A Dynamic Interaction Network for Recognizing Text Entailment in Matrix Space (Postprint)

Authors: Huo Huan, Liu Liang

Date: 2018-06-19T00:00:00+00:00

Abstract

We propose a Dynamic Interactive Network (DIN) for recognizing textual entailment problems. Unlike existing interaction models, DIN projects the word vectors of two sentences into a two-dimensional matrix space for interaction, and then utilizes the output matrix to generate dynamic weights for a GRU encoder that simultaneously processes contextual information and controls information flow. The former mines deep logical fragments through higher-order forms of information interaction, while the latter helps the encoder effectively distinguish the importance differences between the two by altering the combination pattern of interaction information and contextual information. The model achieves 88.0% recognition accuracy on the SNLI test set, surpassing existing state-of-the-art models while using only half the training parameters.

Full Text

Preamble

http://www.arocmag.com/article/02-2019-10-005.html ChinaXiv Partner Journal *Computer Application Research*

A Dynamic Interactive Network over Matrix-Space for Recognizing Textual Entailment

Huo Huan[1,2], Liu Liang[1]†

(1. School of Optical-Electrical & Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China; 2. Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)

Abstract: This paper presents a dynamic interactive network (DIN) for recognizing textual entailment. Unlike other interactive models, DIN facilitates interaction by projecting embedding vectors into a two-dimensional matrix space, and then uses the output matrices to produce dynamic weights for the GRU

encoder that both processes context information and controls information flow. This approach empowers the extraction of logic segments through higher-order information interactions and helps the encoder better distinguish between the relative importance of context and interactive information. Experiments on the SNLI corpus show that our model achieves a test accuracy of 88.0%, outperforming the state-of-the-art while using only half the training parameters.

Keywords: textual entailment recognition; interactive network; matrix space; dynamic weights

Classification: TP399 doi: 10.3969/j.issn.1001-3695.2018.03.0196

Dynamic interactive network over matrix-space for recognizing textual entailment

(1. School of Optical-Electrical & Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China; 2. Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)

Abstract: This paper presented a dynamic interactive network (DIN) for recognizing textual entailment. Unlike other interactive models, DIN facilitates the interaction by projecting the embedding vectors into a two-dimensional matrix space, and then uses the output matrices to produce dynamic weights for the GRU encoder that both processes the context information and controls the information flow. It empowers the extraction of logic segments through higher-orders of information interactions and helps the encoder better choose between the context and the interactive information. Experiments on the SNLI corpus show that our model achieves a test accuracy of 88.0%, outperforming the state-of-the-art with only a small amount of the training parameters introduced.

Key words: textual entailment recognition; interactive network; matrix space; dynamic weights

0 Introduction

As research in natural language processing continues to deepen, many scholars face the challenge of enabling machines to truly understand natural language rather than merely grasping surface-level semantics. Among the numerous studies addressing this issue, one fundamental task is recognizing textual entailment (RTE), which relies on natural language sentences and aims to identify logical relationships between a premise (P) and a hypothesis (H) through reasoning (detailed definitions will be provided in the next section). This task represents an important practice in deep semantic mining and has significant applications in semantic search, human-computer dialogue, and question-answering systems.

In recent years, thanks to the success of neural networks in machine translation research, new approaches have emerged for recognizing textual entailment. Traditional recognition models with complex reasoning processes have gradually been abandoned in favor of end-to-end trainable neural network models.

Initially, such models typically employed encoders to independently encode the two sentences, commonly using either linear structures (e.g., GRU) or tree-based structures (e.g., quasi-TreeLSTM). After obtaining two encoding vectors that highly summarized the context information, various comparison methods (such as vector subtraction or element-wise multiplication) were applied to merge them before feeding them into a classifier for label prediction. However, when a sentence is compressed into a single vector representation after encoding, much important information is inevitably lost, making it difficult for such models to achieve high recognition accuracy.

To address this limitation, models further proposed using attention mechanisms to capture semantic connections between words. Compared to previous models, attention mechanisms perform reasoning through word-by-word matching, fully utilizing information preserved in each word' s encoding vector that was previously ignored, thereby significantly improving recognition accuracy. However, since sentence information becomes fixed in a single vector after encoding, the interactivity of such models cannot manifest as information flow between the two sentences, and thus they are considered to have only weak interactivity.

To promote genuine information flow between sentence pairs and better model their semantic-logical relationships, a series of strong interaction models have been proposed. A key characteristic of these models is that when encoding one sentence, they must simultaneously consider the encoded information of the other sentence. Due to their emphasis on information flow, strong interaction models generally achieve higher recognition accuracy than independent encoding and weak interaction models.

Based on the strong interaction concept, this paper proposes an efficient dynamic interactive network (DIN) for recognizing textual entailment. Figure 1 [Figure 1: see original paper] shows the overall structure of DIN, which generally follows the word-by-word interaction approach of previous work. Compared to existing strong interaction models, DIN makes two key modifications:

a) Inspired by continuation semantics, DIN projects the word embeddings of both sentences into a two-dimensional matrix space for interaction. Unlike traditional interaction models that use element-wise multiplication or weighted sums between vectors, matrix multiplication represents interaction between higher-dimensional information, which helps the model uncover deeper logical relationship fragments hidden within.

b) Inspired by dynamic network weight generation models, at each time step t when encoding the H sentence, the output matrix obtained from interaction can generate a set of weights specifically for the current time step: a weight triple (Wr t, Wh t, Wz t) that serves as the computation weights for the GRU encoder' s reset gate, update gate, and candidate activation at time t.

Notably, the GRU encoder in this strong interaction model plays two roles: encoding the current sentence' s context information and controlling informa-

tion flow between the two sentences. Traditional interaction models introduce the interaction vector from both sentences as an additional input during the word-by-word encoding process. A drawback of this approach is that it easily confuses context information with interaction information. Since both are inputs and both are in vector form, it is difficult for the encoder to weigh their relative importance, and when one contains excessive invalid information, it can easily overwrite the valid information in the other. In contrast, DIN leverages the advantage of matrixization to transform interaction information into computation weights for the encoder, fusing it with context information in a different form to minimize the adverse effects of direct superposition.

Experiments on the SNLI (Stanford Natural Language Inference) dataset demonstrate that DIN achieves higher recognition accuracy than existing interaction models while using fewer training parameters.

1 Related Work

There are few existing strong interaction models for sentence pair modeling, as such models emphasize the need to consider the other sentence' s information while encoding one sentence. Based on the direction of information flow, they can be roughly divided into two categories:

a) Unidirectional (generally flowing from P to H). Lei et al. proposed rLSTM, which modifies the classic word-by-word attention model by using the cell state rather than the hidden state of the LSTM encoder during H sentence encoding to align with the P sentence. This means P sentence information directly influences the encoding process of H at each time step, and the output encoding vectors necessarily contain interactive information from each other.

b) Bidirectional. Literature [8,9] proposed two strong interaction models: DF-LSTMs and Coupled-LSTMs. Unlike unidirectional models, they encode both sentences simultaneously, with interactive information participating in the encoding process of both sentences. Similar to LSTMN, DF-LSTMs use an additional memory region to store new interaction information obtained by comparing the current word with existing interaction information, where the existing interaction information is composed synchronously by reading interaction information saved in external memory for both sentences using attention mechanisms. Similar to SIN, Coupled-LSTMs abandon attention mechanisms and directly encode the context information of both sentences through grid-arranged LSTM cells, resulting in encoded vectors with stronger coupling characteristics.

A common feature of these models is that to satisfy information flow requirements, they modify the encoder' s computation process to incorporate the other sentence' s encoded information into the current sentence' s encoding process. This approach is similar to classic neural machine translation models, with the disadvantage that the influence is indirect and it is unclear how the encoder can

distinguish the importance difference between its own captured context information and the additional interaction information. In contrast, the dynamic interaction model DIN proposed in this paper uses dynamically generated weights to carry information flow, where the influence is direct and the mentioned differences can be quantified and displayed through changing weights (refer to Figure 3 [Figure 3: see original paper] and related analysis), facilitating understanding of the model' s mechanism.

2 Methodology

2.1 Problem Description

Recognizing textual entailment is a key task in machine understanding of natural language, with the core being the comprehension of natural logics. According to MacCartney et al., there are generally considered to be 16 basic semantic-logical relationships, 9 of which are degenerate (their expressions are relatively empty and rarely seen in practice). The remaining 7 logical relationships are unified into three major categories: entailment, contradiction, and neutral. Table 1 lists three examples with the same P sentence but different H sentences and labels. They share the P sentence: "families waiting in line at an amusement park for their turn to ride the carousel." Depending on the H sentence's wording, they are marked with different labels. In the first example, "people" in H and "families" in P have a clear equivalence relationship, and "at an amusement park" finds exact correspondence, so the sentence pair is judged to have entailment. In the second example, the actions "see a movie" in H and "ride the carousel" in P have asymmetry, thus they are judged contradictory. In the third example, the restaurant evaluation in H has no logical connection with P, so no clear judgment label can be given. These three examples are relatively simple for machines to judge correctly, but when a sentence contains excessive redundant information or clues distributed across multiple positions requiring holistic understanding, machines struggle to reason correctly.

An RTE task can be represented by the following triple:

$$(P, H, y) \tag{1}$$

where $P = (x'_1, \cdots, x'_{\ell_P})$ represents the P sentence of length ℓ_P, $H = (x_1, \cdots, x_{\ell_H})$ represents the H sentence of length ℓ_H, and y is the golden label. Given P and H, the model predicts the label for the sentence pair:

$$y^* = \arg\max_{y \in \gamma} \Pr(y|P, H) \tag{2}$$

where $\gamma = \{\text{Entailment}, \text{Contradiction}, \text{Neutral}\}$. The conditional probability $\Pr(y|P, H)$ is a vector where each dimension represents the probability of selecting a label from γ, and y^* is the label with the highest probability, whose solution will be discussed further.

Here, each x is a fixed-length word vector, typically initialized using pre-trained word vectors such as GloVe or word2vec. Words outside the vocabulary are initialized with random vectors of the same length.

2.2 Embedding Layer

In this layer, each word is mapped to a fixed-length vector representation. Generally, pre-trained word vectors like GloVe or word2vec are used for initialization. Words outside the vocabulary are initialized with random vectors of the same length.

2.3 Encoding Layer

As shown in Figure 1, information in our model flows unidirectionally from P to H, meaning P sentence information remains fixed. To better capture its context information, this layer uses a GRU encoder to encode it. GRU (Gated Recurrent Unit) was first proposed by Chung et al. as a variant of LSTM (Long Short-Term Memory Networks). It abandons the independent memory cell in LSTM, thus having a simpler control gate structure.

Specifically, the encoding vector h'_τ of the current word $x'_\tau, \tau \in [1, \cdots, \ell_P]$ is obtained through a linear interpolation function between the candidate state $\tilde{h}'_\tau$ and the previous state $h'_{\tau-1}$:

$$\mathbf{h}'_\tau = (1 - \mathbf{z}'_\tau)\mathbf{h}'_{\tau-1} + \mathbf{z}'_\tau \tilde{\mathbf{h}}'_\tau \tag{3}$$

where $\mathbf{z}'_\tau$ is the update gate that determines which information needs to be updated, calculated as:

$$\mathbf{z}'_\tau = \sigma(\mathbf{W}'_z \cdot [\mathbf{h}'_{\tau-1}, \mathbf{x}'_\tau] + \mathbf{b}'_z) \tag{4}$$

This shows the update gate is a linear sum of the previous state $h'_{\tau-1}$ and the current word x'_τ.

The candidate state $\tilde{h}'_\tau$ is calculated similarly to traditional recurrent units:

$$\tilde{\mathbf{h}}'_\tau = \tanh(\mathbf{W}'_h \cdot [\mathbf{r}'_\tau * \mathbf{h}'_{\tau-1}, \mathbf{x}'_\tau] + \mathbf{b}'_h) \tag{5}$$

where $\mathbf{r}'_\tau$ is the reset gate that determines which information needs to be reset, calculated similarly to the update gate:

$$\mathbf{r}'_\tau = \sigma(\mathbf{W}'_r \cdot [\mathbf{h}'_{t-1}] + \mathbf{b}'_r) \tag{6}$$

The $\mathbf{W}'_*$ above are weight matrices and bias vectors for each control gate' s computation.

2.4 Transformation Layer

Inspired by programming language theory, Baker et al. proposed the concept of continuation semantics. They argued that a large portion of expressions in natural language are obtained through semantic composition, but not through simple word concatenation; instead, they need to be transformed into higher-order forms. This work does not directly involve semantic composition, but to some extent, matching logical relationships between word pairs is similar to semantic composition, as both work by comparing and fusing word pairs based on their semantic relationships. Therefore, this layer projects the P sentence' s embedding layer word vectors and the H sentence' s word vectors encoded by the previous layer into a two-dimensional matrix space, as follows:

$$\tilde{\mathbf{W}} = \tanh(\mathbf{W}_{\text{trans}}\mathbf{c} + \mathbf{b}_{\text{trans}}) \tag{7}$$

where $\mathbf{W}_{\text{trans}} \in \mathbb{R}^{\sqrt{d}\times\sqrt{d}\times d_c}$, $\mathbf{b}_{\text{trans}} \in \mathbb{R}^{\sqrt{d}\times\sqrt{d}}$, d is the encoding vector length, and d_c is the vector $\mathbf{c}$ length. For the P sentence, $\mathbf{c}$ is the encoding vector $\mathbf{h}'_t \in \mathbb{R}^d, \tau \in [1, \cdots, \ell_P]$; for the H sentence, $\vec{\mathbf{c}}$ is the word embedding vector $\mathbf{x}_t \in \mathbb{R}^{d_{\text{emb}}}, t \in [1, \cdots, \ell_H]$. The transformation layer' s output $\tilde{\mathbf{W}}$ is a matrix of size $\sqrt{d} \times \sqrt{d}$. Considering the square root operation, in practice d is generally set to values that can be square-rooted, such as 256, 324, and 400.

2.5 Interaction Layer

After the previous layer' s transformation, we obtain higher-order matrix representations for each word. To promote interaction between matrices, this layer adopts a word-by-word matching approach similar to attention mechanisms, with the difference that when aligning the H sentence' s current word with each word in the P sentence, no weights are assigned to P sentence words. Instead, simpler matrix multiplication is used to achieve one-by-one interaction between the current word and each word in the P sentence. Specifically, the interaction between the H sentence' s current word x_t and the P sentence' s x'_τ is:

$$\mathbf{W}_t = \tanh(\tilde{\mathbf{W}}_\tau \mathbf{t} + \mathbf{b}_{\text{inter}}) \tag{8}$$

where $\mathbf{W}_{\text{inter}} \in \mathbb{R}^{\sqrt{d}\times\sqrt{d}}$, $\mathbf{b}_{\text{inter}} \in \mathbb{R}^{\sqrt{d}\times\sqrt{d}}$. This computation can be analogized to a simple recurrent neural network unit for matrix operations: $\mathbf{W}_t = \text{RNN}(\mathbf{W}_{\tau-1}, \tilde{\mathbf{W}}_t)$.

2.6 Control Layer

After the previous layer' s interaction, at each time step t of the H sentence, an output matrix $\mathbf{W}_t$ is produced. Similar to the encoding layer, the core of this layer is also a GRU encoder, but while incorporating interactive information into the encoding of the H sentence, it also controls the degree of information flow between the two sentences. Therefore, we call this layer the control layer. Unlike existing strong interaction models that modify the encoder's computation

steps by introducing the two sentences' interactive information as an additional input source, our model makes no modifications to the GRU itself. Instead, leveraging the advantage of matrixization, we use the output matrix from the interaction layer to dynamically generate the three weight matrices required for GRU computation: update gate $\mathbf{W}_z^t$, reset gate $\mathbf{W}_r^t$, and candidate activation $\mathbf{W}_h^t$.

Noting that the interaction layer' s output matrix ($\sqrt{d} \times \sqrt{d}$) and this layer' s two inputs $\mathbf{x}_t$, $\mathbf{h}_{t-1}$ (d) are dimensionally asymmetric, this paper proposes two methods to implement this GRU control layer with dynamic weights:

a) **DIN-1**. Convert the two d-dimensional input vectors into $\sqrt{d} \times \sqrt{d}$ matrices:

$$\mathbf{H}_{t-1} = \text{TO-MATRIX}(\mathbf{h}_{t-1}) \tag{9}$$

$$\mathbf{Z}_t = \sigma(\mathbf{W}_z^t \cdot [\mathbf{H}_{t-1}, \tilde{\mathbf{W}}_t] + \mathbf{b}_z^t) \tag{10}$$

$$\mathbf{R}_t = \sigma(\mathbf{W}_r^t \cdot [\mathbf{H}_{t-1}, \tilde{\mathbf{W}}_t] + \mathbf{b}_r^t) \tag{11}$$

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{W}_h^t \cdot [\mathbf{R}_t * \mathbf{H}_{t-1}, \tilde{\mathbf{W}}_t] + \mathbf{b}_h^t) \tag{12}$$

$$\mathbf{H}_t = (1 - \mathbf{Z}_t) * \mathbf{H}_{t-1} + \mathbf{Z}_t * \tilde{\mathbf{H}}_t \tag{13}$$

$$\mathbf{h}_t = \text{VECTORIZE}(\mathbf{H}_t) \tag{14}$$

where:

$$\mathbf{W}_*^t = \tanh(\mathbf{W}_{\text{con}}^* \cdot \mathbf{W}_t + \mathbf{B}_{\text{con}}^*) \tag{15}$$

with $\mathbf{W}_{\text{con}}^* \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}$, $\mathbf{b}_{\text{con}}^* \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}$, $\mathbf{B}_{\text{con}}^* \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}$.

b) **DIN-2**. Convert the $\sqrt{d} \times \sqrt{d}$ matrix output from the interaction layer into a d-dimensional vector:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z^t \cdot [\mathbf{h}_{t-1}, \mathbf{x}_{t-1}] + \mathbf{b}_z^t) \tag{17}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r^t \cdot [\mathbf{h}_{t-1}, \mathbf{x}_{t-1}] + \mathbf{b}_r^t) \tag{18}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h^t \cdot [\mathbf{r}_t * \mathbf{h}, \mathbf{x}_{t-1}] + \mathbf{b}_h^t) \tag{19}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) * \mathbf{h}_{t-1} + \mathbf{z}_t * \tilde{\mathbf{h}}_t \tag{20}$$

where:

$$\mathbf{W}_*^t = \text{VECTORIZE}(\mathbf{W}_{\text{lp}} \mathbf{W}_t) \tag{21}$$

$$\mathbf{W}_*^t = \tanh(\mathbf{W}_{\text{con}}^* \cdot \mathbf{W}_*^t + \mathbf{b}_{\text{con}}^*) \tag{22}$$

The difference between the two methods is: DIN-1 has slightly more complex computation requiring multiple conversions between vectors and matrices, but uses fewer training parameters than DIN-2, though this may limit its ability to learn more expressions.

2.7 Output Layer

In the model' s final layer, average pooling is used to combine the control layer' s outputs at each time step into a fixed-length vector:

$$\bar{\mathbf{h}} = \frac{1}{\ell_H} \sum_{t=1}^{\ell_H} \mathbf{h}_t \tag{24}$$

where ℓ_H is the length of the H sentence. At this point, all word or phrase-level reasoning is fused to determine the final judgment. Then $\bar{\mathbf{h}}$ is fed into a multi-layer perceptron (MLP) containing two fully-connected layers and a softmax classifier:

$$\bar{\mathbf{h}} = \mathrm{FC}(\bar{\mathbf{h}}) \tag{25}$$

$$\Pr(y|P,H) = \mathrm{softmax}(\mathbf{W}_{\text{output}}\bar{\mathbf{h}} + \mathbf{b}_{\text{output}}) \tag{26}$$

where $\mathbf{W}_{\text{output}} \in \mathbb{R}^{3\times d}$, $\mathbf{b}_{\text{output}} \in \mathbb{R}^3$. $\Pr(y|P,H) \in \mathbb{R}^3$ is the conditional probability in equation (1), and the label corresponding to the dimension with the highest probability value is selected as the prediction.

2.8 Complexity Analysis

To help readers better understand the training efficiency of the DIN model, this section provides some complexity analysis. For simplified expression, both word vector and encoding vector lengths are uniformly defined as d, and sentence length is denoted by ℓ.

In the encoding layer, the complexity of multiplication operations between weight matrices ($d \times d$) and vectors (d) in the GRU cell is $O(d^2)$. Therefore, the complexity of encoding an entire sentence is $O(\ell d^2)$. In the transformation layer, projecting both sentences from word vectors (d) to matrix space ($\sqrt{d} \times \sqrt{d}$) has complexity $O(\ell d^2)$. In the interaction layer, a nested loop-like structure makes its complexity $O(\ell^2 d)$. In the control layer, depending on the dynamic weight generation method, DIN-1 has complexity $O(\ell d)$, while DIN-2 uses re-parameterization, increasing complexity to $O(\ell d^2)$.

In summary, the overall complexities of DIN-1 and DIN-2 are $O(\ell d^2 + \ell^2 d + \ell d)$ and $O(\ell d^2 + \ell^2 d + \ell d^2)$, respectively. Observations show that ℓ is generally smaller than d; for example, in the SNLI dataset, the average and maximum lengths of P and H sentences are 34/68 and 28/58, respectively, while pre-trained word vector dimensions are generally around 300. Therefore, the complexity of both models is mainly concentrated in the interaction layer with complexity $O(\ell^2 d)$.

2.9 Training Objective

Since RTE is essentially a classification problem, this paper adopts cross-entropy loss as the training objective function. Specifically, when given the ground-truth

labels y_i for each sentence pair in the training set and the training parameter set θ, the objective function is organized as:

$$J(\theta) = -\sum_{i=1}^{N} y_i \log y_i + \lambda \|\theta\|_2 \tag{27}$$

where N is the training set size, y_i can be obtained from equation (1), and λ is the ℓ_2 regularization parameter. Through small-scale grid search within [0.0, 1E-4, 3E-4, 1E-3], it was found that using the ℓ_2 regularization term actually slows down the training process and is accompanied by a certain degree of performance loss. Therefore, ℓ_2 regularization is not used during training to forcibly constrain parameter updates.

3 Experiments

The experiments use the SNLI dataset released by Bowman et al. in 2015, which is widely used to test the performance of neural network models designed for RTE tasks. This dataset contains a total of 570,152 sentence pairs, each manually labeled with one of the following labels: entailment, contradiction, neutral, or "-" (a special label indicating that multiple annotators did not reach consistent agreement on the sentence pair). Consistent with most work, this paper removes sentence pairs with "-" and splits them into training/validation/test sets in a 549,367/9,842/9,824 ratio.

3.1.2 Model Parameters

A total of 34,877 different words were collected from the training/validation/test sets to form the vocabulary. Among them, 30,626 words can be found in the pre-trained word vector set 840B-GloVe. For OOV (out-of-vocabulary) words, random distribution vectors within [-0.05, 0.05] are used for initialization. Word vector length is fixed at 300. Since encoding vector length needs to be square-rootable, three groups were designed for experimental comparison: [16×16, 18×18, 20×20]. Note that although training word vectors simultaneously brings some performance improvement, considering the huge size of the word vector matrix and the resulting memory load, this experiment does not update the word vector matrix during training. Mini-batch training size is 128, with additional null tokens appended to sentences that are not long enough within a batch, corresponding to a 300-dimensional zero vector. Training epochs are set to 30, with early stopping when validation accuracy does not improve or even decreases for three consecutive iterations. The model with the highest validation accuracy is saved as the optimal model for test set prediction.

3.1.3 Hyperparameters

ADAM proposed by Kingma et al. is used as the gradient descent optimizer. The first momentum coefficient β_1 and second momentum coefficient are set to 0.9 and 0.999, respectively. The initial learning rate is 0.001. To accelerate

gradient descent, a decay rate of 0.95 is set for every 1000 training steps globally. Meanwhile, it is observed that end-to-end models tested on the SNLI dataset are particularly prone to overfitting, so dropout is introduced, randomly disabling 20% of neurons at the input and output ends of the encoding layer and the output end of the control layer.

3.2 Quantitative Analysis

To more comprehensively evaluate DIN' s performance, Table 2 lists a series of existing models for comparison. Para represents the number of training parameters used by the model besides word vectors, while Train and Test represent accuracy (%) on the training and test sets, respectively.

Two baseline models, HyperLSTM and rLSTM, were also implemented. The following findings emerge:

a) HyperLSTM uses a small LSTM based on the main LSTM to dynamically generate the main LSTM' s weights. Originally designed for single-sentence modeling, when placed in a sentence-pair scenario, it effectively requires training four LSTMs simultaneously. The increased number of training weights slows down training and leads to more severe overfitting. Moreover, the model has complex parameter settings, making it difficult to balance between state vector and word embedding vector dimensions. Although the model' s performance is not ideal (83.2%), its advantage over the LSTM encoder is still obvious, proving that dynamic weights are effective for RTE tasks.

b) rLSTM, as a typical strong interaction model with word-by-word matching similar to DIN, achieves a test set accuracy of 86.6% in our implementation, which has some discrepancy from the 87.5% reported in the original paper due to unavailability of certain important parameters. However, the performance improvement over word-by-word attention is still quite significant, proving that strong interactivity is effective for RTE tasks.

c) The performance of both DIN models proposed in this paper steadily improves as the interaction matrix size increases. The best performance appears with 20×20 DIN-1, surpassing the previous best model 300D rLSTM while using only half the training parameters.

d) Using the ablation method, the dynamic weight generation operation in the control layer of 20×20 DIN-1 is removed, and the interaction layer' s output matrix is directly converted into a vector as an additional input to the GRU encoder. It can be seen that the DIN model with only matrix interaction achieves 86.2% test accuracy, which, although lower than rLSTM, still surpasses most existing models in Table 2. On the other hand, the 1.8% lower test accuracy compared to the complete model further proves the effectiveness of dynamic weights.

e) Interestingly, although DIN-2 uses more training parameters than DIN-1

for dynamic weight generation, its test performance is consistently worse than the latter, contrary to our initial estimate in Section 3.6. The possible reason is that DIN-1 not only maintains matrix computation when generating dynamic weights (equations (15)(16)), but also matrixizes all GRU encoding steps (equations (9)-(14)), ensuring computational consistency with the interaction layer and demonstrating that matrices can carry more semantic information than vectors.

Experimental results prove that through the mutual promotion of matrixization and dynamic weights, the model can achieve higher test set recognition accuracy while maintaining a streamlined structure (with fewer training parameters than most existing models).

3.3 Qualitative Analysis

To more intuitively understand DIN' s interactivity, the interaction vector between H sentence word x_t and the P sentence is obtained as:

$$\mathbf{v}_t = \text{VECTORIZE}(\mathbf{W}_\tau t) \in \mathbb{R}^d \tag{28}$$

$$\alpha_\tau^t = \ell_2\text{-norm}\|\mathbf{v}_\tau^t\| \tag{29}$$

$$i_\tau^t = \frac{\alpha_\tau^t - \min\{\alpha_\tau^t\}}{\max\{\alpha_\tau^t\} - \min\{\alpha_\tau^t\}} \in [0,1] \tag{30}$$

$$\mathbf{I}_t = \{i_\tau^t\} \tag{31}$$

where $\mathbf{W}_\tau^t, \tau \in [1, \cdots, \ell_P]$ are the output matrices at each time step $t \in [1, \cdots, \ell_H]$ in the interaction layer, and each element i_τ^t in the interaction vector represents the matching degree between the current word x_t and P sentence word x'_τ.

Figure 2 [Figure 2: see original paper] shows three examples of the best model 20×20 DIN-1' s interactivity. Heat maps are used to display interaction vectors at each time step. The three examples are manually selected from the test set, sharing the same P sentence: "a guy in glasses is biting into a pink marshmallow chick while somebody else is puckering their lips out wanting a bite."

The first example (Figure 2 top) shows the model first correctly identifies two approximate word pairs (man, guy) and (eats, biting). If based solely on this, the sentence pair would be misjudged as entailment. But then the model discovers the obvious contradictory relationship between hamburger and marshmallow, causing the logical relationship to reverse and ultimately be judged as contradiction. The second example (Figure 2 middle) shows that although both P and H sentences have long text structures, most clearly identified word pairs only have approximate relationships, such as (desires, wanting), so the model judges them as entailment. The third example (Figure 2 bottom) is similar to the previous one with long text structures, but the H sentence contains modifiers like "his last" that don' t exist in P, character relationships like "his friend" that don' t exist in P, and an additional 转折句 "but he does not give in." This

extra information has no counterpart in P, making it impossible for the model to make a clear judgment, thus giving a neutral verdict.

On the other hand, the same method is used to display the dynamic changes of control layer weights at each time step. The three examples in Figure 3 correspond to the three H sentences in Figure 2. Red, blue, and green colors represent the update gate ($\mathbf{W}_z^t$), reset gate ($\mathbf{W}_r^t$), and candidate activation weights ($\mathbf{W}_h^t$) of the control layer GRU cell at each time step, with value ranges of [0,1]. Two observations are:

a) Words with relatively light color blocks across all three weights, such as "a,""while,""his"(Figure 3 top), "else"(Figure 3 middle), "as,""for,""does,""in" (Figure 3 bottom), do not produce important alignment information affecting the model' s judgment in Figure 2. This means these words cannot stimulate the model to modify weights to encode useless information, and the dynamic weight generation is dormant at these times, with the control layer focusing only on encoding the H sentence' s context information.

b) Although the three computed weights generated by the model at each time step fluctuate randomly without fixed patterns, if we look at the overall number of dark blocks across all time steps, the candidate activation weights change relatively frequently, indicating more obvious differences in candidate activation at each time step and playing a more primary role in controlling information flow between the two sentences.

4 Conclusion

To solve the textual entailment recognition problem, this paper proposes a dynamic interactive network DIN that combines matrixization and dynamic network weights during the word-by-word matching process between H and P sentences. Word vectors are projected into a two-dimensional matrix space for interaction, and the output matrices are used to dynamically generate computation weights for the GRU encoder. Experimental results on the SNLI dataset show that the optimal model proposed in this paper surpasses the previous best model while using only half the training parameters. Additionally, using the ablation method, the model with only matrix interaction (without dynamic network weights) still achieves recognition accuracy surpassing most interaction models, further proving the effectiveness of both methods individually. Strong interaction models like DIN generally outperform weak or non-interaction models, but such research is rare. Future work will continue to explore such models, with a feasible direction being to allow interactive information to flow back to the originally fixed P sentence, promoting true bidirectional information flow between the two sentences to improve recognition accuracy.

References

[1] Huo Huan, Zhang Wei, Liu Liang, et al. Collaborative filtering recommendation model based on convolutional denoising auto encoder [C]// Proc of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing. Chongqing China: ACM Digital Library, 2017: 64-71.

[2] Liang Liu, Huan Huo, Xiufeng Liu, et al. Recognizing textual entailment with attentive reading and writing operations [C]// Proc of the 23rd International Conference on Database Systems for Advanced Applications. Gold Coast Australia: Springer, 2018: 1-14.

[3] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling [EB/OL]. (2014-12-11) [2018-03-15]. https://arxiv.org/abs/1412.3555.

[4] Huo Huan, Zhang Wei, Liu Liang, et al. A hybrid neural network model on syntax tree structures [J]. Journal of Chinese Information Processing, 2017, 31 (06): 58-66.

[5] Rocktäschel Tim, Grefenstette Edward, Hermann Karl Moritz, et al. Reasoning about entailment with neural attention [EB/OL]. (2015-09-22) [2018-03-15]. https://arxiv.org/abs/1509.06664

[6] Shuohang Wang, Jing Jiang. Learning natural language inference with LSTM [C]// Proc of Conference of the North American Chapter of the Association for Computational Linguistics. Stroudsburg: ACL, 2016: 1442-1451.

[7] Lei Sha, Baobao Chang, Zhifang Sui, et al. Reading and Thinking: Re-read LSTM Unit for Textual Entailment Recognition [C]// Proc of International Conference on Computational Linguistics. Stroudsburg: ACL, 2016: 2870-2879.

[8] Biao Liu, Minlie Huang, Song Liu, et al. A Sentence Interaction Network for Modeling Dependence between Sentences [C]// Proc of Annual Meeting of the Association for Computational Linguistics. Stroudsburg: ACL, 2016: 2249-2255.

[9] Pengfei Liu, Xipeng Qiu, Jifan Chen, et al. Deep Fusion LSTMs for Text Semantic Matching [C]// Proc of Annual Meeting of the Association for Computational Linguistics. Stroudsburg: ACL, 2016: 1034-1043.

[10] Pengfei Liu, Xipeng Qiu, Yaqian Zhou, et al. Modelling Interaction of Sentence Pair with coupled-LSTMs [C]// Proc of Conference on Empirical Methods in Natural Language Processing. Stroudsburg: ACL, 2015: 1703-1712.

[11] Barker C. Continuations in natural language [EB/OL]. (2013-12-10). http://www.cs.bham.ac.uk/~hxt/cw04/barker.pdf.

[12] Ha David, Dai Andrew, Le Quoc V. Hyper networks [EB/OL]. (2016-09-27) [2018-03-15]. https://arxiv.org/abs/1609.09106v4.

[13] Samuel R. Bowman, Gabor Angeli, Christopher Potts, et al. A large annotated corpus for learning natural language inference [C]// Proc of Conference on Empirical Methods in Natural Language Processing. Stroudsburg: ACL, 2015: 632-642.

[14] Jason Weston, Sumit Chopra, Antoine Bordes. Memory networks [EB/OL]. (2014-10-15) [2018-03-15]. https://arxiv.org/abs/1410.3916v11.

[15] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, et al. End-to-end memory networks [C]// Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2015: 2440-2448.

[16] Cheng Jianpeng, Dong Li, Lapata Mirella. Long short-term memory-networks for machine reading [C]// Proc of Conference on Empirical Methods in Natural Language Processing. Stroudsburg: ACL, 2016: 551-561.

[17] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate [EB/OL]. (2014-09-01) [2018-03-15]. https://arxiv.org/abs/1409.0473.

[18] Lakoff G. Linguistics and natural logic [J]. Synthese, 1970, 22 (1): 151-271.

[19] MacCartney Bill. Natural language inference [D]. Stanford: Stanford University, 2009.

[20] Milne R, Strachey C. A theory of programming language semantics [M]. London: Chapman and Hall, 1977.

[21] Jeffrey Pennington, Richard Socher, Christopher D. Manning. Glove: Global vectors for word representation [C]// Proc of Conference on Empirical Methods in Natural Language Processing. Stroudsburg: ACL, 2014: 1532-1543.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, et al. Distributed representations of words and phrases and their compositionality [C]// Advances in Neural Information Processing Systems 26. Cambridge: MIT Press, 2013: 3111-3119.

[23] Diederik P. Kingma, Jimmy Ba. Adam: A method for stochastic optimization [EB/OL]. (2014-12-22) [2018-03-15]. https://arxiv.org/abs/1412.6980v9.

[24] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, et al. Dropout: a simple way to prevent neural networks from overfitting [J]. Journal of Machine Learning Research, 2014, 15 (1): 1929-1958.

[25] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, et al. A decomposable attention model for natural language inference [C]// Proc of Conference on Empirical Methods in Natural Language Processing. Stroudsburg: ACL, 2016: 2249-2255.

Note: Figure translations are in progress. See original paper for figures.