Generalize Sentence Representation with Self-Inference

Kai-Chou Yang,[1] **Hung-Yu Kao**[2]
National Cheng Kung University
Tainan, Taiwan
zake7749@gmail.com,[1] hykao@mail.ncku.edu.tw[2]

Abstract

In this paper, we propose Self Inference Neural Network (SINN), a simple yet efficient sentence encoder which leverages knowledge from recurrent and convolutional neural networks. SINN gathers semantic evidence in an interaction space which is subsequently fused by a shared vector gate to determine the most relevant mixture of contextual information. We evaluate the proposed method on four benchmarks among three NLP tasks. Experimental results demonstrate that our model sets a new state-of-the-art on MultiNLI, Scitail and is competitive on the remaining two datasets over all sentence encoding methods. The encoding and inference process in our model is highly interpretable. Through visualizations of the fusion component, we open the black box of our network and explore the applicability of the base encoding methods case by case.

Introduction

Deep neural networks have achieved remarkable success in sentence representation learning. Based on sequential models like recurrent neural networks (RNN), convolutional neural networks (CNN) or self-attention, we can build a sentence encoder that transforms a sentence into a meaningful context vector for use in various downstream tasks.

Recently, the prevalent sentence encoders are usually based on a single type of encoding method: either RNN, CNN or self-attention. However, we argue that in order to obtain general contextual representations, a sentence encoder should simultaneously use multiple encoding methods. Considering the nature of language, the meaning of a word can depend on what has been stated as well as the company it keeps. Similarly, given a variety of encoding hypotheses, a sentence can be interpreted with multiple semantic readings each of which is not contradictory but complementary to the others. Therefore, instead of using a single encoding method arbitrarily, a better strategy would be to integrate semantic evidence from different perspectives and control the information flow according to context dynamically.

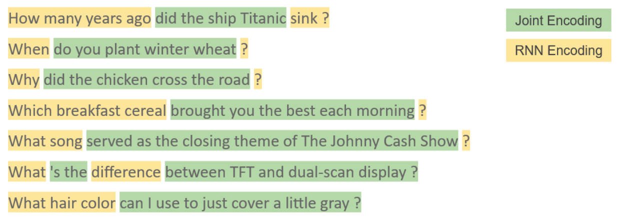
How many years ago did the ship Titanic sink ?
When do you plant winter wheat ?
Why did the chicken cross the road ?
Which breakfast cereal brought you the best each morning ?
What song served as the closing theme of The Johnny Cash Show ?
What 's the difference between TFT and dual-scan display ?
What hair color can I use to just cover a little gray ?
Joint Encoding
RNN Encoding

Figure 1: The reading behavior of our model. The RNN encoder catches the topic words alone and cooperates with the CNN encoder for the detailed description.

To this end, we introduce SINN, a novel neural architecture which leverages the information from various encoding methods through self-inference. Our model is composed of an RNN and CNN as two base encoders which generate contextual representations individually. Inspired by the common strategy for natural language inference (NLI), we propose a self-inference component based on heuristic interaction to learn the view of each base encoder and furthermore to create more generalized sentence representations by striking a balance between them. Figure 1 illustrates the reading behavior of our model. The RNN encoder would catch the topic words alone while it tends to cooperate with the CNN encoder when the semantic readings become more complicated.

To show the generalization ability, We evaluate the proposed method on four benchmarks among three different NLP tasks: natural language inference, text classification and sentiment classification. The experimental results demonstrate that our model sets a new state-of-the-art on two benchmarks while showing strong performance on the others, which indicates that through self-inference, the shallow base encoders can still outperform many deep and complicated architectures.

In addition, the process of self-inference is highly interpretable. By visualizing the importance weight of each contextual representation, we found that in our model, the base encoders adopt a collaborative strategy, each of which captures different and complementary semantic features. We conduct extensive analysis on the fusion gate to explain the reading behavior of our model as well as the applicability of our base encoders for various tasks.

Background

Sentence Encoder

The main purpose of a sentence encoder is to represent a variable length sentence into a fixed-size context vector which contains the semantic information of the original sentence. To achieve this goal, a sentence encoder usually consists of two sub-modules: an encoding method which captures the dependencies between words to build the contextual representations and a pooling method which summarizes the encoded sequences into a single vector. In this section, we will introduce the encoding methods which can be divided into three general categories: RNNs, CNNs and self-attention, depending on how to aggregate the dependencies between words.

Because of their superior ability to capture long-term dependencies, encoding methods based on RNNs are widely used in various NLP tasks. To build the contexts, RNNs process a sentence word by word and aggregate the current word with the hidden state which stores the semantic information of previous words. There are several RNN variants such as GRU (Chung et al. 2014) which improves sequence modeling in different ways, but in general they all have the same reading behavior as the vanilla RNN. Currently, one of the most popular embeddings is based on the RNN (Peters et al. 2018). They used a three layer BiLSTM as a language model to learn deep contextualized word representations. According to their observations, each recurrent layer in the language model encodes different types of linguistic knowledge, which indicates that RNNs are efficient to understand a language system.

Unlike RNNs which capture sequential dependencies, CNNs model sentence representations with local dependencies. CNNs have proven effective to learn meaningful regional representations such as phrases by aggregating word embeddings with convolutional filters (Kalchbrenner, Grefenstette, and Blunsom 2014). Based on this observation, Kim (2014) proposed a simple-yet-efficient CNN architecture which concatenates the outputs of filters with various sizes to build feature-rich n-gram representations. From another point of view, instead of concatenating features which make a network wide, Johnson and Zhang (2017) proposed deep pyramid convolutional neural networks to capture both local and global dependencies, which achieves the state-of-the-art performance on several text classification benchmarks.

The attention mechanism was originally used in neural machine translation to gather relevant information according to context. Vaswani et al. (2017) connected the dot between attention and sentence representation learning. They described self-attention as mapping a query and a set of key-value pairs to an output where the query, keys and values are in the same sequence. The output is a weighted sum of values where the weights are determined by a scoring function which models the relations between query and keys. Self-attention is highly efficient since it does not involve any recurrence process. In recent years, many researchers tried to adapt self-inference to different tasks. Devlin et al. (2018) proposed a deep self-attention based architecture and it reached the state-of-the-art for many tasks, which shows that although self-attention is a new encoding method, it still can compete with traditional methods like CNNs and RNNs.

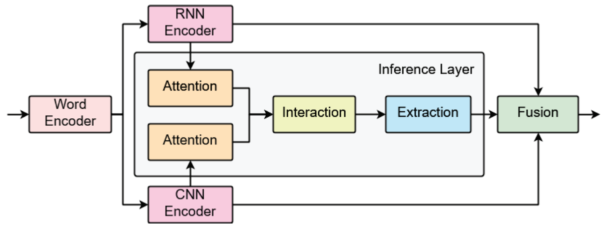

Figure 2: The high-level overview of our model.

Hybrid System

In addition to making a single encoding method deeper and more complicated, there are many studies which improve sentence modeling by mixing multiple encoding methods. As we discussed in our motivations, the encoders based on different hypotheses would tend to focus on different aspects of semantic information which can be further combined to from the general and comprehensive representation. Lai et al. (2015) analyzed the pros and cons of RNNs and CNNs and proposed the first hybrid system where a CNN is stacked on the RNNs to aggregate the left and right contexts of each word. Following the similar strategy, Zhou et al. (2016) stacked the compositions of 2D CNNs and 2D max-pooling on top of BiLSTMs to model the context on both the time dimension and the feature dimension, which achieved state-of-the-art performance on several benchmarks. To handle the hierarchical structure in natural language, stacking RNNs on top of CNNs is also a popular method (Wang, Jiang, and Luo 2016; Zhou et al. 2015). They applied CNNs on the word embeddings to extract meaningful n-gram features which are subsequently forwarded by the RNNs to capture the high-level dependency between the regional representations.

Although many studies have explored the implementation of hybrid systems, most of the prior works were based primarily on stacking various encoders, which raises two problems. First, there is no universal rule to determine the order of each encoder. We therefore have to search the best stacking strategy case by case. Second, the input of the encoders on the upper layer is the output of the encoders on the lower layer, which leads to the information bias and hence the performance of upper encoder would be limited by the lower encoder. It is also more difficult to fine-tune the model in this case because we cannot easily tell which layer in the architecture would be the bottleneck.

We therefore propose a collaborative strategy where encoders would capture observations on its own hypothesis and model the sentences independently. In other words, the encoders are totally unrelated to each other, which solves the problem in the current hybrid systems and makes our architecture quite flexible and parallelizable.

Our Method

The overview of our model is shown in Figure 2. Our model is composed of four parts: the word encoder, the context encoders, the self-inference layer and representation fusion layer. We will describe the architecture layer-by-layer in the following sections.

Word Encoding

To construct distributed word representations, we concatenate the word embedding and character embedding for each word. The word embedding was pretrained in an unsupervised manner on a large corpus to learn the relations among words in a low dimensional vector space. The character embedding is constructed by a convolutional neural network with max-pooling (Kim et al. 2015) which captures the sub-word structures and mitigates the impacts of out-of-vocab words. For each d-dimensional concatenated embedding w_t, where t indicates the time step, we pass it through a position-wise highway network (Srivastava, Greff, and Schmidhuber 2015) to obtain our feature-rich word representation x_t:

$$x_t = t_t \cdot h_t + (1 - t_t) \cdot w_t \tag{1}$$

$$h_t = \phi(\mathbf{W_h} w_t + b_h) \tag{2}$$

$$t_t = \sigma(\mathbf{W_t} w_t + b_t) \tag{3}$$

where $\mathbf{W_{h,t}} \in \mathbb{R}^{d \times d}$ and ϕ, σ are non-linear activation functions. The highway network plays the role of word encoder which aims at learning task-specific features by aggregating the information on word level and character level.

Context Encoding

Given an encoded sequence $(x_1, x_2, ..., x_n)$, to incorporate context information from various aspects, we model the sequence using both an RNN and CNN which capture the sequential and regional dependencies, respectively.

The RNN encoder is a single layer bidirectional GRU. The context representation at time step t is a concatenation of the output from forward and backward GRUs:

$$\overrightarrow{r_t} = \overrightarrow{\mathrm{GRU}_\mathrm{t}}(x_1, ..., x_n) \tag{4}$$

$$\overleftarrow{r_t} = \overleftarrow{\mathrm{GRU}_\mathrm{t}}(x_1, ..., x_n) \tag{5}$$

$$r_t = [\overrightarrow{r_t}, \overleftarrow{r_t}] \tag{6}$$

where $[\cdot, \cdot]$ is the concatenation operator.

The CNN encoder follows the idea of an n-gram but makes it more flexible. It uses several convolution units with different kernel sizes to extract local information in parallel. We then concatenate the output of each convolution unit to construct the context representation:

$$c_t = [\mathrm{Conv_{k1}}(x_t), ..., \mathrm{Conv_{kn}}(x_t)] \tag{7}$$

where $\mathrm{Conv_{kt}}$ is a 1-D convolution operation with a kernel size k_t. We use appropriate zero padding to make the concatenation operation feasible.

The RNN and CNN are regarded as the **base encoders** whose outputs are in the same k-dimensional space for the purpose of encoding context information in an equal manner.

Self-Inference

With two base encoders, a word can be interpreted as having two non-contradictory contexts. The aim of self-inference is to learn a single but well-generalized contextual representation by inferring the relations between the outputs of the base encoders.

Before inference, we first gather the semantic evidence. Besides contexts themselves, the dependency between contexts is also a key to understanding how a base encoder works. Specifically, we want to know how c_m, the context at the m time step, affects c_n for any possible (m, n) pair. To learn this pairwise relation, we apply self-attention (Vaswani et al. 2017) over the encoded contexts. The attention energy a_k is determined by the additive method which is described as:

$$a_k = \tanh(\mathbf{W_q} h_q + \mathbf{W_k} h_k + b_{qk}) \tag{8}$$

where h_q is the query and h_k is the key. The energies are normalized by softmax to represent the result as a probability weighted average of values. We concatenate the contexts with the results of self-attention to augment the evidence:

$$e_{rt} = [r_t, \mathrm{attention_t}(r_1, ..., r_n)] \tag{9}$$

$$e_{ct} = [c_t, \mathrm{attention_t}(c_1, ..., c_n)] \tag{10}$$

where $e_{rt,ct} \in \mathbb{R}^{2k}$, is the evidence at t time step to corresponding base encoders.

Inspired by the strategy in natural language inference, we use an interaction layer which describes the relations between evidence using the dense features, the distance features and the similarity features. The interaction vector would be further inferred by a feed-forward network:

$$i_t = [e_{rt}, e_{ct}, |e_{rt} - e_{ct}|, e_{rt} \odot e_{ct}] \tag{11}$$

$$g_t = \phi(\mathbf{W_i} i_t + b_i) \tag{12}$$

where $\mathbf{W_i} \in \mathbb{R}^{k \times 8k}$ and $\odot$ is the element-wise multiplication. With self-inference, g_t is the generalized contextual representation which captures semantic information based on not only the view but also the interaction of recurrent and convolutional encoders.

Representation Fusion

Considering the effectiveness of self-inference relies heavily on the performance of the base encoders, the contextual representations of our model are the weighted average of generalized representations and outputs from the base encoders.

The gate mechanism retains the connections between the output layers and base encoders, so part of gradients can be propagated directly back without being affected by the self inference component, which is important because we expect the base encoders to retain their original characteristics and capture word dependencies mainly based on their own encoding hypotheses. In particular, the gate dynamically generates the most relevant contextual mixture, depending on the importance of each representation at the same time step. Specifically, we concatenate g_t, r_t, c_t as an observation vector based on which we determine the importance using non-linear transformations:

$$\text{imp}_{rt} = \phi(\mathbf{W_{gr}}[r_t, c_t, g_t] + b_{gr}) \quad (13)$$
$$\text{imp}_{ct} = \phi(\mathbf{W_{gc}}[r_t, c_t, g_t] + b_{gc}) \quad (14)$$
$$\text{imp}_{gt} = \phi(\mathbf{W_{gg}}[r_t, c_t, g_t] + b_{gg}) \quad (15)$$

where the importance $\text{imp}_{rt,ct,gt} \in \mathbb{R}^k$ and $\mathbf{W_{gr,gc,gg}} \in \mathbb{R}^{k\times 3k}$. To make the importance of each dimension comparable, we permute the importance as a matrix whose rows are subsequently normalized by the softmax function:

$$\mathbf{W} = \text{softmax}([\text{imp}_{rt}; \text{imp}_{ct}; \text{imp}_{gt}]) \quad (16)$$

where $\mathbf{W} \in \mathbb{R}^{k\times 3}$ and $[\cdot;\cdot]$ is the operator for column vectors concatenation. Each column in $\mathbf{W}$ is a normalized weight vector. We then multiply the representations with weights to obtain the optimal fusion:

$$f_t = \mathbf{W^1} \odot r_t + \mathbf{W^2} \odot c_t + \mathbf{W^3} \odot g_t \quad (17)$$

Sentence Representation

We summarize the fused sequences $(f_1, ..., f_n)$ with max and average pooling:

$$f = [\max(f_1, ..., f_n), \text{avg}(f_1, ..., f_n)] \quad (18)$$

where f is a $2d$-dimensional vector which is the generalized sentence representation for subsequent downstream tasks.

Experiments

Dataset

We evaluate our model on four widely-studied benchmark datasets among three NLP tasks: natural language inference, text classification and sentiment classification.

Natural Language Inference (NLI) is a fundamental task in natural language understanding which aims to determine entailment relations between premise and hypothesis. We compare our approach against the state-of-the-art sentence encoders on two competitive dataset: MultiNLI (Williams, Nangia, and Bowman 2017), and Scitail (Khot, Sabharwal, and Clark 2018). To show the capacity of our method, we implement SINN as a Siamese encoder which would not propagate any information across the premise and hypothesis pairs.

For text classification, we use AG News (Zhang, Zhao, and LeCun 2015), a balanced dataset which consists of 120K web news articles pertaining to the 4 classes. Since there is no official validation set for this dataset, we split 5% training data for early-stopping and hyper-parameter searching.

For sentiment analysis, we use SST (Socher et al. 2013) to evaluate our model. We removed the neutral reviews and represent the sentiment scores with binary labels. Our model is trained with the phrases in the parse trees and tested on the whole sentence.

Experimental Settings

We initialize word embeddings using the pretrained FastText common-crawl vectors (Mikolov et al. 2018) and freeze the weights during training. The embedding of out of out-of-vocabulary words are initialized by a Gaussian distribution whose mean and standard deviation were calculated from the word vectors. The character embedding is composed of $(50, 55, 60, 65, 70)$ filters with kernel sizes $(2, 3, 4, 5, 6)$ followed by max pooling. The pooled outputs are then aggregated by a single-layer highway network.

For all tasks, we choose GELU (Hendrycks and Gimpel 2016) as the activation function. The CNN base encoder has $k/4$ filters with sizes $(2, 3, 4, 5)$, respectively. The classifier is a neural network of two-hidden layers followed by batch normalization and dropout layers. The objective function is cross-entropy loss which is optimized by Adam(Kingma and Ba 2014) with cyclic learning rate(Smith 2015). We applied label smoothing (Szegedy et al. 2015) to penalize confident predictions for NLI datasets and regularize the models with the L2 penalty. We search for the optimal hyper-parameters for each task by grid search.

Results

For the two NLI datasets, we compare SINN against the state-of-the-art sentence encoding based methods. The experimental results of MultiNLI are reported in Table 1. On the mismatched test set, SINN significantly outperforms the current state-of-the-art method by 0.8% with the least number of parameters. On the matched test set, our model is 0.2% lower than the best model. However, we do not train with external data and our number of parameters is one tenth of theirs, which keeps our performance stable across different evaluation sets.

The results of Scitail are reported in Table 2 where C refers to whether information would be passed across sentence pairs and E refers to the use of external datasets or pretrained models. We achieve a new best accuracy in Scitail. SINN can outperform HBMP (Talman, Yli-Jyrä, and Tiedemann 2018) which stacked three BiLSTM layers by 0.3% using a single context encoding layer and achieves around 2% improvement compared to the models that propagates information across the premise and the hypothesis, which shows that SINN can efficiently represent the semantic information of the original sequence with a single vector.

The results of text classification are reported in Table 3 where SINN is compared with models equipped with a single encoding method. For the RNN based models, SINN is obviously better than BiLSTM and Tree-LSTM. For the CNN based models, SINN is compared with two very deep architectures. The result demonstrates that the shallow 240D SINN can outperform VDCNN (Conneau et al. 2016) that stacks 29 layers and DPCNN (Johnson and Zhang 2017) that stacks 15 layers by 2.5% and 0.7%, respectively.

For sentiment analysis, We compare SINN with four baseline encoding methods as well as our base encoders on SST-2. As shown in Table 4, hybrid encoding is more efficient than a single type of encoding. After applying self-attention, we can observe improvements for both LSTM and CNN. Despite the weak performance of the base CNN encoder, with our inference process, SINN still can outperform all baselines and improve the base RNN and CNN by 0.7% and 1.8%, respectively.

Model	SNLI Mix	$\|\theta\|$	Test Accuracy (%) Matched	Mismatched
Deep Gated Attn. BiLSTM encoders (Chen et al. 2017)	X	11.6m	73.5	73.6
HBMP (Talman, Yli-Jyrä, and Tiedemann 2018)	X	22m	73.7	73.0
BiLSTM with generalized pooling (Chen, Ling, and Zhu 2018)	X	65m	73.8	74.0
Distance-based Self-Attention Network (Im and Cho 2017)	X	4.7m	74.1	72.9
Shortcut-Stacked BiLSTM encoders (Nie and Bansal 2017)	O	140.2m	**74.6**	73.6
Our Self-Inference Neural Network (600D)	X	5.6m	74.2	74.4
Our Self-Inference Neural Network (1200D)	X	13.2m	74.4	**74.8**

Table 1: Experimental results of MultiNLI. $\|\theta\|$ is the number of model parameters excluding word embeddings. *D denotes the dimension of sentence embeddings.

Model	C	E	Test acc.
DeIsTE (Yin, Roth, and Schütze 2018)	O	X	82.1
CAFE (Tay, Tuan, and Hui 2018)	O	X	83.3
MIMN (Liu et al. 2019)	O	X	84.0
ConSeqNet (Wang et al. 2019)	O	O	85.2
HBMP	X	X	86.0
Our SINN (1200D)	X	X	**86.3**

Table 2: Experimental results of Scitail where C represents whether the model has cross-sentence attention and E represents whether the model introduces external data.

Model	Test acc.
BiLSTM (Cho et al. 2014)	88.2
Tree-LSTM (Tai, Socher, and Manning 2015)	90.1
VDCNN (Conneau et al. 2016)	91.3
Capsule-B (Zhao et al. 2018)	92.6
DPCNN (Johnson and Zhang 2017)	93.1
Our SINN (240D)	**93.8**

Table 3: Experimental results of AG News.

Model	Test acc.
BiLSTM (Cho et al. 2014)	87.5
CNN (Kim 2014)	87.2
BiLSTM + self-attn. (Yoon, Lee, and Lee 2018)	88.2
CNN + self-attn. (Yoon, Lee, and Lee 2018)	88.3
Our Base CNN (600D)	86.8
Our Base RNN (600D)	87.9
Our SINN (600D)	**88.6**

Table 4: Experimental results of SST binary classification.

Method	Matched	Mismatched	AG News
BaseRNN	73.8	73.9	92.9
BaseCNN	72.3	72.5	93.0
Sum.	73.3	73.4	93.5
Concat.	73.7	73.4	93.8
WAVG.	73.8	74.0	93.5
SINN	**74.2**	**74.4**	**93.8**

Table 5: Comparisons of fusion methods.

Analysis

Fusion Method

In order to show improvement from the interaction and vector gates, we compare the performance of our fusion component with three baselines: concatenate, average, and weighted average of the outputs of base encoders. The results are reported in Table 5. For MultiNLI, most baselines do not improve the BaseRNN but for AG News all baselines outperform both base encoders significantly. This inconsistency is due to the difference between the performance of base encoders. The BaseRNN and BaseCNN have similar performance on AG News while there is a huge gap between them on MultiNLI.

Dataset	Gap	Sum.	Concat.	WAVG.	SINN
M	1.5%	-0.5%	-0.1%	-0.0%	**+0.4%**
MM	1.4%	-0.5%	-0.5%	+0.1%	**+0.4%**
AG	0.1%	+0.5%	**+0.8%**	+0.5%	**+0.8%**
Mean	1.0%	-0.2%	+0.1%	+0.2%	**+0.5%**

Table 6: The relations between the improvement and the gap. The symbols M, MM, AG are the abbreviations for Matched, MisMatched, AG-News, respectively.

To give a clear picture, we visualize the relations between the performance gap and the improvements in Table 6. Since baselines have no relation between base encoders, the information flow can not be gated based on the interaction. Therefore, their performance is dragged down by the BaseCNN. Unlike the baselines, SINN can dynamically choose the optimal representation fusion with self-inference, which is why it can always achieve the best accuracy even though a base encoder has undesirable performance.

To further explore the noise immunity of each baseline on the difficult tasks, we replace BaseCNN with BaseWord, an encoder which only uses a non-linear transform to obtain the context of a word, namely without any sequential dependency. The results are shown in Table 7. SINN still can improve the base encoders even though there is a huge gap since it can dynamically extract the relevant features and determine the optimal representation fusion with self-inference. It is important to note that for a general-purpose encoding architecture, this property is quite critical because we would not know if there is a huge difference in the performance of the base encoders in an unknown downstream task. Therefore, what we need is a solution that can improve the base encoders steadily and SINN is, however, the other baselines are not.

Dataset	Gap	Sum.	Concat.	WAVG.	SINN
M	8.1%	-2.2%	-2.0%	-1.3%	**+0.4%**
MM	8.0%	-2.4%	-2.4%	-1.5%	**-0.1%**

Table 7: The comparison of improvement and the gap between BaseRNN and BaseWord.

	Test Accuracy (%) / Difference (%)		
	Matched	Mismatched	SST-2
SINN	74.2	74.4	88.6
SINN - attn.	74.1 / -0.1	73.8 / **-0.6**	88.2 / -0.4
SINN - mul.	74.2 / -0.0	74.2 / -0.2	88.1 / -0.5
SINN - sub.	74.1 / -0.1	74.2 / -0.2	87.9 / -0.7
SINN - cat.	74.0 / -0.2	73.8 / -0.4	87.7 / **-0.9**
SINN - gate	73.0 / **-1.2**	74.1 / -0.3	88.0 / -0.6

Table 8: Experimental results of ablation study on the self-attention, interaction and fusion gate.

Ablation Study

To ensure the effectiveness of each component in self-inference, we perform a series of ablation analyses in Table 8. The results demonstrate that all components are needed and they make contributions on different parts. In the first experiment, after removing the self-attention module, the accuracy degrades to 74.1 on the matched set and 73.8 on the mismatch set, which indicates the dependency of contexts is also an important piece of evidence to learn how an encoder works. As observed in many studies, we can see that self-attention is an efficient method to aggregate the encoded contexts.

The second part is about the interaction layer. The concatenation is the most critical interaction which improves the matched set and mismatched set by 0.2% and 0.6%, respectively. The help of multiplication is marginal and it only improves the mismatched by set 0.2%. It is worth mentioning that our ablation study reaches the same conclusion as other NLI architectures. It may be a clue that self-inference is actually a special case of language inference and can be improved with certain well-established NLI solutions such as decomposable attention.

The third experiment is to remove the fusion gate and use the generalized representations directly, which results in a drop in both matched and mismatched accuracy. Especially in the matched set, the accuracy degrades to 73.0. As we pointed out in the motivation, the fusion gate that controls the information flow and generates the most relevant mixture plays a pivotal role in our method. In addition, after removing the fusion gate, there is no direct connection between the base encoders and the output layer, which makes it difficult for the base encoders to capture semantic dependency without being affected by the self-inference component.

Error analysis

To understand how self-inference improves our model, we split the test set in SST and the mismatched dev set in MultiNLI based on the following rules: (i) both base RNN and CNN make the correct prediction. (ii) either base RNN or CNN make the correct prediction. (iii) neither base RNN nor CNN make the correct prediction. The sizes of each subset are presented in Table 9 and The experimental results are reported in Table 10.

Dataset size	M	MM	SST	AG
(i) Both are correct	67%	67%	82%	92%
(ii) Someone is correct	12%	12%	10%	5%
(ii) None is correct	21%	21%	8%	3%

Table 9: The proportion of subset to the original dataset.

SINN Accuracy.	M	MM	SST	AG
(i) Both are correct	96%	96%	99%	99%
(ii) Someone is correct	59%	58%	66%	64%
(ii) None is correct	13%	16%	10%	18%

Table 10: The performance report of SINN on each subset.

When both RNN and CNN make the correct prediction, our model usually follows their agreement, especially in SST and AG-News where the error rate is only 1%. For the case that either base RNN or CNN is correct, the performance of SINN is also better than the random baseline, which shows that SINN has the ability to select the relevant context. To our surprise, SINN is able to solve 16% and 18% examples where both base encoders fail for Mismatched and AG-News, respectively. The result indicates that there exist certain linguistic features that cannot be captured by a single encoding method while these features can be simply created by aggregating the contextual representations between multiple encoders.

We visualize the confusion matrix of three methods on the Mismatched development set in Figure 3. The improvement of SINN mainly comes from the contradiction class and entailment class. For the base encoders, their tendencies to make predictions is quite similar. They both tend to mis-classify a contradiction or entailment example as neutral. And even though the overall performance of CNN is significant lower than RNN by 2.4%, it still can keep pace with RNN on detecting neutral examples, which is interesting and may indicate that for determining if a sentence pair is related or not, CNN might also be a considerable solution.

How to Infer

To gain a better insight into the scenarios where each encoder excels, we explore the weight distribution of importance in the fusion component. The results are shown in Table 11. On MultiNLI which require complicated language understanding, their importance is mainly based on the generalized representations. Because the generalized representations are derived from two base contexts, it has more semantic evidence for word sense disambiguation. As for why the importance of Scitail differs, we think it is because Scitail does not contain the contradiction category which simplifies the inference process.

For text classification, the CNN base encoder dominates

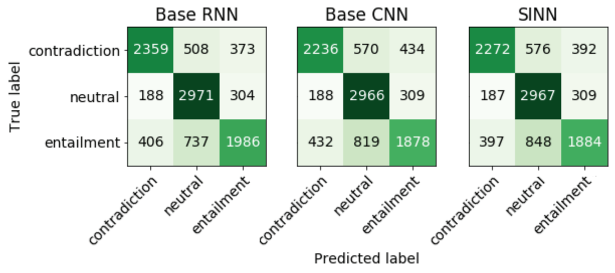

Figure 3: Confusion matrix of base RNN, CNN and SINN on MultiNLI Mismatched development set.

Dataset	Convol.	Recurrent.	General.
MultiNLI	0.24	0.24	**0.51**
Scitail	**0.43**	0.36	0.19
AG News	**0.38**	0.29	0.33
SST-2	**0.34**	**0.34**	0.32

Table 11: Fusion importance of each representation on all datasets.

other representations. It is because the labels in classification tasks are often based on whether a word or phrase appears, compared to long-term dependencies. Therefore regional semantic information, which CNN is good at, is more suitable for solving such problems.

For sentiment classification, there is no representation which stands out, which caught our attention. We therefore explore the relations between importance and contexts by sampling the top 10 important words from SST for each representation. We remove the low-frequency (<50) words to reduce the noise. The results are reported in Table 12. For the base CNN, all important words are verbs, while most words are adjectives for the base RNN and generalized representation. The difference may be caused by writing habits. For a verb, we can usually find the subject and object around it, but for an adjective, the subject can be in other sentence to express our opinion of an event, which needs to be captured by long-term dependency. In SST, the adjectives are more critical than verbs, which explains why the words that need to be further inferred are adjectives. Because the base RNN focuses on the same pattern as inferred unit, we can reweight the importance from $(0.34, 0.34, 0.32)$ to $(0.34, 0.66)$ where the former is for capturing verbs and the latter is for capturing adjectives.

Conclusion

We introduced SINN, a lightweight and effective hybrid system for sentence encoding which leverages the information of the base CNN and base RNN with self-inference. Our method achieves the state-of-the-art performance on four benchmarks among three NLP tasks. To know where the improvements come from, we conduct extensive ablation studies and error analysis. We also explore the importance distribution and the most activated words in the fusion gate to open the black box of our encoding method.

Representation	Top 10 important words
Convol.	walked, wondering, ask, trying, went, pulling did, try, searching, caught
Recurrent.	sobering, unflinching, hugely, harrowing, breathtakingly, unsentimental, enduring, uncommonly, harrowing, startling
General.	nor, insipid, worse, unfulfilling, unimaginative, lackluster, garbage, unfunny, bland, crap

Table 12: The top 10 important words for each representation in SST.

Model	AG News
SINN (240D)	93.8
Deep SINN (240D)	94.3

Table 13: Comparison between naive base encoders and deep base encoders.

In future work, we would like to make SINN much more deeper and wider. We choose two basic sequential models as base encoders to highlight the inference module. However, the performance of SINN is closely related to the base encoders. Table 13 illustrates that by simply stacking the base encoders with more layers, we can obtain a significant improvement. Therefore, we intend to evaluate a collection of complex sequential models, including the pre-trained language models, to investigate the interaction between different encoding hypotheses and to figure out the optimal combination of the base encoders.

References

Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; Jiang, H.; and Inkpen, D. 2017. Recurrent neural network-based sentence encoder with gated attention for natural language inference. *CoRR* abs/1708.01353.

Chen, Q.; Ling, Z.; and Zhu, X. 2018. Enhancing sentence embedding with generalized pooling. *CoRR* abs/1806.09828.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chung, J.; Gülçehre, Ç.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555.

Conneau, A.; Schwenk, H.; Barrault, L.; and LeCun, Y. 2016. Very deep convolutional networks for natural language processing. *CoRR* abs/1606.01781.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hendrycks, D., and Gimpel, K. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR* abs/1606.08415.

Im, J., and Cho, S. 2017. Distance-based self-attention network for natural language inference. *CoRR* abs/1712.02047.

Johnson, R., and Zhang, T. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 562–570.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A convolutional neural network for modelling sentences. *CoRR* abs/1404.2188.

Khot, T.; Sabharwal, A.; and Clark, P. 2018. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2015. Character-aware neural language models. *CoRR* abs/1508.06615.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lai, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Liu, C.; Jiang, S.; Yu, H.; and Yu, D. 2019. Multi-turn inference matching network for natural language inference. *CoRR* abs/1901.02222.

Mikolov, T.; Grave, E.; Bojanowski, P.; Puhrsch, C.; and Joulin, A. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Nie, Y., and Bansal, M. 2017. Shortcut-stacked sentence encoders for multi-domain inference. *CoRR* abs/1708.02312.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *CoRR* abs/1802.05365.

Smith, L. N. 2015. No more pesky learning rate guessing games. *CoRR* abs/1506.01186.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Highway networks. *CoRR* abs/1505.00387.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2015. Rethinking the inception architecture for computer vision. *CoRR* abs/1512.00567.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *CoRR* abs/1503.00075.

Talman, A.; Yli-Jyrä, A.; and Tiedemann, J. 2018. Natural language inference with hierarchical bilstm max pooling architecture. *CoRR* abs/1808.08762.

Tay, Y.; Tuan, L. A.; and Hui, S. C. 2018. A compare-propagate architecture with alignment factorization for natural language inference. *CoRR* abs/1801.00102.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *CoRR* abs/1706.03762.

Wang, X.; Kapanipathi, P.; Musa, R.; Yu, M.; Talamadupula, K.; Abdelaziz, I.; Chang, M.; Fokoue, A.; Makni, B.; Mattei, N.; and Witbrock, M. 2019. Improving natural language inference using external knowledge in the science questions domain. *CoRR* abs/1809.05724.

Wang, X.; Jiang, W.; and Luo, Z. 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2428–2437.

Williams, A.; Nangia, N.; and Bowman, S. R. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR* abs/1704.05426.

Yin, W.; Roth, D.; and Schütze, H. 2018. End-task oriented textual entailment via deep exploring inter-sentence interactions. *CoRR* abs/1804.08813.

Yoon, D.; Lee, D.; and Lee, S. 2018. Dynamic self-attention : Computing attention over words dynamically for sentence embedding. *CoRR* abs/1808.07383.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, 649–657.

Zhao, W.; Ye, J.; Yang, M.; Lei, Z.; Zhang, S.; and Zhao, Z. 2018. Investigating capsule networks with dynamic routing for text classification. *CoRR* abs/1804.00538.

Zhou, C.; Sun, C.; Liu, Z.; and Lau, F. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Zhou, P.; Qi, Z.; Zheng, S.; Xu, J.; Bao, H.; and Xu, B. 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *CoRR* abs/1611.06639.