Wasserstein Distance Regularized Sequence Representation for Text Matching in Asymmetrical Domains

Weijie Yu[1], **Chen Xu**[2], **Jun Xu**[2,3,*], **Liang Pang**[4], **Xiaopeng Gao**[5], **Xiaozhao Wang**[5], **Ji-Rong Wen**[2,3]

[1]School of Information, Renmin University of China
[2]Gaoling School of Artificial Intelligence, Renmin University of China
[3]Beijing Key Laboratory of Big Data Management and Analysis Methods
[4]Institute of Computing Technology, Chinese Academy of Sciences; [5]Alibaba Group
{yuweijie, xc_chen, junxu, jrwen}@ruc.edu.cn, pangliang@ict.ac.cn,

Abstract

One approach to matching texts from asymmetrical domains is projecting the input sequences into a common semantic space as feature vectors upon which the matching function can be readily defined and learned. In real-world matching practices, it is often observed that with the training goes on, the feature vectors projected from different domains tend to be indistinguishable. The phenomenon, however, is often overlooked in existing matching models. As a result, the feature vectors are constructed without any regularization, which inevitably increases the difficulty of learning the downstream matching functions. In this paper, we propose a novel match method tailored for text matching in asymmetrical domains, called WD-Match. In WD-Match, a Wasserstein distance-based regularizer is defined to regularize the features vectors projected from different domains. As a result, the method enforces the feature projection function to generate vectors such that those correspond to different domains cannot be easily discriminated. The training process of WD-Match amounts to a game that minimizes the matching loss regularized by the Wasserstein distance. WD-Match can be used to improve different text matching methods, by using the method as its underlying matching model. Four popular text matching methods have been exploited in the paper. Experimental results based on four publicly available benchmarks showed that WD-Match consistently outperformed the underlying methods and the baselines.

1 Introduction

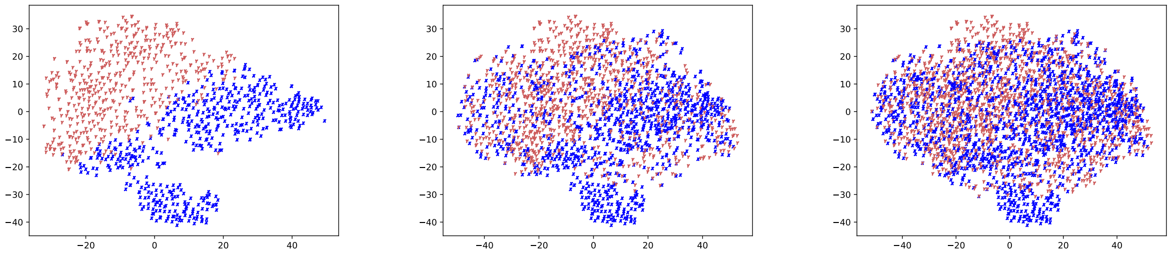
Asymmetrical text matching, which predicts the relationship (e.g., category, similarity) of two text sequences from different domains, is a fundamental problem in both information retrieval (IR) and natural language processing (NLP). For example, in natural language inference (NLI), text matching has been used to determine whether a hypothesis is entailment, contradiction, or neutral given a premise (Bowman et al., 2015). In question answering (QA), text matching has been used to determine whether a answer can answer the given question (Wang et al., 2007; Yang et al., 2015). In IR, text matching has been widely used to measure the relevance of a document to a query (Li and Xu, 2014; Xu et al., 2020).

One approach to asymmetrical text matching is projecting the text sequences from different domains into a common latent space as feature vectors. Since these feature vectors have identical dimensions and in the same space, matching functions can be readily defined and learned. This type of approach includes a number of popular methods, such as DSSM (Huang et al., 2013), DecAtt (Parikh et al., 2016), CAFE (Tay et al., 2018a), and RE2 (Yang et al., 2019). In real-world matching practices, it is often observed that learning of the matching models is a process of moving the projected feature vectors together in the semantic space. For example, Figure 1 shows the distribution of the feature vectors generated by RE2. During the training of RE2 (Yang et al., 2019) on SciTail dataset (Khot et al., 2018), it is observed that at the early stage of the training, the feature vectors corresponding to different domains are often separately distributed (according to the visualization by tNSE (Maaten and Hinton, 2008)) (Figure 1(a)). With the training went on, these separated feature vectors gradually moved closer and finally mixed together (Figure 1(b) and (c)).

The phenomenon can be explained as follows. Given two text sequences from two asymmetrical domains (e.g., NLI), the first sequence (e.g., premise) and the second sequence (e.g., hypothesis) are heterogeneous and there exists a lexical gap that needs to be bridged between them (Tay

[*] Corresponding author

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 2985–2994,
November 16–20, 2020. ©2020 Association for Computational Linguistics



(a) Epoch 1

(b) Epoch 10

(c) Epoch 20

Figure 1: t-SNE visualization of the projected feature vectors, based on the RE2 models trained on SciTail dataset. Subfigure (a), (b), and (c) respectively illustrates the vector distributions at epochs 1, 10, and 20. The blue 'X' and red 'Y' correspond to the premise and the hypothesis respectively.

et al., 2018c), similar to that of learning cross-modal matching model (Wang et al., 2017a). Existing studies (Wang et al., 2017a; Kamath et al., 2019) have shown that it is essentially critical that the projection network should generate domain- or modal-invariant features. That is, the global distributions of feature vectors should be similar in a common subspace such that their origins cannot be discriminated. The phenomenon is not unique but recurs in the experiments based on other matching models and other datasets.

Existing text matching models, however, are still lack of constraints or regularizations to ensure that the projected vectors are well distributed for matching. One natural question is, can we design a mechanism that can explicitly guide the mix of the feature vectors and better distribute them. To answer the question, this paper presents a novel learning to match method in which the Wasserstein distance (between the two distributions respectively corresponding to the two asymmetrical domains) is introduced as a regularizer, called WD-Match. WD-Match consists of three components: (1) a feature projection component which jointly projects each pair of text sequences into a latent semantic space, as a pair of feature vectors; (2) a regularizer component which estimates the Wasserstein distance with a feed-forward neural network on the basis of the projected features; (3) a matching component which conducts the matching, also on the same set of projected features.

The training of WD-Match amounts to repeatedly interplays between two branches under the adversarial learning framework: a regularizer branch that learns a neural network for estimating the upper bound on the dual form Wasserstein distance, and a matching branch that minimizes a Wasserstein distance regularized matching loss. In this way, the minimization of the loss function leads to a learning method not only to minimize the matching loss, but also to well distribute the feature vectors in the semantic space for better matching.

To summarize, this paper makes the following main contributions:

- We highlight the critical importance of the global distribution of the projected feature vectors in matching texts between asymmetrical domains, which has not yet been seriously studied in existing models.
- We propose a new learning to match method under the adversarial framework, in which the text matching model is learned by minimizing a Wasserstein distance-regularized matching loss.
- We conducted empirical studies on four large scale benchmarks, and demonstrated that WD-Match achieved better performance than the baselines and the underlying models. Extensive analysis showed the effects of Wasserstein distance-based regularizer in terms of guiding the distributions of feature vectors and improving the matching accuracy.

The source code of WD-Match is available at https://github.com/RUC-WSM/WD-Match

2 Related Work

In this section, we first review the sequence representation used in text matching, then introduce the Wasserstein distance and its applications.

2.1 Sequence Representation in Text Matching

Sequence representation lies in the core of text matching (Xu et al., 2020). Early works inspired by Siamese architecture assign respective neural networks to encode two input sequences into high-level representations. For example, DSSM (Huang et al., 2013) is one of the classic representation-based matching approaches to text matching which uses feed-forward neural networks to project a text sequence. CDSSM (Shen et al., 2014), ARC-I (Hu et al., 2014) and CNTN (Qiu and Huang, 2015) change sequence encoder to a convolutional neural network which shares parameters in a fixed size sliding window. To further capture the long-term dependence of a text sequence, a group of recurrent neural network based methods were proposed, including RNN-LSTM (Palangi et al., 2016) and MV-LSTM (Wan et al., 2015).

Recently, with the help of attention mechanism (Parikh et al., 2016), the sequence representation is obtained by aligning the sequence itself and the other sequence in the input pairs. For example, CSRAN (Tay et al., 2018b) performs multi-level attention refinement with dense connections among multiple levels. DRCN (Kim et al., 2019) stacks encoding layers and attention layers, then concatenates all previously aligned results. RE2 (Yang et al., 2019) introduces a consecutive architecture based on augmented residual connection between convolutional layers and attention layers. These models yield strong performance on several benchmarks.

2.2 Wasserstein Distance

Wasserstein distance (Chen et al., 2018) is a metric based on the theory of optimal transport. It gives a natural measure of the distance between two probability distributions.

Wasserstein distance has been successfully used in the Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) framework of deep learning. Arjovsky et al. (2017) propose WGAN which uses the Wasserstein-1 metric as a way to improve the original framework of GAN, to alleviate the vanishing gradient and the mode collapse issues in the original GAN. The Wasserstein distance has also been explored to learn the domain-invariant features in domain adaptation tasks. For example, Chen et al. (2018) propose to minimize the Wasserstein distance between the feature distributions of the source and the target domains, yielding better performance and smoother training than the standard training method with a Gradient Reversal Layer (Ganin et al., 2016). Shen et al. (2017b) propose to learn domain-invariant features with the guidance of Wasserstein distance.

Inspired by its success in variant applications, this paper introduces Wasserstien distance to text matching in asymmetrical domains, as a regularizer to improve the sequence representations.

3 Our Approach: WD-Match

In this section, we describe our proposed method WD-Match.

3.1 Model Architecture

Suppose that we are given a collection of N instances of sequence-sequence-label triples: $\mathcal{D} = \{(X_i, Y_i, \mathbf{z}_i)\}_{i=1}^N$ where $X_i \in \mathcal{X}$, $Y_i \in \mathcal{Y}$, and $\mathbf{z}_i \in \mathcal{Z}$ respectively denote the first sequence, the second sequence, and the label indicating the relationship of X_i and Y_i. As shown in Figure 2, WD-Match consists of three components:

The feature projection component: Given a sequence pair (X, Y), it is first processed by the feature projection component F,

$$[\mathbf{h}^X, \mathbf{h}^Y] = F(X, Y),$$

where the feature projection function F outputs a pair of K-dimensional feature vectors $\mathbf{h}^X, \mathbf{h}^Y$ in the semantic space. We suppose that F is a neural network with a set of parameters θ_F and all the parameters in θ_F are sharing for X and Y.

The matching component: The output vectors from the feature projection component are then fed to the matching component M,

$$\hat{\mathbf{z}} = M([\mathbf{h}^X, \mathbf{h}^Y]),$$

M outputs the predicted label $\hat{\mathbf{z}}$. We suppose that M is a neural network with a set of parameters θ_M.

The regularizer component: Given two sets of the projected feature vectors $\mathbf{h}^X$ and $\mathbf{h}^Y$, the regularizer component estimates the Wasserstein distance between $\mathbb{P}_F^X$ and $\mathbb{P}_F^Y$, we denote $\mathbb{P}_F^X$ and $\mathbb{P}_F^Y$ are two distributions defined over the two groups of feature vectors $\mathbf{h}^X$ and $\mathbf{h}^Y$ respectively.

$$\begin{aligned} \mathbb{P}_F^X &\triangleq P\left(\mathbf{h}^X | [\mathbf{h}^X, \mathbf{h}^Y] = F(X, Y) \wedge (X, Y) \sim \mathcal{X} \times \mathcal{Y}\right), \\ \mathbb{P}_F^Y &\triangleq P\left(\mathbf{h}^Y | [\mathbf{h}^X, \mathbf{h}^Y] = F(X, Y) \wedge (X, Y) \sim \mathcal{X} \times \mathcal{Y}\right), \end{aligned} \tag{1}$$

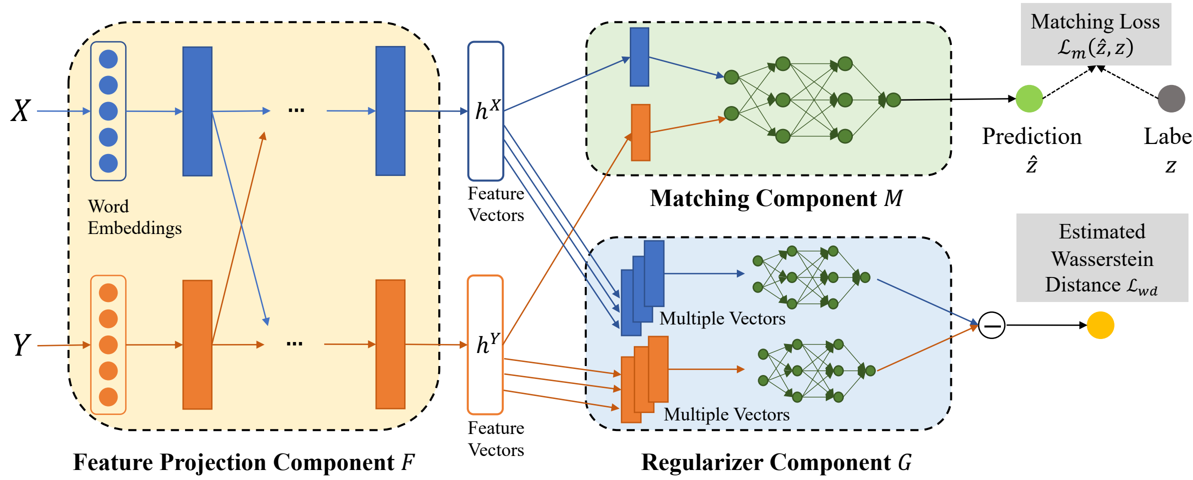


Figure 2: WD-Match architecture. Note that the multiple parallel arrow lines to the regularizer component G means that G takes a set of feature vectors (based on a batch of sequence pairs), rather than one feature vector, as its inputs.

where '$\sim$' means that the pairs (X, Y) are sampled from the joint space $\mathcal{X} \times \mathcal{Y}$. Specifically, the Wasserstein distance between two probabilistic distributions $\mathbb{P}_F^X$ and $\mathbb{P}_F^Y$ is defined as:

$$W(\mathbb{P}_F^X, \mathbb{P}_F^Y) = \inf_{\gamma \in \mathcal{J}(\mathbb{P}_F^X, \mathbb{P}_F^Y)} \int \|X - Y\| d\gamma(X, Y),$$

where $\mathcal{J}(\mathbb{P}_F^X, \mathbb{P}_F^Y)$ denotes all joint distributions, γ stands for (X, Y) that have marginal distributions $\mathbb{P}_F^X$ and $\mathbb{P}_F^Y$. It can be shown that W has the dual form (Villani, 2003):

$$W(\mathbb{P}_F^X, \mathbb{P}_F^Y) = \sup_{|G|_L \leq 1} E_{\mathbb{P}_F^X}[G(\mathbf{h}^X)] - E_{\mathbb{P}_F^Y}[G(\mathbf{h}^Y)], \tag{2}$$

where '$|G|_L \leq 1$' denotes that the 'sup' is taken over the set of all 1-Lipschitz[1] function G; and function $G : \mathcal{R}^K \rightarrow \mathcal{R}$ maps each K-dimensional feature vector in the semantic space to a real number. In this paper, G is set as a two-layer feed-forward neural network with a set of parameters θ_G clipped to $[-c, c]$, where $c > 0$ is a hyper-parameter.

Please note that different configurations of the feature projection component F, matching component M, and matching loss $\mathcal{L}_m$ leads to different matching models. Therefore, WD-Match can improve a matching model by setting the matching method as its underlying model.

3.2 Adversarial Training

To learn the model parameters $\{\theta_F, \theta_M, \theta_G\}$, WD-Match sets up two training goals: minimizing the Wasserstein distance between $\mathbb{P}_F^X$ and $\mathbb{P}_F^Y$, and minimizing the loss in prediction in terms of the mistakenly predicted matching labels. The training process can be implemented under the adversarial learning framework and amounts to repeatedly interplays between two learning branches: the regularizer branch and the matching branch.

In the regularizer branch, the objective term in the dual form Wasserstein distance (Equation (2)) is approximately written as:

$$\mathcal{O}_G(\theta_F, \theta_G) = \sum_{(X,Y)} \left[G(\mathbf{h}^X) - G(\mathbf{h}^Y)\right],$$

where $[\mathbf{h}^X, \mathbf{h}^Y] = F(X, Y)$ are the projected feature vectors for (X, Y). Maximizing $\mathcal{O}_G$ w.r.t. the parameters θ_G can achieve an approximation of the Wasserstein distance between $\mathbb{P}_F^X$ and $\mathbb{P}_F^Y$ in the semantic space defined by F:

$$\mathcal{L}_{wd}(\theta_F) = \max_{\theta_G} \mathcal{O}_G(\theta_F, \theta_G). \tag{3}$$

To make G a Lipschitz function (up to a constant) and following the practices in (Arjovsky et al., 2017), all of the parameters in θ_G are always clipped to a fixed range $[-c, c]$. In practice, the sequence pairs for training G are randomly sampled from the training set $\mathcal{D}$. Note that $\mathcal{L}_{wd}$ still takes θ_F as parameters because it is calculated on the basis of features generated by F.

[1] G is 1-Lipschitz $\Leftrightarrow |G(\mathbf{h}) - G(\mathbf{h}')| \leq |\mathbf{h} - \mathbf{h}'|$ for all $\mathbf{h}$ and $\mathbf{h}'$

The matching branch simultaneously updates the matching network M and feature projection network F by seeking the minimization of the Wasserstein distance-regularized matching loss:

$$\min_{\theta_F,\theta_M} \mathcal{L}_{reg} = \mathcal{L}_m(\theta_F, \theta_M) + \lambda \cdot \mathcal{L}_{wd}(\theta_F), \quad (4)$$

where $\lambda \in [0, 1]$ is a trade-off coefficient to balance the matching loss and regularizer, and $\mathcal{L}_m(\theta_F, \theta_M)$ is defined as

$$\mathcal{L}_m(\theta_F, \theta_M) = \sum_{(X,Y,z)\in\mathcal{D}} \ell_m(M(F(X, Y)), \mathbf{z}),$$

where $\ell_m(\cdot, \cdot)$ is the matching loss function defined over each sequence-sequence-label triple in the training data. It can be, for example, the cross-entropy loss that measure the goodness of the predicted label $\hat{\mathbf{z}} = M(F(X, Y))$ by the matching network, compared to the ground truth label $\mathbf{z}$.

Algorithm 1 shows the general procedure of WD-Match. WD-Match takes training set $\mathcal{D} = (X_i, Y_i, z_i)_{i=1}^N$ and a number of hyper-parameters as inputs, and outputs the learned parameters θ_F and θ_M. WD-Match run multiple rounds until convergence, and at each round it estimates the Wasserstein distance of the projected features and then update the projection component F and matching component M. At each round, WD-Match alternatively maintains two branches. The regularizer branch updates the parameters θ_G, with the θ_F fixed[2]. It contains a sub-iteration in which the parameters are optimized in an iterative manner: First, objective $\mathcal{O}_G$ is constructed based on the sampled sequence pairs (line 4 - line 6); Then θ_G is updated with gradient ascent (line 7); Finally, each parameter in θ_G is clipped to $[-c, c]$ for satisfying the 1-Lipschitz constraint (line 8). The matching branch updates θ_F and θ_M, with θ_G fixed. It first samples another mini-batch data from the training data and estimates the regularized loss $\mathcal{L}_{adv}$ using the fixed G (line 11 - line 13). Then, the gradients of the parameters is estimated and used to update the parameters (line 14).

4 Experiments

We conducted experiments to test the performances of WD-Match, and analyzed the results.

Algorithm 1 The WD-Match algorithm.

```
Require: Training set $\mathcal{D} = \{(X_i, Y_i, \mathbf{z}_i)\}_{i=1}^N$; mini-batch sizes $n_1$ and $n_2$; adversarial training step $k$; trade-off coefficient $\lambda$; learning rates $\eta_1$ and $\eta_2$; clipping threshold $c$
1:  repeat
2:      ▷ Regularizer branch
3:      for $t = 0$ to $k$ do
4:          Sample a mini-batch $\{(X_i, Y_i, \mathbf{z}_i)\}_{i=1}^{n_1}$ from $\mathcal{D}$
5:          $[\mathbf{h}_i^X, \mathbf{h}_i^X] \leftarrow F(X_i, Y_i), \forall i = 1, \cdots, n_1$
6:          $\mathcal{O}_G = \sum_{i=1}^{n_1}[G(\mathbf{h}_i^X) - G(\mathbf{h}_i^Y)]$
7:          $\theta_G \leftarrow \theta_G + \eta_1 \nabla_{\theta_G} \mathcal{O}_G$ {Eq. (3)}
8:          ClipWeights$(\theta_G, -c, c)$
9:      end for
10:     ▷ Matching branch
11:     Sample a mini-batch $\{(X_i, Y_i, z_i)\}_{i=1}^{n_2}$ from $\mathcal{D}$
12:     $[\mathbf{h}_i^X, \mathbf{h}_i^Y] \leftarrow F(X_i, Y_i), \forall i = 1, \cdots, n_2$
13:     $\mathcal{L}_{reg} = \sum_{i=1}^{n_2}[\ell_m(M(F(X, Y)), \mathbf{z}_i) + \lambda[G(\mathbf{h}_i^X) - G(\mathbf{h}_i^Y)]]$
14:     $\{\theta_F, \theta_M\} \leftarrow \{\theta_F, \theta_M\} - \eta_2 \nabla_{\theta_F,\theta_M} \mathcal{L}_{reg}$ {Eq. (4)}
15: until convergence
16: return $\{\theta_F, \theta_M\}$
```

Table 1: Statistics of four dataset used in our experiment, $|C|$ denotes the number of classes and R denotes a ranking formulation.

Dataset	Task	$\|C\|$	Pairs
SNLI	premise-hypothesis	3	570k
SciTail	premise-hypothesis	2	27k
TrecQA	question-answer	R	56k
WikiQA	question-answer	R	20k

4.1 Datasets and Metrics

We use four large scale publicly matching benchmarks: SNLI (Stanford Natural Langauge Inference) (Bowman et al., 2015), SciTail (Khot et al., 2018), TrecQA (Wang et al., 2007), WikiQA (Yang et al., 2015). Table 1 provides a summary of the datasets used in our experiments.

SNLI [3] is a benchmark for natural language inference. In SNLI, each data record is a premise-hypothesis-label triple. The premise and hypothesis are two sentences and the label could be "entailment", "neutral", "contradiction", or "-". In our

[2] Note that the regularizer does not depend on M, given F.

[3] https://nlp.stanford.edu/projects/snli

experiments, following the practices in (Bowman et al., 2015), the data with label "-" are ignored. We follow the original dataset partition. Accuracy is used as the evaluation metric for this dataset.

SciTail [4] is an entailment dataset based on multiple-choice science exams and web sentences. Each record is a premise-hypothesis-label triple. The label is "entailment" or "neutral", because scientific factors cannot contradict. We follow the original dataset partition. Accuracy are used as the evaluation metric for this dataset.

TrecQA [5] is a answer sentence selection dataset designed for the open-domain question answering setting. We use the raw version TrecQA, questions with no answers or with only positive/negative answers are included. The raw version has 82 questions in the development set and 100 questions in the test set. Mean average precision (MAP) and mean reciprocal rank (MRR) are used as the evaluation metrics for this task.

WikiQA [6] is a retrieval-based question answering dataset based on Wikipedia. We follow the data split of original paper. This dataset consists of 20.4k training pairs, 2.7k development pairs, and 6.2k testing pairs. We use MAP and MRR as the evaluation metrics for this task.

4.2 Experimental Setup

In WD-Match, different configurations of the feature projection component F and matching component M lead to different matching models. In the experiments, RE2 (Yang et al., 2019), DecATT (Parikh et al., 2016), CAFE (Tay et al., 2018a) and BERT (Devlin et al., 2018) were set as the underlying models, achieving new models respectively denoted as "WD-Match (RE2)", "WD-Match (DecAtt)", "WD-Match (CAFE)", and "WD-Match (BERT)".

Specifically, in WD-Match(RE2), F is a stacked blocks which consist of multiple convolution layers and multiple attention layers, and M is an MLP; in WD-Match(DecAtt), F is an attention layer and a aggregation layer, M is an MLP. Please note that we did not implement the Intra-Sentence Attention in our experiments; in WD-Match(CAFE), F is a highway encoder with a alignment layer and a factorization layer and M is another highway network. Please note that we remove the character embedding and position embedding in our experiments; in WD-Match(BERT), F is a pre-trained BERT-base[7] model, M is an MLP. Please note that for easing of combining with WD-Match, BERT was only used to extract the sentence features separately in our experiments. The G module for four models are identical: a non-linear projection layer and a linear projection layer.

For all models, the parameters of F and M were directly set as its original settings. In the training, all models were trained using the Adam optimizer with learning rate η_2 tuned amongst $\{0.0001, 0.0005, 0.001\}$. Batch size n_2 was tuned amongst $\{256, 512, 1024\}$. The trade-off coefficient λ was tuned from [0.0001, 0.01]. Clipping threshold was tuned from [0.1, 0.5]. Word embeddings were initialized with GloVe (Pennington et al., 2014) and fixed during training. We implemented WD-Match models in Tensorflow.

4.3 Experimental Results

Table 2 reports the results of WD-Match and the popular baselines on the SNLI test set. The baselines results are reported from their original papers. From the results, we found that WD-Match (RE2) outperformed all of the baselines, including the underlying model RE2. The results indicate the effectiveness of WD-Match and its Wasserstein distance-based regularizer in the asymmetric matching tasks of natural language inference. We further tested the performances of WD-Match (DecAtt) and WD-Match(BERT) which used DecAtt and BERT as the underlying matching models, respectively, to show whehter WD-Match can improve a matching method by using the method as its underlying model. From the results shown in Table 2, we can see that on SNLI, WD-Match (DecAtt) ourperform DecAtt in terms of accuracy. Similarly, WD-Match (BERT) improved BERT about 0.4 points in terms of accuracy.

Table 3 reports the results of WD-Match and the baselines on the SciTail test set. The baselines results are reported from the original papers. We found that WD-Match (RE2) outperformed all of the baselines. The result further confirm WD-match's effectiveness in the asymmetric matching task of scientific entailment. We also tested the performances of WD-Match (DecAtt) and WD-

[4] http://data.allenai.org/scitail/

[5] https://github.com/castorini/NCE-CNN-Torch/tree/master/data/TrecQA

[6] https://www.microsoft.com/en-us/download/details.aspx?id=52419

[7] https://github.com/google-research/bert

Table 2: Performance comparison on SNLI test set.

Models	Acc.(%)	#Params
BiMPM (Wang et al., 2017b)	86.9	1.6M
ESIM (Chen et al., 2016)	88.0	4.3M
DIIN (Gong et al., 2017)	88.0	4.4M
MwAN (Tan et al., 2018)	88.3	14M
HIM (Chen et al., 2016)	88.6	7.7M
SAN (Liu et al., 2018)	88.6	3.5M
CSRAN (Tay et al., 2018b)	88.7	13.9M
DRCN (Kim et al., 2019)	88.9	6.7M
RE2 (Yang et al., 2019)	89.0	2.8M
WD-Match (RE2)	**89.1**	2.9M
DecAtt (Parikh et al., 2016)	82.5	0.26M
WD-Match (DecAtt)	**82.6**	0.30M
BERT (Devlin et al., 2018)	83.7	0.11B
WD-Match (BERT)	**84.1**	0.11B

Table 3: Performance comparison on SciTail test set

Models	Acc.(%)
ESIM (Chen et al., 2016)	70.6
DGEM (Khot et al., 2018)	77.3
HCRN (Tay et al., 2018c)	80.0
CSRAN (Tay et al., 2018b)	86.7
RE2 (Yang et al., 2019)	86.6
WD-Match (RE2)	**87.0**
BERT (Devlin et al., 2018)	79.2
WD-Match (BERT)	**81.9**
DecAtt (Parikh et al., 2016)	81.7
WD-Match (DecAtt)	**82.9**

Match(BERT) on Scitail dataset. From the results shown in Table 3, we can see that WD-Match (DecAtt) improved DecAtt 1.2 points in terms of accuracy. Similarly, WD-Match (BERT) improved BERT about 2.7 points in terms of accuracy. The results verified that WD-Match's ability in improving its underlying model.

Table 4 reports the results of WD-Match and the baselines on the WikiQA test set. The baselines result are reported from the original papers. Following RE2, point-wise binary classification loss rather than pairwise ranking loss was used to train the model. The best hyperparameters including early stopping were tuned on WikiQA development set. From the results we can see that WD-Match (RE2) obtained a better result in terms of MAP and MRR on WikiQA. To further verify the effectiveness of WD-Match on QA task, we incorporated DecAtt and CAFE (Tay et al., 2018a) into WD-Match, then compare their performance to the respective underlying models on WikiQA and TrecQA datasets. Table 4 and Table 5 report the experimental results on WikiQA test set and TrecQA test set respectively. Similarly, WD-Match outperformed its underlying model on both datasets.

Table 4: Performance comparison on WikiQA test set.

Models	MAP(%)	MRR(%)
KVMN (Miller et al., 2016)	70.69	72.65
BiMPM (Wang et al., 2017b)	71.80	73.10
IWAN (Shen et al., 2017a)	73.30	75.00
CA (Wang and Jiang, 2016)	74.33	75.45
HCRN (Tay et al., 2018c)	74.30	75.60
RE2 (Yang et al., 2019)	74.96	76.58
WD-Match (RE2)	**75.31**	**76.89**
DecAtt (Parikh et al., 2016)	64.03	65.92
WD-Match (DecAtt)	**65.16**	**67.24**
CAFE (Tay et al., 2018a)	64.19	65.65
WD-Match (CAFE)	**66.36**	**67.59**

Table 5: Performance comparison on TrecQA test set.

Model	MAP(%)	MRR(%)
DecAtt (Parikh et al., 2016)	70.62	76.88
WD-Match (DecAtt)	**72.30**	**76.91**
CAFE (Tay et al., 2018a)	65.00	71.86
WD-Match (CAFE)	**67.49**	**73.05**

We list the number of parameters of different text matching models in Table 2. Compared to the underlying model, the additional parameters of WD-Match come from the regularizer component G. We can see that the parameters of the regularizer component G are far less than the underlying model. G module is implemented as a two-layer MLP (the number of neurons in the second layer is set as one). Therefore, the additional computing cost comes from the training of the two-layer MLP, which is of $O(T * N * K * 1)$, where T is the number of training iterations, N number of training examples, K number of neurons in the first layer of MLP (without considering the compute cost of the activation function). We can see that the additional computing overhead is much lower than that of the underlying methods which usually learn much more complex neural networks for the feature projection and the matching.

Summarizing the results above and the results reported in Section 4.3, we can conclude that WD-

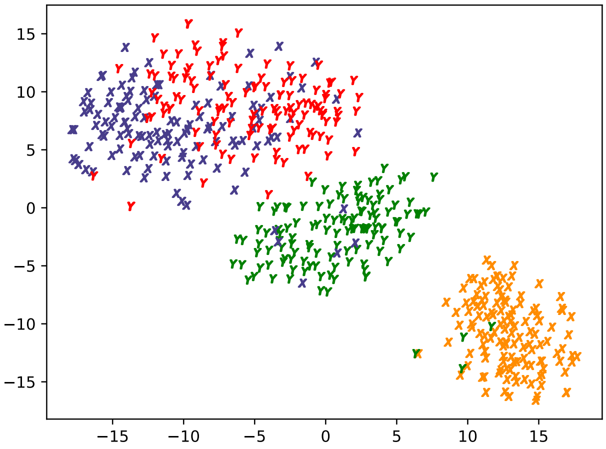


Figure 3: t-SNE visualization of the projected feature vectors, based on the RE2 and WD-Match (RE2) models trained on SciTail dataset. This figure illustrates the vector distributions after 5 training steps. The orange 'X' and green 'Y' correspond to $\mathbb{P}_F^X$ and $\mathbb{P}_F^Y$ of RE2, The dark blue 'X' and red 'Y' correspond to $\mathbb{P}_F^X$ and $\mathbb{P}_F^Y$ of WD-Match (RE2), respectively.

Match is a general while strong framework that can improve different matching models by using them as its underlying matching model.

4.4 Visualization of the Distributions of Feature Vectors

Figure 1(a) shows that there exists a gap between two feature vectors, due to the heterogenous nature of the texts from two asymmetrical domains. We conducted experiments to analyze how the feature vectors (i.e., $\mathbf{h}^X$ and $\mathbf{h}^Y$) generated by WD-Match distributed in the common semantic space, using WD-Match(RE2) as an example. Specifically, we trained a RE2 model and a WD-Match (RE2) model based on SciTail dataset. Note that in this experiment, the adversarial training step k is set as 5, that is, WD-Match (RE2) repeats regularizer branch for 5 times before matching branch. We recorded all of the training feature vectors (i.e., $\mathbf{h}^X$ and $\mathbf{h}^Y$) and illustrated them in the Figure 3 by t-SNE . The orange 'X' and green 'Y' correspond to $\mathbb{P}_F^X$ and $\mathbb{P}_F^Y$ of RE2, The dark blue 'X' and red 'Y' correspond to $\mathbb{P}_F^X$ and $\mathbb{P}_F^Y$ of WD-Match (RE2), respectively. As we can see from Figure 3, the feature vectors from RE2 are separately distributed while the feature vectors from WD-Match (RE2) are indistinguishable. It demonstrates that compared to the underlying model RE2, WD-Match (RE2) distributes the feature vectors in semantic space better and faster.

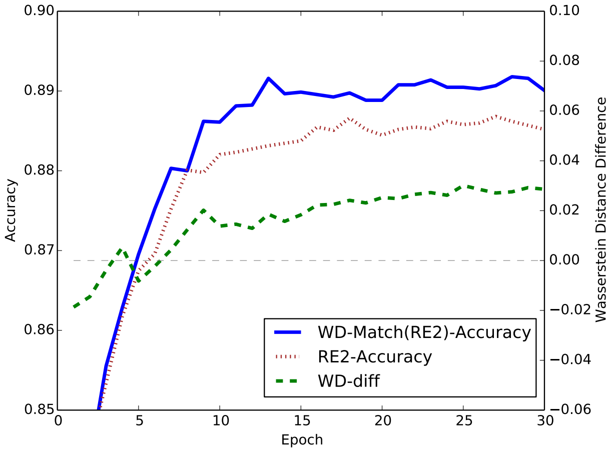


Figure 4: Accuracy curves and Wasserstein distance difference curve w.r.t. training epochs for RE2 and WD-Match (RE2).

4.5 Convergence and Effects of Wasserstein Distance-based Regularizer

We conducted experiments to test how the Wasserstein distance-based regularizer guides the training of matching models.

Specifically, we tested the WD-Match (RE2) and RE2 models generated at each training epochs. The accuracy curve on the basis of development set of SNLI was illustrated in Figure 4 (denoted as "WD-Match (RE2)-Accuracy" and "RE2-Accuracy"). Comparing these two training curves, we can see that WD-Match (RE2) outperformed RE2 when the training closing to converge (after about 15 epochs). We can conclude that WD-Match (RE2) obtained higher accuracy than RE2.

To investigate how the Wasserstein distance guides the training of matching models, we recorded the estimated Wasserstein distances at all of the training epochs of RE2 and WD-Match (RE2) based on the converged G network. The curve "WD-Diff" shows the differences between the Wasserstein distances by RE2 and that of by WD-Match (RE2) at each of the training epoch (i.e., $\mathcal{L}_{wd}(\theta_F)$ of RE2 minus $\mathcal{L}_{wd}(\theta_F)$ of WD-Match (RE2)). From the curve we can see that at the beginning of the training (i.e., epoch 1 to 5), the "WD-Diff" was near to zero. With the training went on (i.e., epoch 5 to 30), the Wasserstein distance by WD-Math(RE2) became smaller than that of by RE2 (the WD-Diff curve is above the zero line), which means that WD-Match (RE2)'s feature projection module F was guided to move feature vectors together more thoroughly and faster, which are more suitable for matching. The results indicate

WD-Match achieved its design goal of guiding the distributions of the projected feature vectors.

It is interesting to note that, comparing all of the three curves in Figure 4, we found the WD-Diff curve is close to zero at the beginning of the training, and the accuracy curves of WD-Match (RE2)-Accuracy and RE2-Accuracy are similar at the beginning. With the training went on (after epoch 10), the Wasserstein distance differences became larger. At the same time, the accuracy gaps (between WD-Match (RE2)-Accuracy and RE2-Accuracy) also become larger. The results clearly reflect the effects of Wasserstein distance-based regularizer: minimizing the regularizer leads to better distribution of feature vectors in terms of matching.

5 Conclusion and Future Work

In this paper, we proposed a novel Wasserstein distance-based regularizer to improve the sequence representations, for text matching in asymmetrical domains. The method, called WD-Match, amounts to adversarial interplay of two branches: estimating the Wasserstein distance given the projected features, and minimizing the Wasserstein distance regularized matching loss. We show that the regularizer helps WD-Match to well distribute the generated feature vectors in the semantic space, and therefore more suitable for matching. Experimental results on four benchmarks showed that WD-Match can outperform the baselines including its underlying models. Empirical analysis showed the effectiveness of the Wasserstein distance-based regularizer in text matching.

In the future, we plan to study different regularizers in the asymmetrical text matching task, for further exploring their effectiveness in bridging the gap between asymmetrical domains.

Acknowledgments

This work was funded by the National Key R&D Program of China (2019YFE0198200), the National Natural Science Foundation of China (61872338, 61832017, 61906180, 62006234), Beijing Academy of Artificial Intelligence (BAAI2019ZD0305), and Beijing Outstanding Young Scientist Program (BJJWZYJH012019100020098).

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 632–642.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

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.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338. ACM.

Anush Kamath, Sparsh Gupta, and Vitor Carvalho. 2019. Reversing gradients in adversarial domain adaptation for question deduplication and textual entailment tasks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5545–5550.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*, volume 17, pages 41–42.

Seonhoon Kim, Inho Kang, and Nojun Kwak. 2019. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6586–6593.

Hang Li and Jun Xu. 2014. Semantic matching in search. *Foundations and Trends® in Information Retrieval*, 7(5):343–469.

Xiaodong Liu, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for natural language inference. *arXiv preprint arXiv:1804.07888*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017a. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189. ACL.

Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2017b. Wasserstein distance guided representation learning for domain adaptation. *arXiv preprint arXiv:1707.01217*.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 101–110.

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway attention networks for modeling sentence pairs. In *IJCAI*, pages 4411–4417.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018a. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1565–1575, Brussels, Belgium. ACL. [link].

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018b. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4492–4502, Brussels, Belgium. ACL. [link].

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018c. Hermitian co-attention networks for text matching in asymmetrical domains. In *IJCAI*, pages 4425–4431.

Cédric Villani. 2003. *Topics in optimal transportation*. 58. American Mathematical Soc.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2015. A deep architecture for semantic matching with multiple positional sentence representations. *arXiv preprint arXiv:1511.08277*.

Bokun Wang, Yang Yang, Xing Xu, Alan Hanjalic, and Heng Tao Shen. 2017a. Adversarial cross-modal retrieval. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 154–162.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. pages 22–32.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *arXiv: Computation and Language*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017b. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Jun Xu, Xiangnan He, and Hang Li. 2020. Deep learning for matching in search and recommendation. *Foundations and Trends® in Information Retrieval*, 14(2–3):102–288.

Runqi Yang, Jianhai Zhang, Xing Gao, Feng Ji, and Haiqing Chen. 2019. Simple and effective text matching with richer alignment features. *arXiv preprint arXiv:1908.00300*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.