Co-Stack Residual Affinity Networks with Multi-level Attention Refinement for Matching Text Sequences

Yi Tay[†*], **Luu Anh Tuan**[ψ*], **Siu Cheung Hui**[φ]
[†φ]Nanyang Technological University, Singapore
[ψ]Institute for Infocomm Research, A*Star Singapore
ytay017@e.ntu.edu.sg[†], at.luu@i2r.a-star.edu.sg[ψ]
asschui@ntu.edu.sg[φ]

Abstract

Learning a matching function between two text sequences is a long standing problem in NLP research. This task enables many potential applications such as question answering and paraphrase identification. This paper proposes Co-Stack Residual Affinity Networks (CSRAN), a new and universal neural architecture for this problem. CSRAN is a deep architecture, involving stacked (multi-layered) recurrent encoders. Stacked/Deep architectures are traditionally difficult to train, due to the inherent weaknesses such as difficulty with feature propagation and vanishing gradients. CSRAN incorporates two novel components to take advantage of the stacked architecture. Firstly, it introduces a new bidirectional alignment mechanism that learns affinity weights by fusing sequence pairs across stacked hierarchies. Secondly, it leverages a multi-level attention refinement component between stacked recurrent layers. The key intuition is that, by leveraging information across all network hierarchies, we can not only improve gradient flow but also improve overall performance. We conduct extensive experiments on six well-studied text sequence matching datasets, achieving state-of-the-art performance on all.

1 Introduction

Determining the semantic affinity between two text sequences is a long standing research problem in natural language processing research. This is understandable, given that technical innovations in this domain would naturally bring benefits to a diverse plethora of applications ranging from paraphrase detection to standard document retrieval. This work focuses on short textual sequences, focusing on a myriad of applications such as natural language inference, question answering, reply prediction and paraphrase detection. This paper presents a new deep matching model for universal text matching.

Neural networks are dominant state-of-the-art approaches for many of these matching problems (Gong et al., 2017; Shen et al., 2017; Wang et al., 2017; Chen et al., 2017). Fundamentally, neural networks operate via a concept of feature hierarchy, in which hierarchical representations are constructed as sequences propagate across the network. In the context of matching, representations are often (1) encoded, (2) matched, and then (3) aggregated for prediction. Each key step often comprises several layers, which consequently adds to the overall depth of the network.

Unfortunately, it is a well established fact that deep networks are difficult to train. This is attributed to not only vanishing/exploding gradients but also an instrinsic difficulty pertaining to feature propagation. To this end, commonly adopted solutions include Residual connections (He et al., 2016) and/or Highway layers (Srivastava et al., 2015). The key idea in these approaches is to introduce additional (skip/residual) connections, propagating shallower layers to deeper layers via shortcuts. To the best of our knowledge, these techniques are generally applied to single sequences and therefore the notion of pairwise residual connections have not been explored.

This paper presents Co-Stack Residual Affinity Networks (CSRAN), a stacked multi-layered recurrent architecture for general purpose text matching. Our model proposes a new co-stacking mechanism that computes bidirectional affinity scores by leveraging all feature hierarchies between text sequence pairs. More concretely, word-by-word affinity scores are not computed just from the final encoded representations but across all the entire feature hierarchy.

There are several benefits to our co-stacking

[*]Denotes equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4492–4502
Brussels, Belgium, October 31 - November 4, 2018.

mechanism. Firstly, co-stacking acts as a form of residual connector, alleviating the instrinsic issues with network depth. Secondly, there are more extensive matching interfaces between text sequences as the affinity matrix is not computed by just one representation but multiple representations instead. Naturally, increasing the opportunities for interactions between sequences is an intuitive method for improving performance.

Additionally, our model incorporates a Multi-level Attention Refinement (MAR) architecture in order to fully leverage the stacked recurrent architecture. The MAR architecture is a multi-layered adaptation and extension of the CAFE model (Tay et al., 2017c), in which attention is computed, compressed and then re-fed into the input sequence. In our approach, we use CAFE blocks to repeatedly *refine* representations at each level of the stacked recurrent encoder.

The overall outcome of the above-mentioned architectural synergies is a highly competitive model that establishes state-of-the-art performance on six well-known text matching datasets such as SNLI and TrecQA. The overall contributions of this work are summarized as follows:

- We propose a new deep stacked recurrent architecture for matching text sequences. Our model is based on a new co-stacking mechanism which learns to align by exploiting matching across feature hierarchies. This can be interpreted as a new way to incorporate shortcut connections within neural models for sequence matching. Additionally, we also propose a multi-level attention refinement scheme to leverage our stacked recurrent model.

- While stacked architectures can potentially lead to considerable improvements in performance, our experiments show that in the absence of our proposed CSRA (Co-stack Residual Affinity) mechanism, stacking may conversely lead to performance degradation. As such, this demonstrates that our proposed techniques are essential for harnessing the potential of deep architectures.

- We conduct extensive experiments on four text matching tasks across six well-studied datasets, i.e., *Natural Language Inference* (SNLI (Bowman et al., 2015), SciTail (Khot et al., 2018)), *Paraphrase Identification* (Quora, TwitterURL (Lan et al., 2017)), *Answer Sentence Selection* (Wang et al., 2007) and *Utterance-Response Matching* (Ubuntu (Lowe et al., 2015)). Our model achieves state-of-the-art performance on all datasets.

2 Co-Stack Residual Affinity Networks

In this section, we introduce our proposed model architecture for general/universal text matching. The key idea of this architecture is to leverage deep stacked layers, while mitigating the inherent weaknesses of going deep. As such, our network is in similar spirit to highway networks, residual networks and DenseNets, albeit tailored specifically for pairwise architectures. Figure 1 illustrates a high-level overview of our proposed model architecture.

2.1 Input Encoder

The inputs to our model are standard sequences of words A and B which represent sequence a and sequence b respectively. In the context of different applications, a and b take different roles such as premise/hypothesis or question/answer. Both sequences are converted into word representations (via pretrained word embeddings) and character-based representations. Character embeddings are trainable parameters and a final character-based word representation of d dimensions is learned by passing all characters into a Bidirectional LSTM encoder. This is standard, following many works such as (Wang et al., 2017). Word embeddings and character-based word representations are then concatenated to form the final word representation. Next, the word representation is passed through a (optional and tuned as a hyperparameter) 2-layered highway network of d dimensions.

2.2 Stacked Recurrent Encoders

Next, word representations are passed into a stacked recurrent encoder layer. Specifically, we use Bidirectional LSTM encoders at this layer. Let k be the number of layers of the stacked recurrent encoder layer.

$$h_t^i = \text{BiLSTM}_i(h_{t-1}) \ \ \forall t \in [1, 2 \cdots \ell] \quad (1)$$

where BiLSTM_i represents the i-th BiLSTM layer and h_t^i represents the t-th hidden state of the i-th BiLSTM layer. ℓ is the sequence length. Note that the parameters are shared for both a and b.

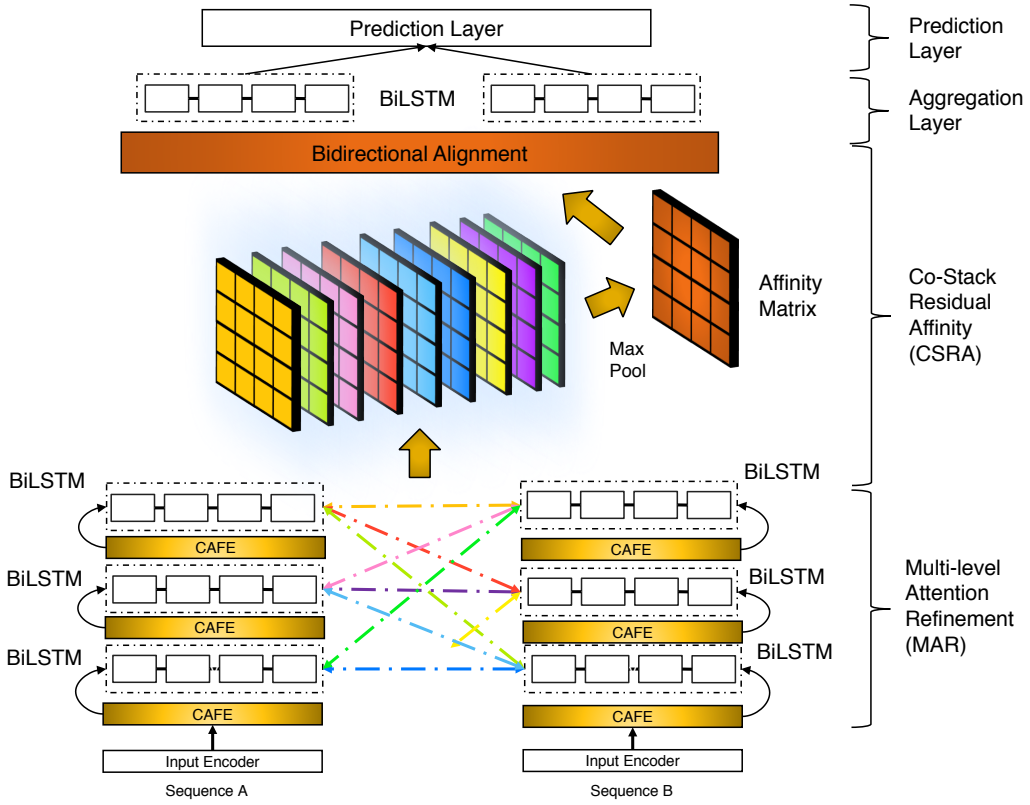

Figure 1: Illustration of the proposed Co-Stack Residual Affinity Network (CSRAN) architecture. Each color coded matrix represents the interactions between two layers of sequence A and sequence B. (Best viewed in color)

2.3 Multi-level Attention Refinement (MAR)

Inspired by CAFE (Tay et al., 2017c) (Compare-Align-Factorized Encoders), a top performing model on the SNLI benchmark, we utilize CAFE blocks between the BiLSTM layers. Each CAFE block returns *six* features, which are generated by a factorization operation using factorization machines (FM). While the authors in (Tay et al., 2017c) simply use this operation in a single layer, we utilize this in a multi-layered fashion which we found to have worked well. This constitutes our multi-level attention refinement mechanism. More concretely, we apply the CAFE operation to the outputs of each BiLSTM layer, allowing the next BiLSTM layer to process the 'augmented' representations. The next layer retains its dimensionality by projecting the augmented representation back to its original size using the BiLSTM encoder. This can be interpreted as repeatedly refining representations via attention. As such, adding CAFE blocks is a very natural fit to the stacked recurrent architecture.

CAFE Blocks This section describes the operation of each CAFE block. The key idea behind CAFE blocks is to align a and b, and compress alignment vectors such as $b' - a$ (subtraction), $b' \odot a$ (element-wise multiplication) and $[b'; a]$ (concatenation) into scalar features. These scalar features are concatenated to the original input embedding, which can be pased into another BiLSTM layer for refining representations. Firstly, a, b are modeled aligned via $E_{ij} = F(a)^\top F(b)$ and then aligned via:

$$A' = E^\top B \text{ and } B' = AE^\top \tag{2}$$

Given aligned pairs (A', B) and (B', A), we generate *three* matching vectors for the concatenation $([a'_i; b_i])$, element-wise multiplication $(a'_i \odot b_i)$ and subtraction vectors $(a'_i - b_i)$ of each pair. After which, we apply a factorization machine (Rendle, 2010) $M(x)$ on each matching vector.

$$M(x) = w_0 + \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle\, x_i\, x_j \tag{3}$$

where $v \in \mathbb{R}^{d \times k}$, $w_0 \in \mathbb{R}$ and $w_i \in \mathbb{R}^d$. The output $M(x)$ is a scalar. Intuitively, this layer tries

to learn pairwise interactions between every x_i and x_j using factorized (vector) parameters v.

Factorization machines model low-rank structure within the matching vector, producing a scalar feature. This enables efficient propagation of these matching features to the next layer. The output of each CAFE block is the original input to the CAFE module, augmented with the output of the factorization machines. As such, if the input sequence is of d dimensions, then the output is $d + 3$ dimensions. Additionally, intra-attention is applied in similar fashion as above to generate *three* more features for each sequence. As a result, the output dimensions for each word becomes $d + 6$.

2.4 Co-Stack Residual Affinity (CSRA)

This layer is the cornerstone of our proposed approach and is represented as the middle segment of Figure 1 (the colorful matrices).

Co-Stacking Co-stacking refers to the fusion of a and b across multiple hierarchies. Recall that the affinity score between two words is typically computed by $s_{ij} = a^\top b$. We extend this to a residual formulation. More concretely, the affinity score between both words is now computed as the *maximum* influence it has over all layers.

$$s_{ij} = \max \sum_p \sum_q a_{pi}^\top b_{qj} \tag{4}$$

where a_{pi} is the i-th word for the p-th stacked layer for a and b_{qj} is the j-th word for the q-th stacked layer for b. The choice of the maximum operator is intuitive and is strongly motivated by the fact that we would like to give a high affinity for each word pair that shows a strong match at any of different hierarchical stages of learning representations. Note that this layer can be interpreted as constructing a matching tensor based on multi-hierarchical information and selecting the most informative match across all representation hierarchies.

Bidirectional Alignment In order to learn (bidirectionally) attentive representations, we first concatenate all stacked outputs to form a $\ell \times kd$ vector. Next, we apply the following operations to $A \in \mathbb{R}^{\ell_a \times kd}$ and $B \in \mathbb{R}^{\ell_b \times kd}$.

$$\bar{A} = S^\top B \text{ and } \bar{B} = AS^\top \tag{5}$$

where $\bar{A} \in \mathbb{R}^{\ell_b \times kd}$, $\bar{B} \in \mathbb{R}^{\ell_a \times kd}$ are the attentive (aligned) representations.

2.4.1 Matching and Aggregation Layer

Next, we match the attentive (aligned) representations using the subtraction, element-wise multiplication and concatenation of each aligned word. Subsequently, we pass this matching vector into a k layered BiLSTM layer.

$$a'_i = \text{BiLSTM}_k([\bar{b}_i - a_i, \bar{b}_i \odot a_i, \bar{b}_i, a_i]) \tag{6}$$
$$b'_i = \text{BiLSTM}_k([\bar{a}_i - b_i, \bar{a}_i \odot b_i, \bar{a}_i, b_i]) \tag{7}$$

The final feature representation is learned via the summation across the temporal dimension as follows:

$$z = [\sum_{i=1}^{\ell_a} a'_i \; ; \; \sum_{i=1}^{\ell_b} b'_i] \tag{8}$$

where $[.;.]$ is the concatenation operator.

2.5 Output and Prediction Layer

Our model predicts using the feature vector z for every given sequence pair. At this layer, we utilize standard fully connected layers. The number of output layers is typically 2-3 and is a tuned hyperparameter. Softmax is applied onto the final layer. The final layer is application specific, e.g., k classes for classification tasks and a two-class softmax for pointwise ranking. For all datasets, we optimize the cross entropy loss.

3 Experimental Evaluation

In this section, we introduce our experimental setup, baselines and results.

3.1 Datasets and Competitor Baselines

We use six public benchmark datasets for evaluating our proposed approach. This section briefly introduces each dataset, along with several state-of-the-art approaches that we compare against. Table 1 provides a summary of the datasets used in our experiments.

Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) is a well-known dataset for *entailment classification* (or natural language inference). The task is to determine if two sequences entail/contradict or are neutral to each other. This task is a three-way classification problem. On this dataset, we compare with several state-of-the-art models such as BiMPM (Wang et al., 2017), ESIM (Chen et al., 2017), DIIN (Gong et al., 2017), DR-BiLSTM (Ghaeini et al., 2018) and CAFE (Tay et al., 2017c).

Dataset	Task	$\|C\|$	Pairs
SNLI	Premise-Hypothesis	3	570K
Scitail	Premise-Hypothesis	2	27K
Quora	Question-Question	2	400K
Twitter	Tweet-Tweet	2	51K
TrecQA	Question-Answer	R	56K
Ubuntu	Utterance-Response	R	1M

Table 1: Statistics of datasets used in our experiment. $|C|$ denotes the number of classes and R denotes a ranking formulation. Twitter stands for the TwitterURL dataset.

Science Entailment (SciTail) (Khot et al., 2018) is a new entailment classification dataset that was constructed from science questions and answers. This dataset involves two-way classification (entail or non-entail). We compare with DecompAtt (Parikh et al., 2016), ESIM, DGEM (Khot et al., 2018) and CAFE.

Quora Duplicate Detection is a well-studied paraphrase identification dataset[1]. We use the splits provided by (Wang et al., 2017). The task is to determine if two questions are paraphrases of each other. This task is formulated as a binary classication problem. We compare with L.D.C (Wang et al., 2016b), BiMPM, the DecompAtt implementation by (Tomar et al., 2017) (word and char level) and DIIN.

TwitterURL (Lan et al., 2017) is another dataset for paraphrase identification. It was constructed using Tweets referring to news articles. This task is also a binary classification problem. We compare with (1) MultiP (Xu et al., 2014), a strong baseline, (2) the implementation of (He and Lin, 2016) by (Lan et al., 2017) and (3) the Subword + LM model from (Lan and Xu, 2018).

TrecQA (Wang et al., 2007) is a well-studied dataset for *answer sentence selection task* (or question-answer matching). The goal is to rank answers given a question. This task is formulated as a pointwise learning-to-rank problem. Baselines include HyperQA (Tay et al., 2017a), Ranking-based Multi-Perspective CNN (He et al., 2015) implementation by (Rao et al., 2016), BiMPM, the compare-aggregate (Wang and Jiang, 2016a) model extension by (Bian et al., 2017) (we denote this model as CA), IWAN (Shen et al., 2017) and the recent MCAN model, i.e., Multi-Cast Attention Networks (Tay et al., 2018c). A leaderboard is maintained at https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art).

Ubuntu (Lowe et al., 2015) is a dataset for *Utterance-Response Matching* and comprises 1-million utterance-response pairs. This dataset is based on the Ubuntu dialogue corpus. The goal is to predict the response to a message. We use the same setup as (Wu et al., 2016). Baselines include CNTN (Qiu and Huang, 2015), APLSTM (dos Santos et al., 2016), MV-LSTM (Wan et al., 2016a) and KEHNN (Wu et al., 2016). Results are reported from (Wu et al., 2016).

Metrics For all datasets, we follow the evaluation procedure from all the original papers. The metric for SNLI, SciTail and Quora is the accuracy metric. The metric for the TwitterURL dataset is the F1 score. The metric for TrecQA is the Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) metric. The metric for Ubuntu is the Recall@K for $k = 1, 2, 5$ (given 9 negative samples) and the binary classification accuracy score.

3.2 Experimental Setup

All baselines are reported from the respective papers. All models are trained with the Adam optimizer (Kingma and Ba, 2014) with learning rates tuned amongst $\{0.001, 0.0003, 0.0004\}$. Batch size is tuned amongst $\{32, 64, 128, 256\}$. The dimensions of the BiLSTM encoders are tuned amongst $\{64, 100, 200, 300\}$ and the number of hidden dimensions of the prediction layers are tuned amongst $\{100, 200, 300, 600\}$. The number of stacked recurrent layers is tuned from $[2, 5]$ and the number of aggregation BiLSTM layers is tuned amongst $\{1, 2\}$. The number of prediction layers is tuned from [1,3]. Parameters are initialized using glorot uniform (Glorot and Bengio, 2010). All unspecified activation functions are ReLU activations. Word embeddings are initialized with GloVe (Pennington et al., 2014) and fixed during training. We implement our model in Tensorflow (Abadi et al., 2015) and use the CUDNN implementation for all BiLSTM layers.

3.3 Experimental Results

Overall, our proposed CSRAN architecture achieves state-of-the-art performance on all six well-established datasets.

[1] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

On SNLI (Table 2), CSRAN achieves the best[2] single model performance to date on the well-established dataset. This demonstrates the effectiveness of CSRAN, taking into consideration of the inherent competitiveness of this well-known benchmark. On SciTail (Table 3), CSRAN similarly achieves the best performance to date on this dataset, outperforming the existing CAFE model by $+3.4\%$ absolute accuracy.

On Quora (Table 4), CSRAN also achieves the best single model score, outperforming strong baselines such as BiMPM ($+1.1\%$) and DIIN ($+0.2\%$). Moreover, there is also considerable performance improvement on the TwitterURL dataset (Table 5) as CSRAN outperforms the existing state-of-the-art Subword + LM model ($+8\%$) and Deep Pairwise Word ($+9.1\%$).

On TrecQA (Table 6), CSRAN achieves the best performance on this dataset. CSRAN outperforms the existing state-of-the-art model, IWAN ($+3.2\%/ + 4.6\%$). CSRAN also outperforms strong competitive baselines such as BiMPM ($+5.2\%/+3.6\%$) and MPCNN ($+5.3\%/+5.8\%$). Finally, on Ubuntu (Table 7), CSRAN also outperforms many competitive models such as CNTN, APLSTM and KEHNN. Performance improvement over all metrics are $\approx 9\% - 10\%$ compared to the existing state-of-the-art.

Overall, CSRAN achieves state-of-the-art performance on six well-studied datasets. On several datasets, our achieved performance is not only the highest reported score but also outperforms the existing state-of-the-art models by a considerable margin.

Model	Acc
BiMPM (Wang et al., 2017)	87.5
ESIM (Chen et al., 2017)	88.0
DIIN (Gong et al., 2017)	88.0
DR-BiLSTM (Ghaeini et al., 2018)	88.5
CAFE (Tay et al., 2017c)	88.5
CSRAN	**88.7**

Table 2: Experimental results on single model SNLI dataset.

[2]For fair comparison, we do not compare with (1) models that use external contextualized word embeddings, e.g., CoVe (McCann et al., 2017) / ELMo (Peters et al., 2018) / generative pretraining (Radford et al.) and (2) ensemble systems. As either (1) and/or (2) would also intuitively boost the performance of the base CSRAN model.

Model	Acc
DecompAtt (Parikh et al., 2016)	72.3
ESIM (Chen et al., 2017)	70.6
DGEM (Khot et al., 2018)	77.3
CAFE (Tay et al., 2017c)	83.3
CSRAN	**86.7**

Table 3: Experimental results on SciTail dataset.

Model	Acc
L.D.C (Wang et al., 2016b)	87.5
Word DecompAtt (Tomar et al., 2017)	87.5
BiMPM (Wang et al., 2017)	88.1
Char DecompAtt (Tomar et al., 2017)	88.4
DIIN (Gong et al., 2017)	89.0
CSRAN	**89.2**

Table 4: Experimental results on Quora Duplicate Detection dataset.

Model	F1
MultiP (Xu et al., 2014)	0.536
DeepPairwiseWord (He and Lin, 2016)	0.749
Subword + LM (Lan and Xu, 2018)	0.760
CSRAN	**0.840**

Table 5: Experimental results on TwitterURL paraphrase dataset.

Model	MAP/MRR
HyperQA (Tay et al., 2017a)	0.784/0.865
MPCNN (Rao et al., 2016)	0.801/0.877
BiMPM (Wang et al., 2017)	0.802/0.899
CA (Bian et al., 2017)	0.821/0.899
IWAN (Shen et al., 2017)	0.822/0.889
MCAN (Tay et al., 2018c)	0.838/0.904
CSRAN	**0.854/0.935**

Table 6: Experimental results on TrecQA dataset.

Model	Acc	R@1	R@2	R@5
CNTN	0.743	0.349	0.512	0.797
LSTM	0.725	0.361	0.494	0.801
APLSTM	0.758	0.381	0.545	0.801
MV-LSTM	0.767	0.410	0.565	0.800
KEHNN	0.786	0.460	0.591	0.819
MCAN	0.834	0.551	0.684	0.875
CSRAN	**0.839**	**0.556**	**0.692**	**0.880**

Table 7: Experimental results on the Ubuntu dataset for utterance-response matching. Baseline results are reported from (Wu et al., 2016).

3.3.1 Training Efficency

With many BiLSTM layers, it is natural to be skeptical about the training efficiency of our model. However, since we use the CUDNN implementation of the BiLSTM model, the runtime is actually very manageable. On SNLI, with a batch size of 128, our model with 3 stacked recurrent layers and 2 aggregation BiLSTM layers runs at ≈17 minutes per epoch and converges in less than 20 epochs. On SciTail, our model runs at ≈ 2 minutes per epoch with a batch size of 32. This is benchmarked on a TitanXP GPU. While our model is targetted at performance and not efficiency, this section serves as a reassurance that our model is not computationally prohibitive.

3.4 Ablation Study

In order to study the effectiveness of the key components in our proposed architecture, we conduct an extensive ablation study. Table 8 reports the results on several ablation baselines. There are three key ablation baselines as follows: (1) we removed MAR from the stacked recurrent network, (2) we removed CSRA from the network and finally (3) we removed both MAR and CSRA from the network. All ablation baselines reported are stacked with 3 layers. Firstly, we observe that both

Ablation	SNLI	SciTail	TrecQA
Original	88.6	88.0	0.86/0.90
w/o MAR	88.4	82.5	0.79/0.85
w/o CSRA	88.1	86.2	0.84/0.89
w/o MAR/CSRA	88.0	83.0	0.79/0.84

Table 8: Ablation study (development score) of our key model components on three datasets.

MAR and CSRA are critical components in our model, i.e., removing any of them would result in a drop in performance. Secondly, we observe that the relative utility of CSRA and MAR depends on the dataset. Removing MAR sigificantly reduces performance on SciTail and TrecQA. On the other hand, removing CSRA degrades the performance more than MAR on SNLI. Finally, it is good to note that, while performance degradation on SNLI development set may not seem significant, the *w/o MAR and CSRA* ablation performance baseline achieved only 87.7% accuracy on the test set, compared to 88.7% of the original model. This is equivalent to dropping from state-of-the-art to the 5th ranked model. Overall, we are able to conclude that the CSRA and MAR make meaningful improvements to our model architecture.

3.5 Effect of Stack Depth

In this section, our goals are twofold - (1) studying the effect of stack depth on model performance and (2) determining if the proposed CSRAN model indeed helps with enabling deeper stack depths. In order to do so, we compute the development set performance of two models. The first is the full CSRAN architecture and the second is a baseline stacked model architecture. Note that the bidirectional alignment layer and remainder of the model architecture (highway layers, etc.) remain completely identical to CSRAN to make this study as fair as possible.

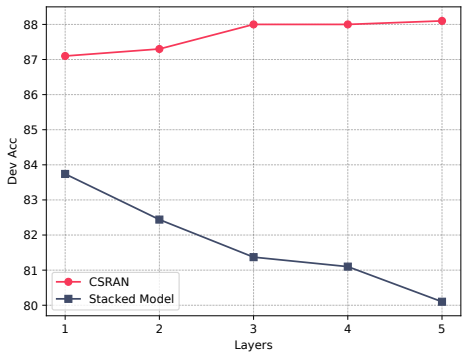

Figure 2: Relative effect of stack depth on CSRAN and the baseline Stacked Model on SciTail dataset.

Figure 2 illustrates the model performance with varying stack depth. As expected, the performance of the stacked model declines when increasing the stack depth. On the other hand, the performance of CSRAN improves by adding additional layers. The largest gain is when jumping from 2 layers to 3 layers. The subsequent performance improvement from 3-5 layers is marginal. From this study, the takeaway is that standard stacked architectures are insufficient. As such, our proposed CSRA mechanism can aid in enabling deeper models which can result in stronger model performance[3].

Next, we study the general effect of stack depth (number of layers) on model performance. Figure 3 reports the model performance (dev accuracy) of our CSRAN architecture on Quora and

[3]The best result on Scitail was obtained with 5 layers. Moreover, the difference in test performance between stacked and single-layered model was considerably high (+2.5%) even though dev performance increased by +1%.

SNLI datasets. We observe that a stacked architecture with 3 layers is significantly better than a single-layered architecture. The optimal development score is 3-4 layers for SNLI and 3 layers for Quora. However, we observe the performance of Quora declines after 3 layers (notably it is still higher than an unstacked model). However, the performance on SNLI remains relatively stable.

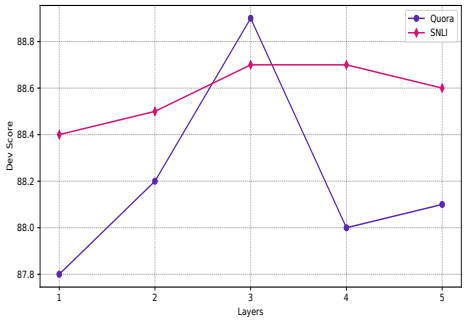

Figure 3: Effect of stack depth on CSRAN performance on Quora and SNLI datasets.

4 Related Work

Learning to matching text sequences is a core and fundamental research problem in NLP and Information Retrieval. A wide range of NLP applications fall under this paradigm such as natural language inference (Bowman et al., 2015; Khot et al., 2018), paraphrase identification (Lan and Xu, 2018), question answering (Severyn and Moschitti, 2015), document search (Shen et al., 2014; Hui et al., 2017), social media search (Rao et al., 2018) and entity linking (Phan et al., 2017). As such, universal text matching algorithms are generally very attractive, in lieu of the prospects of potentially benefitting an entire suite of NLP applications.

Neural networks have been the prominent choice for text matching. Earlier works are mainly concerned with learning a matching function between RNN/CNN encoded representations (Severyn and Moschitti, 2015; Yu et al., 2014; Qiu and Huang, 2015; Tay et al., 2017b, 2018b). Models such as Recursive Neural Networks have also been explored (Wan et al., 2016b). Subsequently, attention-based models were adopted (Rocktäschel et al., 2015; Wang et al., 2016a; Parikh et al., 2016), demonstrating superior performance relative to their non-attentive counterparts.

Today, the dominant state-of-the-art approaches for text matching are mostly based on neural models configured with bidirectional attention layers (Shen et al., 2017; Tay et al., 2017c). Bidirectional attention comes in various flavours which can be known as soft alignment (Shen et al., 2017; Chen et al., 2017), decomposable attention (Parikh et al., 2016), attentive pooling (dos Santos et al., 2016) and even complex-valued attention (Tay et al., 2018a). The key idea is to jointly soft align text sequences such that they can be compared at the index level. To this end, various comparison functions have been utilized, ranging from feed-forward neural networks (Parikh et al., 2016) to factorization machines (Tay et al., 2017c). Notably, these attention (and bi-attention) mechanisms are also widely adopted (or originated) from many related sub-fields of NLP such as machine translation (Bahdanau et al., 2014) and reading comprehension (Xiong et al., 2016; Seo et al., 2016; Wang and Jiang, 2016b).

Many text matching neural models are heavily grounded in the compare-aggregate architecture (Wang and Jiang, 2016a). In these models, matching and comparisons occur between text sequences, aggregating features for making the final prediction. Recent state-of-the-art models such as BiMPM (Wang et al., 2017) and DIIN (Gong et al., 2017) are representative of such architectural paradigm, utilizing an attention-based matching scheme and then a CNN or LSTM-based feature aggregator. Earlier works (Wan et al., 2016a; He et al., 2015; He and Lin, 2016) exploit a similar paradigm, albeit without the usage of attention.

Across many NLP and machine learning applications, utilizing stacked architectures is a common way to enhance representation capability of the encoder (Sutskever et al., 2014; Graves et al., 2013; Zhang et al., 2016; Nie and Bansal, 2017), leading to performance improvement. Deep networks suffer from inherent difficulty in feature propagation and/or vanishing/exploding gradients. As a result, residual strategies have often been employed (He et al., 2016; Srivastava et al., 2015; Huang et al., 2017). However, to the best of our knowledge, this work presents a new way of residual connections, leveraging on the fact that pairwise formulation of the text matching task.

5 Conclusion

We proposed a deep stacked recurrent architecture for general-purpose text sequence matching. We proposed a new co-stack residual affin-

ity mechanism for matching sequence pairs, leveraging multi-hierarchical information for learning bidirectional alignments. Our proposed CSRAN model achieves state-of-the-art performance across six well-studied benchmark datasets and four different problem domains.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

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

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1987–1990, New York, NY, USA. ACM.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1657–1668.

Reza Ghaeini, Sadid A Hasan, Vivek Datla, Joey Liu, Kathy Lee, Ashequl Qadir, Yuan Ling, Aaditya Prakash, Xiaoli Z Fern, and Oladimeji Farri. 2018. Dr-bilstm: Dependent reading bidirectional lstm for natural language inference. *arXiv preprint arXiv:1802.05577*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *CoRR*, abs/1709.04348.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Hua He, Kevin Gimpel, and Jimmy J. Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1576–1586.

Hua He and Jimmy J. Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 937–948.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. Pacrr: A position-aware neural ir model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. *arXiv preprint arXiv:1708.00391*.

Wuwei Lan and Wei Xu. 2018. The importance of subword embeddings in sentence pair modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 41–45.

Ankur P. 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, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Minh C Phan, Aixin Sun, Yi Tay, Jialong Han, and Chenliang Li. 2017. Neupl: Attention-based semantic matching and pair-linking for entity disambiguation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1667–1676. ACM.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1305–1311.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Jinfeng Rao, Hua He, and Jimmy J. Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 1913–1916.

Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. 2018. Multi-perspective relevance matching with hierarchical convnets for social media search. *arXiv preprint arXiv:1805.08159*.

Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 373–382.

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

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. ACM.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017a. Enabling efficient question answer retrieval via hyperbolic neural networks. *CoRR*, abs/1707.07847.

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

Yi Tay, Minh C. Phan, Anh Tuan Luu, and Siu Cheung Hui. 2017b. Learning to rank question answer pairs with holographic dual LSTM architecture. In *Proceedings of the 40th International ACM SIGIR*

Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017, pages 695–704.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017c. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018b. Cross temporal recurrent networks for ranking question answer pairs. In *Proceedings of AAAI 2018*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018c. Multi-cast attention networks. In *Proceedings of KDD 2018*, KDD '18, pages 2299–2308, New York, NY, USA. ACM.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural paraphrase identification of questions with noisy pretraining. *arXiv preprint arXiv:1704.04565*.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016a. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2835–2841.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016b. Match-srnn: Modeling the recursive matching structure with spatial rnn. *arXiv preprint arXiv:1604.04378*.

Bingning Wang, Kang Liu, and Jun Zhao. 2016a. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? A quasi-synchronous grammar for QA. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 22–32.

Shuohang Wang and Jing Jiang. 2016a. A compare-aggregate model for matching text sequences. *CoRR*, abs/1611.01747.

Shuohang Wang and Jing Jiang. 2016b. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*.

Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2016. Knowledge enhanced hybrid neural network for text matching. *arXiv preprint arXiv:1611.04684*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. 2016. Highway long short-term memory rnns for distant speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5755–5759. IEEE.