giving an assignment matrix, then discuss how we learn the assignment matrix.

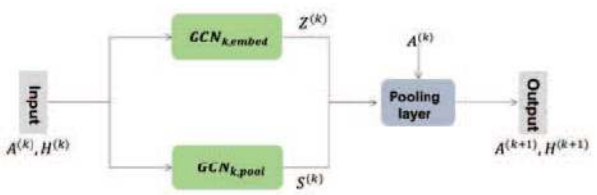
Here we use the superscript k ($k \geq 1$) to denote the module index. At the k-th GP module, we denote the input embedding matrix as $H^{(k-1)}$, the node embedding matrix as $Z^{(k-1)}$, the cluster assignment matrix as $S^{(k-1)}$ and the adjacency matrix as $A^{(k-1)}$. The first module takes the aforementioned constructed graph $(H^{(0)}, A^{(0)})$ as input, where the vertex feature matrix $H^{(0)} \in \mathbb{R}^{|V| \times d}$ uses the corresponding word embeddings as features. The word embeddings are d-dimensional vectors, which will be optimized in the training step. Note that when $k > 1$, the input graph of the k_{th} module is dynamically generated by the $(k-1)_{th}$ module, which will be discussed later.

Update the embedding and adjacency matrix. Assuming that the assignment matrix $S^{(k)} \in R^{v^k \times d}$ and the node embedding matrix $Z^{(k)} \in R^{v^k \times v^{k+1}}$ of the k-th GP module have been computed, the GP module $(H^{(k+1)}, A^{(k+1)}) = \mathsf{GP}(Z^{(k)}, A^{(k)})$ will generate a new coarsened adjacency matrix $A^{(k+1)}$ and a new input embedding matrix $H^{(k+1)}$, which can form a coarsened graph for next GP module. The $H^{(k+1)}$ and $A^{(k+1)}$ are calculated as follows:

$$H^{(k+1)} = {S^{(k)}}^T Z^{(k)} \in R^{v^{k+1} \times d}, \quad (10)$$



(a) The architecture of graph convolutional network with approximate Personalized propagation.



(b) Graph pooling module

Fig. 3: The illustration of the proposed graph convolution network and graph pooling module. (a) Given the input (X denotes the feature matrix and A is the adjacent matrix), the graph convolution network first generated from each node's own features (X denotes the feature matrix) by a multilayer perceptron and then propagated using an adaptation of personalized pagerank. The iteration process follows Eq.(8) and Eq.(9), in which α is the restart probability to give the GCN the ability to have many layers and leverage the information from far more propagation steps without leading to oversmoothing. (b) Graph pooling module is combined with two proposed graph convolution network. The input to the k-th module are the adjacency matrix $A^{(k)}$ and input embedding matrix $H^{(k)}$. We first run the $\mathsf{GCN}_{k,embed}$ to learn the node embedding $Z^{(k)}$ and the $\mathsf{GCN}_{k,pool}$ to output the cluster assignment matrix $S^{(k)}$. Then we generate the new adjacency matrix $A^{(k+1)}$ and input embedding matrix $H^{(k+1)}$ according to $S^{(k)}$.

$$A^{(k+1)} = {S^{(k)}}^T A^{(k)} S^{(k)} \in R^{v^{k+1} \times v^{k+1}}, \tag{11}$$

where d is the output feature dimension and v^k is the number of nodes (words) in the k-th GP module. The input embedding matrix $H^{(k+1)}$ at the $(k+1)$-th GP module is computed by aggregating the node embedding matrix $Z^{(k)}$ according to the assignment matrix $S^{(k)}$ at the k-th module. $A^{(k+1)}$ is calculated in a similar way. Through the GP module, the graph is coarsened. The adjacency matrix $A^{(k+1)}$ represents a coarsened graph with v^{k+1} cluster nodes, where each individual cluster node in the new coarsened graph corresponds to a cluster of nodes in the graph at the layer k.

Learn the assignment matrix. We learn the assignment matrix $S^{(k)}$ and node embedding matrix $Z^{(k)}$ via our graph neural networks (Eqs. 8 and 9). Specifically, we generate these two matrices by applying the matrix $A^{(k)}$ and $H^{(k)}$ to separate GCN modules as follows:

$$Z^{(k)} = \mathsf{GCN}_{k,embed}(A^{(k)}, H^{(k)}), \tag{12}$$

$$S^{(k)} = \mathsf{softmax}(\mathsf{GCN}_{k,pool}(A^{(k)}, H^{(k)})), \tag{13}$$

where the softmax operation is applied in a row-wise fashion. Although these two GCN modules use the same input, their parameters are different and they have different meanings. The embedding GCN module generates the node embeddings and the pooling GCN module generates a probabilistic assignment to v^{k+1} clusters at the layer k, and we use $\mathcal{C}^{(k)}$ to denote the number of semantic clusters.

The visualization of the GP module is illustrated in Fig. 3(b). Our trainable GP module can extract the complex hierarchical structures of a graph by mapping nodes (words) of each document to a set of clusters according to the assignment matrix. Our model stacks K graph pooling modules together to form a hierarchical domain shared feature extractor that contribute to text classification.

C. Domain-Adversarial Learning for Cross-Domain Classification

We propose a domain-adversarial graph neural network (DAGNN) to learn the domain shared feature that contribute to text classification. As shown in Eq.(1), our loss function consists of two parts: the classification loss in the source domain $\mathcal{L}_C(f_s(Z_s), Y_s)$ and the domain classification loss $\mathcal{L}_{DC}(Z_s, Z_t)$.

Document Classification The classification loss $\mathcal{L}_C(f_s(Z_s), Y_s)$ is to minimize the cross-entropy for the labeled data $\mathcal{D}_s$ in the source domain:

$$\mathcal{L}_C(f_s(Z_s), Y_s) = -\frac{1}{N_s}\sum_{i=1}^{N_s} y_i \mathsf{log}(\hat{y}_i), \tag{14}$$

where y_i denotes the label of the i-th document in the souce domain, $\hat{y}_i$ are the groundtruth and classification prediction for the i-th source labeled sample $\mathcal{D}_s^i$, respectively.

Domain Classification The domain classification loss $\mathcal{L}_{DC}(Z_s, Z_t)$ enforces that the document feature representation after hierarchical graph pooling networks from source domain $\mathbb{D}_s$ and target domain $\mathbb{D}_t$ are similar. To achieve this, we learn a domain classifier $f(Q_\lambda(Z_s, Z_t); \theta_D)$ parameterized by θ_D with an adversarial training scheme, which tries to disciminative if a document is from $\mathbb{D}_t$ or $\mathbb{D}_s$. On the one hand, we would like our document classifier f_s can classify each document into the correct class via minimizing Eq. (14). On the other hand, we would like that features from different domains are similar, so that the domain classifier cannot differentiate if the document comes from $\mathbb{D}_t$ or $\mathbb{D}_s$.

In our paper, we use Gradient Reversal Layer (GRL) [18] for adversarial training. Mathematically, we define the GRL as $Q_\lambda(x) = x$ with a reversal gradient $\frac{\partial Q_\lambda(x)}{\partial x} = -\lambda I$. Learning a GRL is adversarial in such a way that: on the one side, the reversal gradient enforces $f_s(Z_s)$ to be maximized; on the other side, θ_D is optimized by minimizing the cross-entropy domain classification loss:

$$\mathcal{L}_{DC} = -\frac{1}{N_s + N_t} \sum_{i=1}^{N_s+N_t} m_i \log(\hat{m_i}) + (1-m_i)\log(1-\hat{m_i}) \quad (15)$$

where $m_i \in \{0, 1\}$ denotes the groundtruth, and $\hat{m_i}$ denotes the domain prediction for the i-th document in the source domain and target domain, respectively.

$\mathcal{L}_C(f_s(Z_s), Y_s)$ and $\mathcal{L}_{DC}$ are jointly optimized via our objective function in Eq. (1), and all parameters are optimized using the standard backpropagation algorithms.

V. Experiments

A. Benchmark Datasets

20Newsgroups. The 20Newsgroups dataset has been widely used for evaluating the performance of cross-domain text classification algorithms. It is a collection of 18,774 newsgroup documents across 6 top categories and 20 subcategories in a hierarchical structure. Following the setting of TCA [1], we generate six cross-domain text datasets from 20Newsgroups, which are Comp *vs.* Rec, Comp *vs.* Sci, Comp *vs.* Talk, Rec *vs.* Sci, Rec *vs.* Talk, Sci *vs.* Talk. The details of the generated datasets are shown in Table I, in which one top-category (*e.g.*, Comp) is the positive class and another top-category (*e.g.*, Rec) is the negative class and under each top-category, there are some subcategories (*e.g.*, comp.graphics and rec.motorcycles).

Reuters-21578. The Reuters-21578 is another widely used dataset for evaluating the performance of text classification, which is also organized with a hierarchical structure as 20Newsgroups. Following the work [6], three cross-domain datasets are generated from the Reuters-21578. Table II shows the detailed information.

B. Baselines

We compare our baselines with both state-of-the-art cross-domain text classification models as well as deep learning models with the necessary domain adaption.

State-of-the-art cross-domain text classification models:

- Single Domain Support Vector Machine (SD-SVM) [1]: The SVM classifier is trained with the labeled documents in the source domain to predict the class labels of unlabeled documents in the target domain.
- Single Domain Logistic Regression (SD-LG) [1]: The LG classifier is also trained with the labeled documents from the source domain and used to predict the class labels of unlabeled documents in the target domain.
- Spectral Feature Alignment (SFA) [36]: The spectral clustering algorithm of SFA is adapted to co-cluster all words into the shared clusters for domain adaptation.
- Topic-bridge PLSA (TPLSA) [7]: TPLSA is a topic model method where all topics are assumed to be shared by different domains and used to represent documents.
- Collaborative Dual-PLSA (CDPLSA) [19]: CDPLSA is also a topic model. It jointly models different domains by assuming that the associations between the topics and the document categories are stable across domains.

TABLE I: Cross-domain tasks in 20Newsgroups.

Domain tasks	Source Domain D_s	Target Domain D_t
Comp vs Rec	comp.graphics comp.sys.ibm.pc.hardware rec.motorcycles rec.sport.baseball	comp.os.ms-windows.misc comp.sys.mac.hardware rec.autos rec.sport.hockey
Comp vs Sci	comp.os.ms-windows.misc comp.sys.ibm.pc.hardware sci.electronics sci.space	comp.graphics comp.sys.mac.hardware sci.crypt sci.med
Comp vs Talk	comp.os.ms-windows.misc comp.sys.ibm.pc.hardware talk.politics.mideast talk.politics.misc	comp.graphics comp.sys.mac.hardware talk.politics.guns talk.religion.misc
Rec vs Sci	rec.autos rec.sport.baseball sci.crypt sci.med	rec.motorcycles rec.sport.hockey sci.electronics sci.space
Rec vs Talk	rec.autos rec.sport.baseball talk.politics.mideast talk.politics.misc	rec.motorcycles rec.sport.hockey talk.politics.guns talk.religion.misc
Sci vs Talk	sci.crypt sci.med talk.politics.misc talk.religion.misc	sci.electronics sci.space talk.politics.guns talk.politics.mideast

TABLE II: Cross-domain tasks in Reuters-21578.

Domain tasks	Source Domain Ds	Target Domain Dt
Orgs vs People	Orgs.{...}, People.{...}	Orgs.{...}, People.{...}
Orgs vs Places	Orgs.{...}, Places.{...}	orgs.{...}, Places.{...}
People vs Places	People.{...}, Places.{...}	People.{...}, Places.{...}

- Topic Correlation Analysis (TCA) [1]: TCA extracts both the shared and the domain-specific latent features to facilitate effective knowledge transfer.

Deep learning models with adaption:

- mSDA [9]: mSDA utilizes a marginalized Stacked Denoising AutoEncoder to learn the latent high-level feature representation.
- $l_{2,1}$-SRA [10]: $l_{2,1}$-SRA utilizes a $l_{2,1}$-norm Stacked Robust AutoEncoder learn effective representations.
- MLP + GRL [18]: The feature generator is a 3-layer perceptron and a max-pooling layer to obtain the representation of a document. A gradient reverse layer (GRL) is added for domain classification.
- TextCNN + GRL [18]: The feature generator is a TextCNN architecture and a gradient reverse layer (GRL) is added to train a domain classifier.
- LSTM + GRL [18]: The feature generator is a LSTM layer and a gradient reverse layer (GRL) is added for domain classification.

C. Experimental Settings

All deep learning algorithms are implemented in TensorFlow and are trained with Adam optimizer. We follow the evaluation protocol in [1] and evaluate all approaches through grid search on the hyperparameter space and report the best results of each approach. For each deep approach, we use a

TABLE III: Classification accuracy comparisons on six cross-domain tasks generated from 20Newsgroups.

Domains	SVM	LG	SFA	TPLSA	CDPLSA	TCA	mSDA	$l_{2,1}$-SRA	MLP+GRL	TextCNN+GRL	LSTM+GRL	DAGNN
Comp vs. Rec	89.5	90.6	93.9	91.0	91.4	94.0	91.2	92.1	96.8	96.9	96.3	**97.5**
Comp vs. Sci	71.9	75.9	83.0	80.2	87.7	89.1	82.1	83.3	91.2	91.4	91.1	**91.7**
Comp vs. Talk	89.8	91.1	97.1	93.8	95.5	96.7	91.2	92.7	97.1	97.5	94.3	**97.9**
Rec vs. Sci	69.6	71.9	88.5	92.8	89.5	87.9	71.0	79.6	94.0	94.1	90.4	**95.1**
Rec vs. Talk	82.7	84.8	93.5	84.9	89.9	96.2	84.0	87.3	96.4	96.4	94.6	**97.3**
Sci vs. Talk	74.7	78.0	85.4	89.0	86.2	94.0	82.0	84.5	93.2	93.3	88.9	**93.9**
Average	79.7	82.1	90.2	88.6	90.0	93.0	83.6	86.6	94.8	94.9	92.6	**95.6**

TABLE IV: Classification accuracy comparisons on three cross-domain tasks from Reuters-21578.

Domains	SVM	LG	SFA	TPLSA	CDPLSA	TCA	mSDA	$l_{2,1}$-SRA	MLP+GRL	TextCNN+GRL	LSTM+GRL	DAGNN
Orgs vs. People	67.0	68.1	67.1	74.6	80.8	79.2	77.1	78.0	80.6	81.0	78.7	**82.3**
Orgs vs. Places	66.9	69.2	68.3	71.9	71.4	73.0	71.2	71.5	76.7	77.8	72.5	**78.9**
People vs. Places	52.0	51.3	50.6	62.3	54.8	62.6	65.3	65.5	65.6	65.7	57.7	**66.7**
Average	62.0	62.9	62.0	69.6	69.0	71.6	71.2	71.7	74.3	74.8	69.6	**76.0**

batch size of 128 samples, and a fixed learning rate $1e^{-4}$. We set the embedding size of words to 150 and set the dimension of document feature to 150 for all approaches. The adaptation rate λ is the following schedule: $\lambda = \min(\frac{2}{1+\exp(-10p)} - 1, 0.1)$, and the p is changing from 0 to 1 within the training process as in [18]. We use 2 graph pooling blocks in our model and set $\mathcal{C}^{(0)} = 20$, $\mathcal{C}^{(1)} = 5$, $d^{(0)} = d^{(1)} = d = 150, \alpha = 0.1$. The window size is set to 10.

D. Cross-Domain Classification Results

Tables III and IV list the accuracy of different methods on cross-domain classification tasks. From the results, we have the following observations:

(1) The SD-SVM and SD-LG methods obtain worse performance than the other methods. This is because the traditional shallow methods do not consider the concept of domains to make the knowledge gap smaller.

(2) Deep learning baselines mSDA and $l_{2,1}$-SRA do not have domain classifiers, and their performance is inferior to the MLP+GRL methods, confirming the superiority of domain-loss in cross-domain text classification.

(3) The deep learning approaches (MLP+GRL, TextCNN+GRL) have better performance than the traditional cross-domain text classification methods, which shows that the deep learning method has competitive advantages than traditional models. We also observe that the LSTM+GRL shows weaker performance than other deep learning baselines and our proposed model. This may because that the RNN-based methods are good at dealing with sequential information, but they are unable to model non-consecutive phrases and long-distance word dependency information to obtain the key relationship in long documents, which is very important in cross-domain text classification.

(4) The proposed DAGNN model consistently beats all the baselines on both datasets. It demonstrates that the proposed hierarchical graph neural network can better capture the underlying representation of the documents and reduce the distribution gap across domains by integrating the feature generator and the cross-domain mechanism into a unified framework.

E. Analysis of DAGNN Components

Because the proposed DAGNN contains multiple key components, in this section, we compare variants of DAGNN with respect to the following aspects to demonstrate the effectiveness of DAGNN — (1) the impact of domain-adversarial loss, (2) the effect of the graph neural network module, and (3) the impact of the number of graph pooling module. The following DAGNN variants are designed for comparison.

- DAGNN$\neg g$: A variant of DAGNN with the gradient reverse layer of DAGNN (*i.e.*, domain classifier) being removed.
- DAGNN$\neg p$: A variant of DAGNN with the pooling module of DAGNN being removed.
- DAGNN$\neg h$: A variant of DAGNN with the hierarchical structure of DAGNN being removed, and only using one graph pooling module instead of two.

In addition, we also compare our DAGNN model with the original GCN model (2-layer GCN and a max-pooling layer). For fairness of comparisons, we also add the gradient reverse layer to the original GCN model, which is GCN+GRL in our paper. The results are shown in Tables V and VI.

1) Impact of domain-adversarial loss: In order to verify the effectiveness of the domain-adversarial loss, we compare DAGNN model and DAGNN$\neg g$. From Tables V and VI, we can easily observe the DAGNN model performs significantly better than DAGNN$\neg g$. This confirms that the usage of domain-adversarial loss can learn a superior representation for texts from different domains.

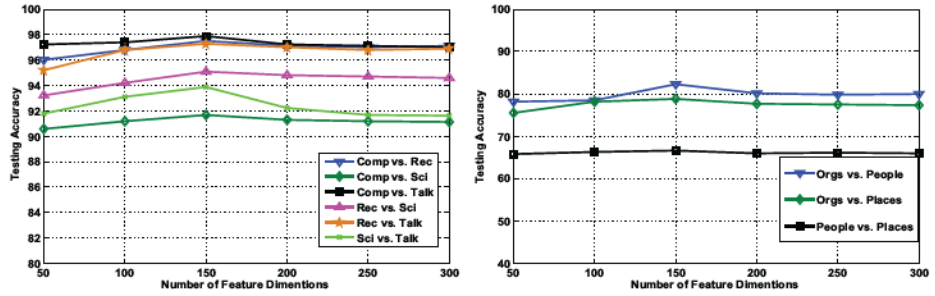
2) Effects of the graph neural network module: We compare DAGNN with DAGNN$\neg p$ to investigate the effectiveness of the novel GNN approach employed in our paper. From the result, we find that DAGNN$\neg p$ performs better than the GCN+GRL model, which confirms the superiority of the novel approximate personalized propagation scheme of our graph neural network in highlighting the inherent document-level connections. In the progress of the experiment, we also find

TABLE V: Classification accuracy comparisons between DAGNN variants on 20Newsgroups.

Domains	GCN+GRL	DAGNN$\neg p$	DAGNN$\neg h$	DAGNN$\neg g$	DAGNN
Comp vs. Rec	96.7	97.0	95.9	90.0	**97.5**
Comp vs. Sci	88.9	89.2	90.0	80.6	**91.7**
Comp vs. Talk	97.5	97.7	97.4	92.1	**97.9**
Rec vs. Sci	92.8	93.8	93.3	67.5	**95.1**
Rec vs. Talk	97.0	97.1	95.5	86.0	**97.3**
Sci vs. Talk	91.1	92.4	92.5	80.2	**93.9**
Average	94.0	94.5	94.1	82.7	**95.6**

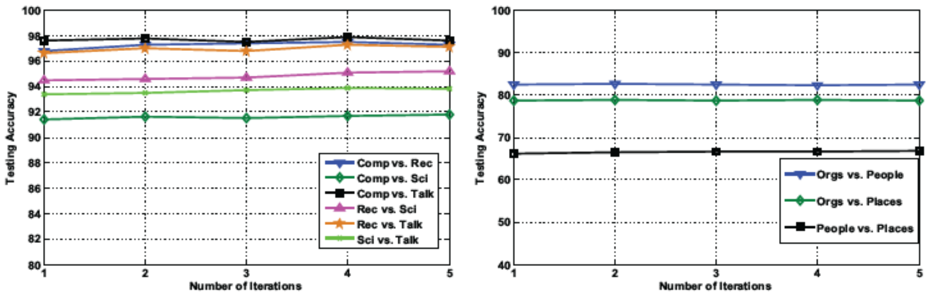
TABLE VI: Classification accuracy comparisons between DAGNN variants on Reuters-21578.

Domains	GCN+GRL	DAGNN$\neg p$	DAGNN$\neg h$	DAGNN$\neg g$	DAGNN
Orgs vs. People	79.9	81.2	80	72.9	**82.3**
Orgs vs. Places	74.6	76.4	78.1	70.3	**78.9**
People vs. Places	65.1	66.2	65.3	58.8	**66.7**
Average	73.2	74.6	74.5	67.3	**76.0**



(a) Testing accuracy on 20Newsgroups. (b) Testing accuracy on Reuters-21578.

Fig. 4: Classification results with different feature dimensions.



(a) Testing accuracy on 20Newsgroups. (b) Testing accuracy on Reuters-21578.

Fig. 5: Classification results with different iteration numbers of the proposed graph convolution network.

that the performance of GCN+GRL decreases quickly when we use the number of layers of GCN exceeding 2, while the DAGNN$\neg p$ can maintain the good performance. This indicates that the novel graph neural network model can ensure the stability of the GCN and avoid oversmoothing.

3) Impact of the number of graph pooling module: DAGNN uses two graph pooling modules to obtain hierarchical latent features of documents for classification. In order to show the superiority of the hierarchical graph pooling model, we design a variant model DAGNN$\neg h$. The only difference between DAGNN$\neg h$ and DAGNN is that DAGNN$\neg h$ uses only one graph pooling module instead of two graph pooling modules. The results in Tables V and VI show the performances of each classification task on both datasets are improved when two stacked graph pooling modules are used, indicating the effectiveness of the hierarchical graph pooling model.

F. Parameter Analysis

1) Impact of feature dimensions d: We set the number of feature dimensions of document representations as the same as that of word embeddings. That is, the word embeddings and the document representations are all d-dimensional feature vectors. We vary d from 50 to 300 and report the results of two classification tasks on two datasets respectively in Figure

4. When d increases from 50 to 150, the testing accuracy is improved on both datasets. Furthermore, only slight differences can be observed with different d and the increase of d does not necessarily result in performance improvements. The results show that with sufficient feature dimensions ($d \geq 200$), DAGNN is stable with the number of feature dimensions.

2) Impact of iteration numbers: In order to explore the effect of the number (j) of propagation steps of the GCN module, we set the number of iteration j from 1 to 5. Figure 5 shows how the accuracy of DAGNN depends on the number of iterations. From the result, we can observe only slight differences with different j and with the increase of the iteration number, the accuracy increases and converges in a stable condition.

VI. Conclusions

In this paper, we propose a DAGNN algorithm which uses domain-adversarial graph neural networks for cross-domain text classification. We argued that existing methods mainly model texts as word sequences, hard to capture semantics of long-distance word dependency and hierarchical structure of documents. In addition, existing cross-domain learning methods mainly focus on using other domains to improve the estimation of feature distributions, failing to use domains as extra source of supervisions. To address the challenges, our domain-adversarial graph neural networks propose to use three technical innovations: (1) using a graph to represent a document to capture non-consecutive and long-distance semantics, (2) using hierarchical graph neural networks to preserve structure of documents for feature learning, and (3) using domain-adversarial learning to jointly learn document classifier, as well as domain separation. The three approaches are combined to form an optimization framework, which uses gradient reverse layer [18] to learn domain-shared feature representations for cross-domain classification. Experiments and comparisons on real-world datasets demonstrate the effectiveness of our algorithm.

Acknowledgment

This work was supported by the US National Science Foundation (NSF) through Grants IIS-1763452 and CNS-1828181, and by the NSFC (No. 61872360), the Youth Innovation Promotion Association CAS (No. 2017210).

References

[1] L. Li, X. Jin, and M. Long, "Topic correlation analysis for cross-domain text classification," in *AAAI*, 2012, pp. 998–1004.

[2] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE TNN*, vol. 22, no. 2, pp. 199–210, 2010.

[3] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in nlp," in *ACL*, 2007, pp. 264–271.

[4] M. Chen, K. Q. Weinberger, and J. Blitzer, "Co-training for domain adaptation," in *NIPS*, 2011, pp. 2456–2464.

[5] L. Duan, I. W. Tsang, and D. Xu, "Domain transfer multiple kernel learning," *IEEE TPAMI*, vol. 34, no. 3, pp. 465–479, 2012.

[6] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Co-clustering based classification for out-of-domain documents," in *SIGKDD*, 2007, pp. 210–219.

[7] G.-R. Xue, W. Dai, Q. Yang, and Y. Yu, "Topic-bridged plsa for cross-domain text classification," in *SIGIR*, 2008, pp. 627–634.

[8] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE TKDE*, vol. 22, no. 10, pp. 1345–1359, 2010.

[9] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," *arXiv:1206.4683*, 2012.

[10] W. Jiang, H. Gao, F.-l. Chung, and H. Huang, "The l2, 1-norm stacked robust autoencoders for domain adaptation," in *AAAI*, 2016, pp. 1723–1729.

[11] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *JMLR*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[12] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] F. Rousseau, E. Kiagias, and M. Vazirgiannis, "Text categorization as a graph classification problem," in *ACL*, 2015, pp. 1702–1712.

[14] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *ICDM*, 2002, pp. 721–724.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv:1609.02907*, 2016.

[16] D. Marcheggiani, J. Bastings, and I. Titov, "Exploiting semantics in neural machine translation with graph convolutional networks," *arXiv:1804.08313*, 2018.

[17] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," *arXiv:1809.05679*, 2018.

[18] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *JMLR*, vol. 17, no. 1, pp. 2096–2030, 2016.

[19] F. Zhuang, P. Luo, Z. Shen, Q. He, Y. Xiong, Z. Shi, and H. Xiong, "Collaborative dual-plsa: mining distinction and commonality across multiple domains for text classification," in *CIKM*, 2010, pp. 359–368.

[20] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *ICML*, 2011, pp. 513–520.

[21] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," *arXiv:1409.7495*, 2014.

[22] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," *arXiv:1502.02791*, 2015.

[23] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv:1901.00596*, 2019.

[24] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *IJCAI*, 2019, pp. 1907–1913.

[25] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *IJCAI*, 2019, pp. 3670–3676.

[26] S. Pan, R. Hu, S.-f. Fung, G. Long, J. Jiang, and C. Zhang, "Learning graph embedding with adversarial training methods," *IEEE Transactions on Cybernetics*, 2019.

[27] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "Mgae: Marginalized graph autoencoder for graph clustering," in *CIKM*. ACM, 2017, pp. 889–898.

[28] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," *arXiv:1704.04675*, 2017.

[29] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang, "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," in *WWW*, 2018, pp. 1063–1072.

[30] A. Rahimi, T. Cohn, and T. Baldwin, "Semi-supervised user geolocation via graph convolutional networks," *arXiv:1804.08049*, 2018.

[31] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv:1607.01759*, 2016.

[32] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *EMNLP 2014*, 2014, pp. 1532–1543.

[33] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *ICLR*, 2019.

[34] T. H. Haveliwala, "Topic-sensitive pagerank," in *WWW*, 2002, pp. 517–526.

[35] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NIPS*, 2018, pp. 4800–4810.

[36] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment," in *WWW*, 2010, pp. 751–760.