Learning Domain Representation for Multi-Domain Sentiment Classification

Qi Liu[1], **Yue Zhang**[1] **and Jiangming Liu**[2]
Singapore University of Technology and Design, Singapore[1]
University of Edinburgh, UK[2]
{qi_liu, yue_zhang}@sutd.edu.sg
jiangming.liu@ed.ac.uk

Abstract

Training data for sentiment analysis are abundant in multiple domains, yet scarce for other domains. It is useful to leveraging data available for all existing domains to enhance performance on different domains. We investigate this problem by learning domain-specific representations of input sentences using neural network. In particular, a descriptor vector is learned for representing each domain, which is used to map adversarially trained domain-general Bi-LSTM input representations into domain-specific representations. Based on this model, we further expand the input representation with exemplary domain knowledge, collected by attending over a memory network of domain training data. Results show that our model outperforms existing methods on multi-domain sentiment analysis significantly, giving the best accuracies on two different benchmarks.

1 Introduction

Sentiment analysis has received constant research attention due to its importance to business (Pang et al., 2002; Hu and Liu, 2004; Choi and Cardie, 2008; Socher et al., 2012; Vo and Zhang, 2015; Tang et al., 2014). For multiple domains, such as movies, restaurants and digital products, manually annotated datasets have been made available. A useful research question is how to leverage resources available across *all* domains to improve sentiment classification on *a certain* domain.

One naive domain-agnostic baseline is to combine all training data, ignoring domain differences. However, domain knowledge is one valuable source of information available. To utilize this, there has been recent work on *domain-aware* models via multi-task learning (Liu et al., 2016; Nam and Han, 2016), building an output layer for each domain while sharing a representation network. Given an input sentence and a specific test domain, the output layer of the test domain is chosen for calculating the output.

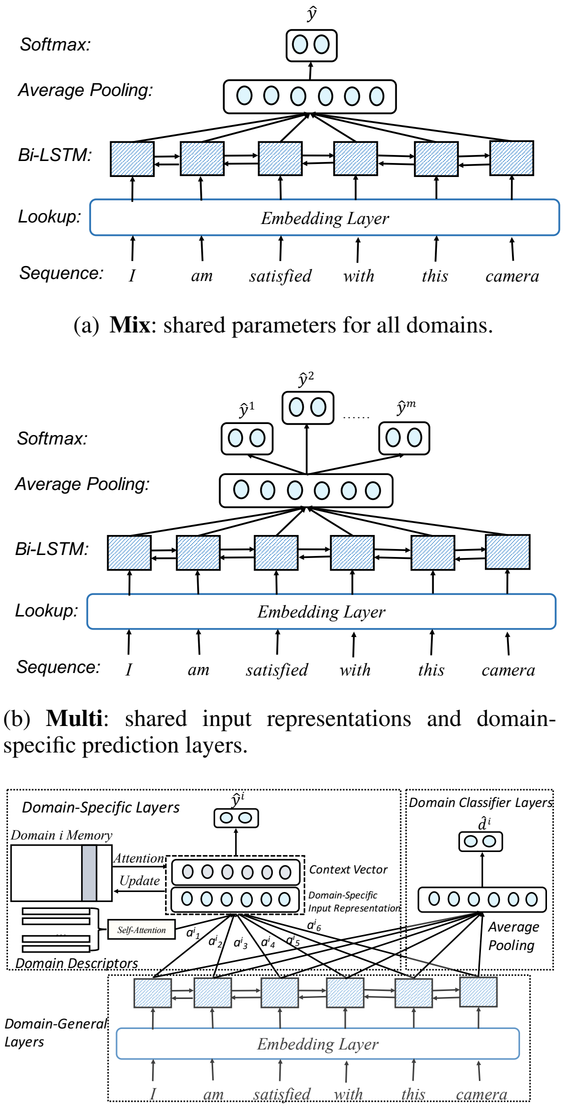
These methods have been shown to improve over the naive domain-agnostic baseline. However, a limitation is that outputs for different domains are constructed using the same domain-agnostic input representation, which leads to weak utilization of domain knowledge. For different domains, sentiment words can differ. For example, the word "beast" can be a positive indicator of camera quality, but irrelevant to restaurants or movies. Also, "easy" is frequently used in the electronics domain to express positive sentiment (e.g. the camera is easy to use), while expressing negative sentiment in the movie domain (e.g. the ending of this movie is easy to guess).

We address this issue by investigating a model that learns domain-specific input representations for multi-domain sentiment analysis. In particular, given an input sentence, our model first uses a bidirectional LSTM to learn a general sentence-level representation. For better utilizing data from all domains, we use adversarial training (Ganin and Lempitsky, 2015; Goodfellow et al., 2014) on the Bi-LSTM representation.

The general sentence representation is then mapped into a domain-specific representation by attention over the input sentence using explicitly learned *domain descriptors*, so that the most salient parts of the input are selected for the specific domain for sentiment classification. Some examples are shown in Figure 2, where our model pays attention to word "engaging" for movie reviews, but not for laptops, restaurants or cameras. Similarly, the word "beast" receives attention for laptops and cameras, but not for restaurants or movies.

In addition to the domain descriptors, we further introduce a memory network for explicitly representing *domain knowledge*. Here domain knowl-

Proceedings of NAACL-HLT 2018, pages 541–550
New Orleans, Louisiana, June 1 - 6, 2018. ©2018 Association for Computational Linguistics



(a) **Mix**: shared parameters for all domains.

(b) **Multi**: shared input representations and domain-specific prediction layers.

(c) Our model: domain knowledge is better utilized by domain descriptors, memories and adversarial training.

Figure 1: Models.

edge refers to example training data in a specific domain, which can offer useful background context. For example, given a sentence ‘Keep cool if you think it’s a wonderful life will be a heartwarming tale about life like finding nemo’, algorithms can mistakenly classify it as positive based on ‘wonderful’ and ‘heartwarming’, ignoring the fact that ‘it’s a wonderful life’ is a movie. In this case, necessary domain knowledge revealed in other sentences, such as ‘The last few minutes of the movie: it’s a wonderful life don’t cancel out all the misery the movie contained’ is helpful. Given a domain-specific input representation, we make attention over the domain knowledge memory network to obtain a background context vector, which is used in conjunction with the input representation for sentiment classification.

Results on two real-world datasets show that our model outperforms the aforementioned multi-task learning methods for domain-aware training, and also generalizes to unseen domains. Our code is released[1].

2 Problem Definition

Formally, we assume the existence of m sentiment datasets $\{D_i\}_{i=1}^{m}$, each being drawn from a domain i. D_i contains $|D_i|$ data points (s_j^i, d_i, y_j^i), where s_j^i is a sequence of words $w_1, w_2...w_{|s_j^i|}$, each being drawn from a vocabulary V, y_j^i indicates the sentiment label (e.g. $y_j^i \in \{-1, +1\}$ for binary sentiment classification) and d_i is a domain indicator (since we use 1 to m to number each domain, $d_i = i$). The task is to learn a function f which maps each input $(s_j^i,\ d_i)$ to its corresponding sentiment label y_j^i. The challenge of the task lies in how to improve the generalization performance of mapping function f both in-domain and cross-domain by exploring the correlations between different domains.

3 Baselines

3.1 Domain-Agnostic Model

One naive baseline solution ignores the domain characteristics when learning f. It simply combines the datasets $\{D_i\}_{i=1}^{m}$ into one and learns a single mapping function f. We refer to this baseline as **Mix**, which is depicted in Figure 1 (a).

Given an input s_j^i, its word sequence $w_1, w_2...w_{|s_j^i|}$ is fed into a word embedding layer to obtain embedding vectors $x_1, x_2...x_{|s_j^i|}$. The word embedding layer is parameterized by an embedding matrix $E_w \in R^{K\times|V|}$, where K is the embedding dimension.

Bidirectional LSTM: To acquire a semantic representation of input s_j^i, a bidirectional extension (Graves and Schmidhuber, 2005) of Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is applied to capture sentence-level semantics both left-to-right and right-to-left. As a result, two sequences of hidden states are obtained, denoted as $\overrightarrow{h}_1, \overrightarrow{h}_2...\overrightarrow{h}_{|s_j^i|}$ and $\overleftarrow{h}_1, \overleftarrow{h}_2...\overleftarrow{h}_{|s_j^i|}$, respectively. We concatenate $\overrightarrow{h}_t$

[1] https://github.com/leuchine/multi-domain-sentiment

and $\overleftarrow{h_t}$ at each time step to obtain the hidden states $h_1, h_2...h_{|s_j^i|}$, which are of sizes $2K$.

Output Layer: Average pooling (Boureau et al., 2010) is applied on the hidden states $h_1, h_2...h_{|s_j^i|}$ to obtain an input representation I_j^i for s_j^i,

$$I_j^i = \frac{\sum_{t=1}^{|s_j^i|} h_t}{|s_j^i|} \tag{1}$$

Finally, softmax is applied over I_j^i to obtain a probability distribution of all sentiment labels. During training, cross entropy is used as loss function, denoted as $L(f(s_j^i), y_j^i)$ for data points (s_j^i, d_i, y_j^i), and AdaGrad (Duchi et al., 2011) is applied to update parameters.

3.2 Multi-Domain Training

We build a second baseline for domain-aware sentiment analysis. A state-of-the-art architecture (Liu et al., 2016; Nam and Han, 2016) is used as depicted in Figure 1 (b), where m mapping functions f_i are learned for each domain. Given the input representation I_j^i obtained in Equation 1, multi-task learning is conducted, where each domain has a domain-specific set of parameters for softmax to predict sentiment labels with shared input representation layers. The input domain indicator d_i instructs which set of softmax parameters to use here and each domain has its own cross entropy loss $L_i(f_i(s_j^i, d_i), y_j^i)$ for data points (s_j^i, d_i, y_j^i). We denote this baseline as **Multi**.

4 Method

4.1 Domain-Aware Input Representation

The above baseline **Multi** achieves state-of-the-art performance for multi-domain sentiment analysis (Liu et al., 2016), yet the domain indicator d_i is used solely to select softmax parameters. As a result, domain knowledge is hidden and under-utilized. Similar to **Mix** and **Multi**, we use a Bi-LSTM to learn representations shared across domains. However, we introduce domain-specific layers to better capture domain characteristics as shown in Figure 1 (c).

Different domains have their own sentiment lexicons and domain differences largely lie in which words are relatively more important for deciding the sentiment signals. We use the neural attention mechanism (Bahdanau et al., 2014) to select words, obtaining domain-specific input representations.

In our model, *domain descriptors* are introduced to explicitly capture domain characteristics, which are parametrized by a matrix $N \in R^{2K \times m}$. Each domain descriptor corresponds to one column of N and has a length of $2K$, the same as the bidirectional LSTM hidden states h_t. This matrix is automatically learned during training.

Given an input (s_j^i, d_i), we apply an embedding layer and Bi-LSTM to generate its domain-general representation $h_1, h_2, ..., h_{|s_j^i|}$ and use the corresponding domain descriptor N_i to weigh $h_1, h_2, ..., h_{|s_j^i|}$ for obtaining a domain-specific representation. To this end, there are two most commonly used attention mechanisms: additive attention (Bahdanau et al., 2014) and dot product attention (Ashish Vaswani, 2017). We choose additive attention here, which utilizes a feed-forward network with a single hidden layer, since it achieves better accuracies in our development. The input representation I_j^i becomes a weighted sum of hidden states:

$$I_j^i = \sum_{t=1}^{|s_j^i|} a_{jt}^i h_t \quad s.t. \sum_{t=1}^{|s_j^i|} a_{jt}^i = 1 \tag{2}$$

The weight a_{jt}^i reflects the similarity between the domain i's descriptor N_i and the hidden state h_t. a_{jt}^i is evaluated as:

$$\begin{aligned} l_{jt}^i &= v^T tanh(PN_i + Qh_t) \\ a_{jt}^i &= \frac{exp(l_{jt}^i)}{\sum_{p=1}^{|s_j^i|} exp(l_{jp}^i)} \end{aligned} \tag{3}$$

Here $P \in R^{4K \times 2K}$, $Q \in R^{4K \times 2K}$ and $v \in R^{4K}$ are parameters of additive attention. P and Q linearly project N_i and h_t to a hidden layer, respectively. The projected space is set as $4K$ empirically, since we find it beneficial to project the vectors into a larger layer. v serves as the output layer. Softmax is applied to normalize l_{jt}^i. We name this method **DSR** for learning domain-specific representations.

4.2 Self-Attention over Domain Descriptors

DSR uses a single domain descriptor to attend over input words. However, relations between domains are not considered (e.g. sentiment lexicons for domain ‘camera’ are more similar to the lexicons of domain ‘laptop’ than those of domain ‘restaurant’). To model the interaction between domains, a self-attention layer is applied using dot product attention empirically, as shown in Figure 1 (c):

$$N_i^{new} = N\ softmax(N^T N_i) \tag{4}$$

We compute dot products between N_i and every domain descriptors. The dot products are normalized using the softmax function, and N_i^{new} is a weighted sum of all domain descriptors. N_i^{new} is used to attend over hidden states, employing Equation 2 and 3. During back propagation training, domain descriptors of similar domains could be updated simultaneously. We name this method **DSR-sa**, which denotes domain-specific representation with self-attention.

4.3 Explicit Domain Knowledge

To further capture domain characteristics, we devise a memory network (Weston et al., 2014; Sukhbaatar et al., 2015; Kumar et al., 2016) framework to explicitly represent domain knowledge. Our memory networks hold example training data of a specific domain for retrieving context data during predictions.

Formally, we use a memory $M^i \in R^{2K \times |D_i|}$ ($|D_i|$ is the total number of training instances of domain i) to hold domain-specific representations I_j^i of training instances for the domain i.

Memory Network: We directly set I_j^i as the jth column of the memory M^i. Formally,

$$M_j^i = I_j^i \tag{5}$$

Obtaining A Context Vector Using Background Knowledge: Given an input I_j^i, we generate a context vector C_j^i to support predictions by memory reading:

$$C_j^i = M^i\ softmax((M^i)^T I_j^i) \tag{6}$$

Dot product attention is applied here, which is faster and more space-efficient than additive attention, since it can be implemented using highly optimized matrix multiplication. Dot products are performed between I_j^i and each column of M^i and the scores are normalized using the softmax function. The final context vector is a weighted sum of M^i’s columns.

Output: We concatenate the context vector and the domain-specific input representation, feeding the result to softmax layers. Similar to the baseline **Multi**, each domain has its own loss $L_i(f_i(s_j^i, d_i), y_j^i)$. We name this method as **DSR-ctx** for context vector enhancements.

Reducing Memory Size: In the naive implementation, the memory size $|M^i|$ is equal to the total number of saved sequences, which can be very large in practice. We explore two ways to reduce memory size.

(1) Organizing memory by the vocabulary. We set $|M^i| = |V|$, where each memory column of M^i corresponds to a word in the vocabulary. During memory writing, I_j^i updates all the columns that correspond to the words w in its input sequence s_j^i by exponential moving average:

$$M_w^i = decay * M_w^i + (1 - decay) I_j^i$$

In this way, two input representations update the same column of the memory network if and only if they share at least one common word.

(2) Fixing the memory size by clustering. $|M^i|$ is set to a fixed size and I_j^i only updates the memory column that is most similar to I_j^i, i.e. I_j^i only update the column $arg\,max\ (M^i)^T I_j^i$. In this way, semantically similar inputs are clustered and update the same column.

4.4 Adversarial Training

We use embeddings and Bi-LSTM, parametrized by θ_{dg}, to generate domain-general representations. However, the distributions of domain-general representations for all domains can be different (Goodfellow et al., 2014), which contaminates the representations (Liu et al., 2017) and imposes negative effects for in-domain predictions. For cross-domain testing, the discrepancies cause domain shift, which harms prediction accuracies on target domains (Ganin and Lempitsky, 2015). Thus, models that can generate domain-invariant representations for all domains are favorable for utilizing multi-domain datasets.

We incorporate adversarial training to enhance the domain-general representations. As shown in Figure 1 (c), domain classifier layers are introduced, parametrized by θ_{dc}, which predicts how likely the input sequence s_j^i comes from each domain i. We denote its cross entropy loss as $L_{at}(f_{at}(s_j^i), d_i)$ for data points (s_j^i, d_i, y_j^i) from domain i (note that we use d_i as its label instead of input here).

Now consisting of domain-general layers, domain-specific layers and domain classifier layers, the model is trained by a minimax game. For dataset D_i drawn from domain i, we minimize its loss $L_i(f_i(s^i_j, d_i), y^i_j)$ for sentiment predictions, while maximizing the domain classifier loss $L_{at}(f_{at}(s^i_j), d_i)$, controlled by λ:

$$\min_{\theta_{dg},\theta_{ds}} \sum_{D_i} L_i(f_i(s^i_j, d_i), y^i_j) - \lambda L_{at}(f_{at}(s^i_j), d_i), \tag{7}$$

where θ_{ds} is the set of domain-specific parameters including domain descriptors, attention weights and softmax parameters. We fix θ_{dc} and update θ_{dg} and θ_{ds} here. Its adversarial part maximizes the loss by updating θ_{dc}, while fixing θ_{dg} and θ_{ds}.

$$\max_{\theta_{dc}} \sum_{D_i} L_i(f_i(s^i_j, d_i), y^i_j) - \lambda L_{at}(f_{at}(s^i_j), d_i) \tag{8}$$

Equations 7 and 8 are performed iteratively to generate domain-invariant representations. We name this method **DSR-at**.

5 Experiments

We evaluate the effectiveness of the model both in-domain and cross-domain. The former refers to the setting where the domain of the test data falls into one of the m training data domains, and the latter refers to the setting where the test data comes from one unknown domain.

5.1 Experimental Settings

We conduct experiments on two benchmark datasets. The datasets are balanced, so we use accuracy as the evaluation metric in the experiments.

The dataset 1 contains four domains. The statistics are shown in Table 1 , which also shows the accuracies using baseline method **Mix** trained and tested on each domain. *Camera*[2] consists of reviews with respect to digital products such as cameras and MP3 players (Hu and Liu, 2004). *Laptop* and *Restaurant* are laptop and restaurant reviews, respectively, obtained from SemEval 2015 Task 12[3]. Movie[4] are movie reviews provided by Pang and Lee (2004).

The dataset 2 is Blitzer's multi-domain sentiment dataset (Blitzer et al., 2007), which contains product reviews taken from Amazon.com, including 25 product types (domains) such as books, beauty and music. More statistics can be found at its official website[5].

Domain	Instance	Vocab Size	Accuracy
Camera (CR)	3770	5340	0.802
Laptop (LT)	1907	2837	0.871
Restaurant (RT)	1572	2930	0.783
Movie (M)	10662	18765	0.773

Table 1: Dataset 1 statistics.

Given each dataset, we randomly select 80%, 10% and 10% of the instances as training, development and testing sets, respectively.

5.2 Baselines and Hyperparameters

In addition to the **Mix** baseline, the **Multi** baseline (Liu et al., 2016) and our domain-aware models, **DSR**, **DSR-sa**, **DSR-ctx**, **DSR-at**, we also experiment with the following baselines:

MTRL (Zhang and Yeung, 2012) is a state-of-the-art multi-task learning method with discrete features. The method models covariances between task classifiers, and in turn the covariances regularize task-specific parameters. The feature extraction for **MTRL** follows (Blitzer et al., 2007). We use this baseline to demonstrate the effectiveness of dense features generated by neural models.

MDA (Chen et al., 2012) is a cross-domain baseline, which utilizes marginalized de-noising auto-encoders to learn a shared hidden representation by reconstructing pivot features from corrupted inputs.

FEMA (Yang and Eisenstein, 2015) is a cross-domain baseline, which utilizes techniques from neural language models to directly learn feature embeddings and is more robust to domain shift.

NDA (Kim et al., 2016) is a cross-domain baseline, which uses $m + 1$ LSTMs, where one LSTM captures global information across all m domains and the remaining m LSTM capture domain-specific information.

We set the size of word embeddings K to 300, which are initialized using the word2vec model[6] on news. To obtain the best performance, the parameters are set using grid search based on development results. The dropout ratio is chosen from $[0.3, , 1]$. Learning rate is chosen from

[2] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[3] Since the original dataset targets aspect-level sentiment analysis, we remove the sentences with opposite polarities on different aspects. The remaining sentences are labeled with the unambiguous polarity.

[4] https://www.cs.cornell.edu/people/pabo/movie-review-data/

[5] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

[6] https://code.google.com/archive/p/word2vec/

Dataset		Method						
Train	Test	MTRL	Mix	Multi	DSR	DSR-sa	DSR-ctx	DSR-at
LT+RT	*LT*	0.817	0.896	0.90	0.908	0.911	0.914	**0.92***
LT+RT	*RT*	0.781	0.820	0.85	0.860	0.859	0.863	**0.883***
LT+M	*LT*	0.825	0.882	0.90	0.887	0.90	0.904	**0.913***
LT+M	*M*	0.743	0.778	0.772	0.788	0.79	0.803	**0.811***
LT+CR	*LT*	0.869	0.904	0.906	0.921	0.915	0.92	**0.925**
LT+CR	*CR*	0.774	0.800	0.802	0.822	0.826	0.832	**0.844***
RT+M	*RT*	0.792	0.830	0.833	0.853	0.86	0.883	**0.9***
RT+M	*M*	0.729	0.765	0.785	0.795	0.801	0.816	**0.83***
RT+CR	*RT*	0.783	0.828	0.822	0.847	0.851	0.878	**0.887***
RT+CR	*CR*	0.756	0.804	0.814	0.812	0.817	0.831	**0.84***
M+CR	*M*	0.745	0.775	0.788	0.798	0.802	0.830	**0.839***
M+CR	*CR*	0.758	0.799	0.811	**0.819**	0.812	0.817	0.812
Average		0.778	0.818	0.832	0.842	0.845	0.857	**0.867***

Table 2: Results using two training domains on dataset 1. * denotes $p < 0.01$ VS. the second best using McNemar's test.

$[0.0001, 0.001, , 1]$. The vocabulary size is chosen from $[6000, 8000, , 16000]$. The batch size is chosen from $[10, , 100]$. λ is chosen from $[0.0001, 0.001, , 1]$. As a result, the mini-batch size, the size of the vocabulary V, dropout rate, learning rate for AdaGra and λ for adversarial training are set to 50, 10000, 0.4, 0.5 and 0.1, respectively. Also, gradient clipping (Pascanu et al., 2013) is adopted to prevent gradient exploding and vanishing during training process. Since all datasets only have thousands of instances, we set memory network sizes as training instance sizes in the experiments.

5.3 Working with Known Domains

In this section, we perform in-domain validations. We first combine two datasets for training and test on each domain's hold-out testing dataset. The results on dataset 1 are shown in Table 2 (the results on Blitzer's dataset exhibit similar results and are omitted due to space constraints).

The accuracies of **MTRL** are significantly lower than the neural models, which demonstrates the effectiveness of dense features over discrete features. The baseline **Mix** improves the average accuracy from 0.778 to 0.818, and most multi-domain training accuracies are better compared to single-domain training in Table 1. **Mix** simply combines the two datasets for trainings and ignores domain characteristics, yet improves over single dataset training. This demonstrates that more data reduces over-fitting and leads to better generalization capabilities. **Multi** further improves the average accuracy by 1.4%, which confirms the effectiveness of utilizing domain information.

Among our models, **DSR** further improves the accuracy over **Multi** by 1%, which confirms the effectiveness of domain-specific input representations in multi-domain sentiment analysis. **DSR-sa** slightly outperforms **DSR** by 0.03%. Adopting an additional self-attention layer, **DSR-sa** trains similar domain descriptors together, thus better modeling domain relations, which will be further studied in Section 5.5.2. **DSR-ctx** outperforms **DSR-sa** by 1.2%, which demonstrates the effectiveness of memory networks in utilizing domain-specific example knowledge. **DSR-at** gives significantly the best results, confirming that domain-invariant representations achieved by adversarial training indeed benefit in-domain training. The results are significant using McNeymar's test.

The results combining all the 4 domains and the 25 domains of the two datasets are shown in the 'In domain' sections of Table 3 and Table 4, respectively. Here the models are trained using all domains' training data, and tested on each domain's hold-out test data. Similar patterns are observed as in Table 2 and **DSR-at** achieves significantly the best accuracies (0.867 and 0.907 for the two datasets, respectively).

5.4 Working with Unknown New Domains

We validate the algorithms cross-domain. For dataset 1, models are trained on three domains, yet validated and tested on the other domain. For dataset 2, models are trained on 24 domains, yet validated and tested on the 25th.

Since **DSR-at** has m outputs (one for each training domain), we adopt an ensemble approach to obtain a single output for unknown test domains. In particular, since the domain classifier outputs probabilities on how likely the test data come from each training domain, we use these probabilities as weights to average the m outputs.

For **NDA**, **Multi**, **DSR** and **DSR-sa** and **DSR-ctx**, we use average pooling to combine the m outputs. Since **MDA** and **FEMA** are devised to train on a single source domain, we combine the training data of m domains for training.

The results are shown in the 'Cross domain' section of Table 3 and Table 4, respectively. One observation is that cross-domain accuracies are worse than in-domain accuracies, showing challenges in unknown-domain testing.

Contrast between our models and **FEMA/NDA** shows the advantage of leveraging resources from all domains, versus a single source domain for cross-domain modelling. Among the baselines,

	In domain							Cross domain									
Dataset	MTRL	Mix	Multi	DSR	DSR-sa	DSR-ctx	DSR-at	MTRL	Mix	MDA	Multi	FEMA	NDA	DSR	DSR-sa	DSR-ctx	DSR-at
LT	0.813	0.831	0.900	0.897	0.902	0.898	**0.915***	0.763	0.792	0.801	0.808	0.811	0.816	0.822	0.823	0.854	**0.878***
RT	0.776	0.801	0.825	0.841	0.845	0.855	**0.870***	0.772	0.786	0.789	0.779	0.774	0.776	0.78	0.784	0.814	**0.847***
M	0.800	0.803	0.783	0.807	0.812	0.820	**0.828***	0.616	0.636	0.642	0.668	0.679	0.684	0.692	0.695	0.725	**0.729**
CR	0.775	0.786	0.819	0.825	0.828	0.836	**0.854***	0.714	0.721	0.736	0.735	0.741	0.745	0.751	0.753	0.789	**0.809***
Average	0.791	0.805	0.832	0.843	0.847	0.852	**0.867***	0.716	0.734	0.742	0.748	0.751	0.755	0.761	0.764	0.796	**0.815***

Table 3: In-domain learning and cross-domain results on dataset 1. * denotes $p < 0.01$ VS. the second best.

	In domain							Cross domain									
Dataset	MTRL	Mix	Multi	DSR	DSR-sa	DSR-ctx	DSR-at	MTRL	Mix	MDA	Multi	FEMA	NDA	DSR	DSR-sa	DSR-ctx	DSR-at
Apparel	0.883	0.912	0.921	0.927	0.928	0.92	**0.938***	0.828	0.843	0.863	0.854	0.865	0.873	0.882	0.899	0.896	**0.909***
Electronics	0.853	0.881	**0.899**	0.884	0.879	0.883	0.891	0.804	0.826	0.836	0.849	0.845	0.834	0.857	0.859	0.861	**0.875***
Office	0.863	0.88	0.89	0.903	0.914	0.925	**0.933***	0.824	0.825	0.818	0.824	0.843	0.839	0.854	0.876	0.883	**0.894***
Automotive	0.842	0.864	0.873	0.886	0.891	0.902	**0.917***	0.791	0.786	0.791	0.797	0.816	0.826	0.835	0.847	0.857	**0.867***
Gourmet	0.814	0.838	0.84	0.852	0.856	0.858	**0.863***	0.777	0.775	0.764	0.784	0.796	0.803	0.814	0.826	**0.832**	0.828
Outdoor	0.853	0.889	0.899	0.903	0.907	0.915	**0.927***	0.785	0.796	0.805	0.815	0.836	0.829	0.856	0.861	0.867	**0.887***
Baby	0.816	0.853	0.86	0.875	0.877	0.892	**0.91***	0.803	0.816	0.814	0.821	0.834	0.84	0.845	0.878	0.873	**0.895***
Grocery	0.862	0.886	0.898	0.907	0.911	0.917	**0.933***	0.806	0.817	0.826	0.846	0.846	0.862	0.88	0.873	0.865	**0.886***
Software	0.851	0.876	0.88	0.893	0.898	0.904	**0.92***	0.795	0.811	0.816	0.836	0.845	0.836	0.85	0.862	0.884	**0.897***
Beauty	0.816	0.843	0.8567	0.862	0.867	0.864	**0.889***	0.756	0.768	0.775	0.785	0.795	0.804	0.812	0.812	0.838	**0.851***
Health	0.871	0.901	0.904	0.896	0.897	0.896	**0.907**	0.785	0.807	0.819	0.832	0.845	0.848	0.843	0.834	0.857	**0.871***
Sports	0.851	0.883	0.899	0.889	0.882	0.895	**0.9**	0.759	0.768	0.775	0.784	0.816	0.819	0.821	0.836	0.848	**0.864***
Book	0.743	0.803	0.79	0.804	0.809	0.815	**0.822***	0.694	0.705	0.716	0.723	0.745	0.743	0.751	0.758	0.779	**0.798***
Jewelry	0.816	0.891	0.881	0.893	0.891	0.894	**0.909***	0.762	0.769	0.774	0.785	0.795	0.808	0.815	0.835	0.857	**0.874***
Camera	0.912	0.937	0.968	0.966	0.959	0.968	**0.989***	0.869	0.878	0.886	0.896	0.894	0.908	0.917	0.925	0.942	**0.963***
Kitchen	0.815	0.858	0.863	0.875	0.887	0.894	**0.913***	0.759	0.768	0.775	0.776	0.794	0.818	0.826	0.856	0.865	**0.884** *
Toy	0.823	0.863	0.875	0.881	0.884	0.88	**0.892***	0.814	0.824	0.815	0.803	0.813	0.832	0.826	0.843	0.845	**0.857***
Phone	0.879	0.936	0.94	0.943	**0.949***	0.941	0.933	0.805	0.813	0.808	0.818	0.821	0.833	0.836	0.856	0.874	**0.894***
Magazine	0.835	0.874	0.872	0.883	0.895	0.917	**0.937***	0.805	0.819	0.817	0.816	0.83	0.841	0.845	0.857	0.871	**0.896***
Video	0.851	0.873	0.882	0.891	0.896	0.912	**0.925***	0.754	0.774	0.794	0.795	0.815	0.822	0.834	0.845	0.855	**0.875***
Games	0.867	0.886	0.89	0.883	0.886	0.887	**0.9***	0.681	0.684	0.708	0.718	0.723	0.734	0.746	0.765	**0.781**	0.778
Music	0.752	0.782	0.8	0.798	0.8	0.798	**0.81***	0.775	0.769	0.779	0.784	0.795	0.824	0.815	0.823	0.842	**0.858***
Dvd	0.795	0.826	0.834	0.847	0.854	0.867	**0.889***	0.801	0.794	0.804	0.794	0.814	0.827	0.835	0.845	0.851	**0.875***
Instrument	0.873	0.943	**0.957***	0.896	0.906	0.898	0.9	0.814	0.805	0.813	0.815	0.825	0.836	0.833	0.835	0.845	**0.865***
Tools	0.887	0.915	0.931	0.928	0.93	0.932	**0.94***	0.805	0.814	0.828	0.835	0.846	0.857	0.864	0.866	0.873	**0.897***
Average	0.841	0.875	0.884	0.887	0.89	0.895	**0.907***	0.786	0.794	0.801	0.807	0.82	0.827	0.835	0.847	0.858	**0.873***

Table 4: In-domain learning and cross-domain results on dataset 2. * denotes $p < 0.01$ VS. the second best.

NDA also considered domain-specific representations. On the other hand, it duplicates the full set of model parameters for each domain, yet underperforms **DSR** and **DSR-sa**, which records only one domain descriptor vector for each domain. The contrast shows the advantages of learning domain descriptors explicitly in terms of both efficiency and accuracy.

Similar to the known domain results, **DST-sa** and **DSR-ctx** further improve upon **DSR** and **DSR-sa**, showing the effectiveness of domain memory and adversarial learning. On both datasets, **DSR-at** achieves significantly the best performances, which shows the advantages of domain-invariant representations for unknown-domain testing.

5.5 Case Study

5.5.1 Input Attention

To obtain a better understanding of input attention with domain descriptors, we examine the attention weights of inputs and three examples are displayed in Figure 2, where the x axis denotes the four domains from the first dataset and the y axis shows the words.

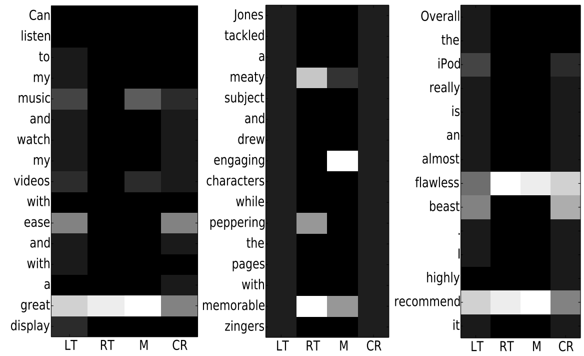
In Figure 2 (a), the domain-specific word 'ease' is only selected for the domains *LT* and *CR*, while the domain-independent word 'great' is salient in all domains. Similarly, in Figure 2 (b), 'meaty' and 'engaging' are only salient in *RT* and *M*, respectively. In Figure 2 (c), the domain-specific word 'beast' is chosen in **LT** and **CR**.

These confirm the effectiveness of input attention and **DSR-ctx** has the capability to pick out sentiment lexicons in conformity with domain characteristics.

5.5.2 Domain Descriptors

With the self-attention layer, one interesting question is whether learned domain descriptors can reflect domain similarities/dissimilarities.

We take out the twenty-five domain descriptors for Blitzer's dataset and calculate the cosine similarities between each pair. Also, we calculate the cosine similarities of twenty-five domains based on unigram and bigram representations for ground truth. Pearson correlation coefficient is used to measure the correlations between two sets of cosine values. The final score is 0.796, which shows that domain descriptor similarities can serve as indicators for domain similarities.



(a) Example1 (b) Example2 (c) Example3

Figure 2: Attention values (0: black, 1: white).

This place blew me away. By far my new favorite restaurant on the upper-east side.
(a1) This is one of my favorite spot, very relaxing. The food is great all the times. Celebrated my engagement and my wedding here. It was very well organized.
(a2) This is one of my favorite restaurants and it is not to be missed.
(a3) I didn't complain. I liked the atmosphere so much.
I started accessing and transferring files to find it to be extremely slow.
(b1) Only thing I don't like about it is slow in changing apps, boot up, and sometime it has problem connect through bluetooth.
(b2) I must say, this one is quite slow to open an application.
(b3) The subscription files are still a little slower to transfer, but it 's only by about 10% or so.
Keep cool if you think it's a wonderful life will be a heartwarming tale about life like finding nemo.
(c1) I heard so much about It's a wonderful life's happy ending and I just wasn't prepared for so much misery.
(c2) The last few minutes of the movie: its a wonderful life dont cancel out all the misery the movie contained.
(c3) It's a wonderful life was so incredibly over-sentimental and highly manipulative.

Table 5: Memory Network Attention.

5.5.3 Memory Network Attention

We further study the attention of memory networks by randomly picking instances in the test sets and listing the context instances with the greatest attention weights obtained from Equation 6. The results of three test instances and their context instances are shown in Table .

One observation is that semantically similar instances are selected to provide extra knowledge for predictions (e.g. a1, a2, b3, c1, c2, c3). Another observation is that the sentiment polarities between test instances and selected context instances are usually the same. We conclude that the memory networks are capable of selecting instructive instances for facilitating predictions.

6 Related Work

Domain Adaptation (Blitzer et al., 2007; Titov, 2011; Yu and Jiang, 2015) adapts classifiers trained on a source domain to an unseen target domain. One stream of work focuses on learning a general representation for different domains based on the co-occurrences of domain-specific and domain-independent features (Blitzer et al., 2007; Pan et al., 2011; Yu and Jiang, 2015; Yang et al., 2017). Another stream of work tries to identify domain-specific words to improve cross-domain classification (Bollegala et al., 2011; Li et al., 2012; Zhang et al., 2014; Qiu and Zhang, 2015). Different from previous work, we utilize *multiple* source domains for cross-domain validation, which makes our method more general and domain-aware.

Multi-domain Learning jointly learn multiple domains to improve generalization. One strand of work (Dredze and Crammer, 2008; Saha et al., 2011; Zhang and Yeung, 2012) uses covariance matrix to model domain relatedness, jointly learns domain-specific parameters and domain-independent parameters of linear classifiers. Another strand of work (Liu et al., 2016; Nam and Han, 2016) adopts neural network with shared input layers and multiple output layers for prediction. Our work belongs to the latter, yet we introduce domain descriptor matrix and memory networks to better capture domain characteristics and achieve better performance.

Memory Networks reason with inference components combined with a long-term memory component. Weston et al. (2014) devise a memory network to explicitly store the entire input sequences for question answering. An end-to-end memory network is further proposed by Sukhbaatar et al. (2015) by storing embeddings of input sequences, which requires much less supervision compared to Weston et al. (2014). Kumar et al. (2016) introduces a general dynamic memory network, which iteratively attends over episodic memories to generate answers. Xiong et al. (2016) extends Kumar et al. (2016) by introducing a new architecture to cater image inputs and better capture input dependencies. In similar spirits, our memory network stores the domain-specific training instances for obtaining context knowledge.

7 Conclusion

We investigated domain representations in multi-task learning for multi-domain sentiment analysis, showing that leveraging domain descriptors, examples and adversarial training to learn domain representations give significant improvements compared with strong multi-task learning baselines.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. Yue Zhang is the corresponding author.

References

Noam Shazeer Niki Parmar Ashish Vaswani. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45nd ACL*. volume 7, pages 440–447.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *Proceedings of the 49th ACL*. Association for Computational Linguistics, pages 132–141.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual recognition. In *ICML*. pages 111–118.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683* .

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 793–801.

Mark Dredze and Koby Crammer. 2008. Online methods for multi-domain learning and adaptation. In *EMNLP*. pages 689–697.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* pages 2121–2159.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*. pages 1180–1189.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. pages 2672–2680.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* pages 602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. MIT Press, volume 9, pages 1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD*. ACM, pages 168–177.

Young-Bum Kim, WA Redmond, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of COLING 2016*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*. pages 1378–1387.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *ACL*. Association for Computational Linguistics, pages 410–419.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101* .

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742* .

Hyeonseob Nam and Bohyung Han. 2016. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*. pages 4293–4302.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks* 22(2):199–210.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*. Association for Computational Linguistics, page 271.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *ACL*. Association for Computational Linguistics, pages 79–86.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML* 28:1310–1318.

Likun Qiu and Yue Zhang. 2015. Word segmentation for chinese novels. In *AAAI*. pages 2440–2446.

Avishek Saha, Piyush Rai, and Hal Daumé III Suresh Venkatasubramanian. 2011. Online learning of multiple tasks and their relationships. *update* .

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics, pages 1201–1211.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1555–1565.

Ivan Titov. 2011. Domain adaptation by constraining inter-domain variability of latent feature representation. In *Proceedings of the 49th ACL*. Association for Computational Linguistics, pages 62–71.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI*. pages 1347–1353.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916* .

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *ICML*. pages 2397–2406.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *HLT-NAACL*. pages 672–682.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345* .

Jianfei Yu and Jing Jiang. 2015. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. *Conference on Empirical Methods in Natural Language Processing* pages 236–246.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Type-supervised domain adaptation for joint segmentation and pos-tagging. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 588–597.

Yu Zhang and Dit-Yan Yeung. 2012. A convex formulation for learning task relationships in multi-task learning. *arXiv preprint arXiv:1203.3536* .