Enhancing Local Feature Extraction with Global Representation for Neural Text Classification

Guocheng Niu[1*], Hengru Xu[2*†], Bolei He[1], Xinyan Xiao[1], Hua Wu[1], Sheng Gao[2]

[1] Baidu Inc., Beijing, China
[2] Beijing University of Posts and Telecommunications
[1]{niuguocheng, hebolei, xiaoxinyan, wu_hua}@baidu.com
[2]{xuhengru, gaosheng}@bupt.edu.cn

Abstract

For text classification, traditional local feature driven models learn long dependency by deeply stacking or hybrid modeling. This paper proposes a novel Encoder1-Encoder2 architecture, where global information is incorporated into the procedure of local feature extraction from scratch. In particular, Encoder1 serves as a global information provider, while Encoder2 performs as a local feature extractor and is directly fed into the classifier. Meanwhile, two modes are also designed for their interactions. Thanks to the awareness of global information, our method is able to learn better instance specific local features and thus avoids complicated upper operations. Experiments conducted on eight benchmark datasets demonstrate that our proposed architecture promotes local feature driven models by a substantial margin and outperforms the previous best models in the fully-supervised setting.

1 Introduction

Text classification is a fundamental task in natural language processing, which is widely used in various applications such as spam detection, sentiment analysis and topic classification. One of the mainstream approaches firstly utilizes explicit local extractors to identity key local patterns and classifies based on them afterwards. In this paper, we call this line of research as *local feature driven models*.

Lots of proposed methods can be grouped into this scope. Ngrams have been traditionally exploited in statistical machine learning approaches (Pang et al., 2002; Wang and Manning, 2012). For deep neural networks, encoding local features into low-dimensional distributed ngrams embeddings (Joulin et al., 2016; Qiao et al., 2018) and simply bagging of them have been proved effective and highly efficient. Convolutional Neural Networks (CNN) (LeCun et al., 2010) are promising methods for their strong capacities in capturing local invariant regularities (Kim, 2014). More recently, Wang (2018) proposes the Disconnected Recurrent Neural Network (DRNN), which utilizes RNN to extract local features for larger windows and has reported best results on several benchmarks.

Case1: ***Apple*** *is really amazing! I am fed up to carry my clunky camera.*
Case2: ***Apple*** *is famous around world and deserves to be called "nutritional powerhouses".*

Table 1: Topic classification examples for *Technology* and *Health*, where ***Apple*** is ambiguous within local context.

Despite having good interpretability and remarkable performance, current local feature extraction still has one shortcoming. As shown in Table 1, the real meaning of *Apple* can only be correctly recognized from overall view instead of narrow window. If the local extractor in charge of *Apple* cannot receive *camera* and *nutritional* from the very beginning, it would require complicated and costly upper structures to help revise the imprecisely local representation and create newer high-level features, such as deeply stacking (Johnson and Zhang, 2017; Conneau et al., 2016) and hybrid integration (Xiao and Cho, 2016). To a certain extend, it is inefficient and hard to train especially in the case of insufficient corpus.

To address this issue, we believe a more efficient approach is to optimize the local extraction process directly. In this paper, we propose

*These authors contributed equally to this work.
†This work was done while the author was an intern at Baidu Inc.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 496–506, Hong Kong, China, November 3–7, 2019.

a novel architecture named Encoder1-Encoder2[1], which innovatively contains two encoders for the identical input sequence respectively, instead of using only one single encoder in previous work. Concretely, the Encoder1 can be any kind of neural network models designed for briefly grasping global background, while the Encoder2 should be a typical local feature driven model. The key point is, the earlier generated global representations from Encoder1 is then incorporated into the local extraction procedure of Encoder2. In this way, local extractors can notice more long-range information on the basis of its natural advantages. As a result, better instance specific local features can be captured and directly utilized for classification owing to global awareness, which means further upper complicated operations can be avoided.

We conduct experiments on eight public text classification datasets introduced by Zhang et al. (2015). The experimental results show that our proposed architecture promotes local feature driven models by a substantial margin. In fully-supervised settings, our best models achieves new state-of-the-art performances on all benchmark datasets. We further demonstrate the ability and generalization of our architecture in the semi-supervised domain.

Our contributions can be concluded as follows:

1. We propose a novel Encoder1-Encoder2 architecture, where better instance specific local features are captured by incorporating global representations into local extraction procedure.

2. Our architecture has great flexibility. Different associations among Encoder1, Encoder2 and Interaction Modes are studied, where any kind of combination promotes vanilla CNN or DRNN significantly.

3. Our architecture is more robust to the window size of local extractors and the corpus scale.

2 Related Work

Local Feature Driven Models FastText uses bag of n-grams embeddings as text representation (Joulin et al., 2016), which has been proved effective and efficient. Qiao et al. (2018) propose a new method of learning and utilizing task specific n-grams embeddings to conquer data sparsity.

[1]Our code will be available at https://github.com/PaddlePaddle/models/tree/develop/PaddleNLP/Research/EMNLP2019-GELE. "GELE" is the abbreviation for Global Encoder and Local Encoder, i.e., Encoder1 and Encoder2 respectively.

CNN (LeCun et al., 2010) are representative methods of this category. Convolution operators are performed at every window based location to extract local features, interleaved with pooling layer for capturing invariant regularities. From Kim (2014), CNN are widely used in text classification. In addition to shallow structure, very deep and more complex CNN based models have also been studied to establish long distance association. Examples are deep character-level CNNs Zhang et al. (2015); Conneau et al. (2016), deep pyramid CNN Johnson and Zhang (2017) and convolution-recurrent networks Xiao and Cho (2016), in which recurrent layers are designed on top of convolutional layers for learning long-term dependencies between local features.

CNN use simple linear operations on n-gram vectors of each window, which enlightens researchers to capture higher order local non-linear feature using RNN. Shi et al. (2016) first replace convolution filters with LSTM for query classification. Wang (2018) proposes DRNN, which exploits large window size equipped with GRU.

To make full use of local and global information, Zhao et al. (2018) propose a sandwich network by carding a CNN in the middle of two LSTM layers, where the output of CNN provides local semantic representations while the top LSTM supplies global structure representations. However, the global information they mainly focus on is the syntax part, which is produced by reorganizing the already obtained local features. Besides, both of them are directly used for final classification, while we use pre-acquired global representations to help capture better local features. To the best of our knowledge, we are the first to incorporate global representation into the extraction procedure of local features for text classification.

Other Neural Network Models Recurrent Neural Networks (RNN) are naturally good at modeling variable-length sequential data and capturing long-term dependencies (Hochreiter and Schmidhuber, 1997; Chung et al., 2014). Global features are encoded by semantically synthesizing each word in the sequence in turn and there is no explicit small regions feature extraction procedure in this process. Lai et al. (2015) equip RNN with max-pooling to tackle the bias problem where later words are more dominant than earlier words. Tang et al. (2015) utilize LSTM to encode semantics of sentences and their relations in doc-

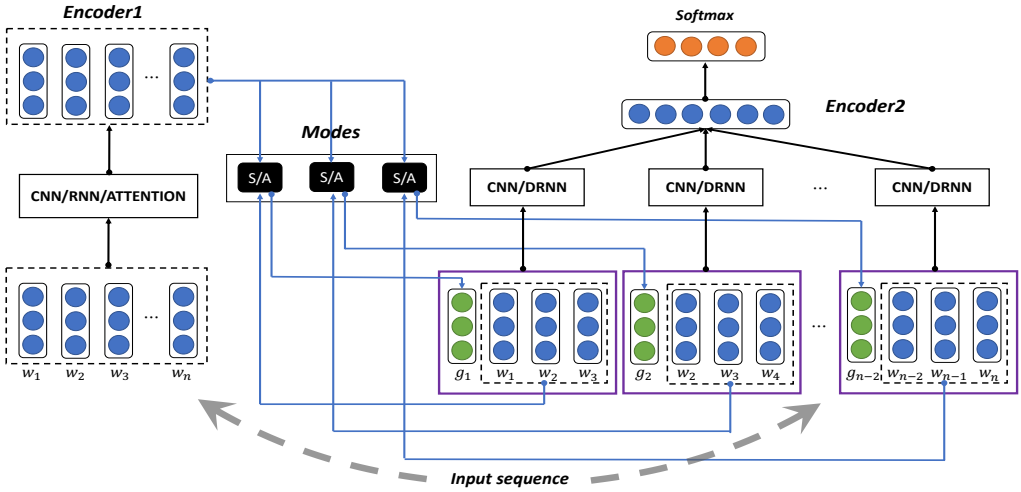


Figure 1: Encoder1-Encoder2 architecture mainly contains three components. (1) Encoder1 serves as a global information provider. (2) Encoder2 is a local feature driven model whose output is directly fed into the classifier. (3) Mode is the interaction manner between them. S and A are abbreviation of SAME and ATTEND respectively.

ument representation. Tai et al. (2015) introduce a tree-structured LSTM for sentiment classification.

The attention mechanism proposed by Bahdanau et al. (2014) has achieved great success in machine translation (Vaswani et al., 2017). For text classification which only has single input sequence, attention based models mainly focus on applying attention mechanism on top of CNN or RNN for selecting the more important information (Yang et al., 2016; Er et al., 2016). Letarte et al. (2018) and Shen et al. (2018) also explore self-attention networks which is CNN/RNN free.

3 Encoder1-Encoder2 Architecture

3.1 Overview

In this paper, we propose a novel neural network architecture named Encoder1-Encoder2 for text classification, which is illustrated in Figure 1. The identical input sequence will be encoded twice by two encoders respectively, but only the output of Encoder2 is used directly for the classifier. In particular, the Encoder1 serves as a pioneer for providing global information, while the Encoder2 focuses on extracting better local features by incorporating the former into the local extraction procedure. Besides, two Interaction Modes are developed for more targeted absorption of global information.

3.2 Encoder1: Global Information Provider

Without loss of generality, we introduce three types of models for Encoder1 in our architecture, each of which can be an independent global information provider and they are compared in our experiments.

CNN Let $\mathbf{x}_t$ be the d-dimensional word vector corresponding to the t-th word in a sequence of length n, $\mathbf{x}_{t-h+1:t}$ refers to the concatenation of words $\mathbf{x}_{t-h+1}, \mathbf{x}_{t-h+2}, \ldots, \mathbf{x}_t$ with size h and k number of filters are applied to the input sequence to generate features. Formally, filters $\mathbf{W}_f$ are applied to window $\mathbf{x}_{t-h+1:t}$ to compute $\mathbf{h}_t$:

$$\mathbf{h}_t = Conv(\mathbf{x}_{t-h+1}, \mathbf{x}_{t-h+2}, \ldots, \mathbf{x}_t) \quad (1)$$

$$= relu(\mathbf{W}_f \mathbf{x}_{t-h+1:t} + \mathbf{b}_f) \quad (2)$$

By same padding, filters are applied to n possible windows in the sequence and the global representation can be represented as $\mathbf{enc}_1$:

$$\mathbf{enc}_1 = [\mathbf{h}_1; \mathbf{h}_2; \ldots; \mathbf{h}_n] \quad (3)$$

GRU Gated recurrent units (GRU) are a gating mechanism in RNN (Cho et al., 2014). Two types of gates are used in GRU: reset gate decides how much new information is updated, while update gate controls the flow of previous information. The hidden state $\mathbf{h}_t$ is computed iteratively based on $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$. As a result, the all previous information can be encoded. For saving space, here we abbreviate it as:

$$\mathbf{h}_t = GRU(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t) \quad (4)$$

The global representation produced by GRU is hidden states of all time steps:

$$\mathbf{enc}_1 = [\mathbf{h}_1; \mathbf{h}_2; \ldots; \mathbf{h}_n] \quad (5)$$

Attention We also introduce attention mechanism on GRU for enhancing valuable information following Zhou et al. (2016). Define a context vector $\mathbf{u}_w$ to measure the importance of each hidden state $\mathbf{h}_t$ in GRU, which is randomly initialized and learned during training. A normalized importance weight α_t is obtained through a softmax function:

$$\alpha_t = \frac{\exp(\tanh(\mathbf{h}_t)^\top \mathbf{u}_w)}{\sum_t \exp(\tanh(\mathbf{h}_t)^\top \mathbf{u}_w)} \quad (6)$$

The global representation produced by this attention mechanism is expressed as:

$$\mathbf{enc}_1 = [\alpha_1 \mathbf{h}_1; \alpha_2 \mathbf{h}_2; \ldots; \alpha_n \mathbf{h}_n] \quad (7)$$

3.3 Encoder2: Variant Local Extractor

Vanilla local feature extractor strictly focuses on a limited size region. Here we propose a variant method. Apart from the expected local context, global information distilled by Encoder1 is also absorbed by a local extractor. In this way, the local features extracted by Encoder2 can notice the global background while still maintaining the position-invariant local patterns.

For Encoder2, we introduce two kinds of local feature driven models, i.e., CNN and DRNN. The former is good at capturing local spatial structure, while the latter is highlighted in capturing local temporal part. Set $\mathbf{g_t}$ as the required global information for a certain size window starting from $\mathbf{x}_t$, which will be introduced in 3.4 in detail.

CNN Here we treat each $\mathbf{g_t} \in \mathbb{R}^d$ as a faked extra global word, and do convolution with window words together. Based on Equation 1, features produced by filters for window $\mathbf{x}_{t-h+1:t}$ can be represented as:

$$\mathbf{h}_t = Conv(\mathbf{g_t}, \mathbf{x}_{t-h+1}, \mathbf{x}_{t-h+2}, \ldots, \mathbf{x}_t) \quad (8)$$

DRNN Different from CNN, DRNN utilizes RNN to extract local features for each window (Wang, 2018). To introduce global information into DRNN, faked global word $\mathbf{g_t}$ is filled in the head of each window like CNN does. Because of the sequential nature of RNN, even for a limited window, global information can be encoded into RNN from scratch and motivate the latter words. Here we use GRU as the local feature extractor, and features produced for window $\mathbf{x}_{t-h+1:t}$ can be represented as:

$$\mathbf{h}_t = GRU(\mathbf{g_t}, \mathbf{x}_{t-h+1}, \mathbf{x}_{t-h+2}, \ldots, \mathbf{x}_t) \quad (9)$$

To maintain translation invariant, a max-over-time pooling layer is then applied to CNN or DRNN layer, the pooling result is regarded as the output of Encoder2:

$$\mathbf{enc}_2 = maxpool([\mathbf{h}_1; \mathbf{h}_2; \ldots; \mathbf{h}_n]) \quad (10)$$

3.4 Interaction Modes between Encoders

Set $\mathbf{enc}_1$ as the global representation produced by Encoder1, required information for a certain window $\mathbf{x}_{t-h+1:t}$ with size h is defined as $\mathbf{g_t}$:

$$\mathbf{g_t} = G(\mathbf{enc}_1, \mathbf{x}_{t-h+1}, \mathbf{x}_{t-h+2}, \ldots, \mathbf{x}_t) \quad (11)$$

where G is a function of interaction mode. Two modes are devised from different point of views.

SAME Treat $\mathbf{enc}_1$ as a "reference book" provided by Encoder1. The basic idea of SAME Mode is each window in Encoder2 will get indiscriminate guidance regardless of the local information itself. For this purpose, max-over-time pooling is operated on $\mathbf{enc}_1$ directly to extract the most important information:

$$\hat{\mathbf{g}}_\mathbf{t} = maxpool(\mathbf{enc}_1) \quad (12)$$

ATTEND Mode ATTEND utilizes global information from another perspective. According to different local contexts, the guidances from Encoder1 can be be more targeted. Specifically, we use attention mechanism.

For window $\mathbf{x}_{t:t+h-1}$ with size h, the context vector is the average pooling of local words embeddings and the importance weight α_t for each hidden state $\mathbf{h}_t$ in Encoder1 can be computed as:

$$\alpha_t = \frac{\exp(\tanh(\mathbf{h}_t)^\top avg(\mathbf{x}_{t:t+h-1}))}{\sum_t \exp(\tanh(\mathbf{h}_t)^\top avg(\mathbf{x}_{t:t+h-1}))} \quad (13)$$

To maximize the profits obtained from Encoder1, we concatenate both of maxpooling results and attention results. Then $\hat{\mathbf{g}}_\mathbf{t}$ in ATTEND mode can be represented as:

$$\hat{\mathbf{g}}_\mathbf{t} = Concat(maxpool(\mathbf{enc}_1), \sum_t \alpha_t \mathbf{h}_t) \quad (14)$$

Finally, to keep consistent dimensions with words in the text, we compress $\hat{\mathbf{g}}_\mathbf{t}$ using MLP and formalized as $\mathbf{g_t}$, which can be easily embedded into the local feature extraction in Encoder2.

Dataset	C	L	N	Test	Tasks
Yelp P.	2	156	560k	38k	SA
Yelp F.	5	158	650k	50k	SA
Amz. P.	2	91	3M	650k	SA
Amz. F	5	93	3.6M	400k	SA
AG	4	44	120k	7.6k	News
Sogou	5	579	450k	60k	News
Yah. A.	10	112	1.4M	60k	QA
DBP	14	55	560k	70k	Ontology

Table 2: Datasets summary. C: Number of target classes. L: Average sentence length. N: Dataset size. Test: Test set size. In tasks, SA refers to sentiment analysis, and QA refers to question answering.

Corpus	Vocabulary	Length	Window size
Yelp P.	150k	300	20
Yelp F.	200k	300	20
Amz. P.	400k	256	15
Amz .F.	400k	256	15
AG	50k	128	15
Sogou	56k	400	15
Yah. A.	400k	256	20
DBP	400k	128	15

Table 3: Model settings. We limit the vocabulary size and set maximum sequence length. We also show the window size in DRNN following Wang (2018).

3.5 Classification Layer

After incorporating the global information obtained from Encoder1 into the local feature extraction of Encoder2, the output vector of latter can be regarded as the representation of the entire text. The vector is then fed into a softmax classifier to predict the probability of each category and cross entropy is used as loss function:

$$\hat{y} = softmax(\mathbf{W}_c\mathbf{enc}_2 + \mathbf{b}_c) \quad (15)$$

$$H(y, \hat{y}) = \sum_i y_i \log \hat{y}_i \quad (16)$$

where $\hat{y}_i$ is the predicted probability and y_i is the true probability of class i.

4 Experiments

We report experiments with proposed models in comparison with previous methods.

4.1 Experiments Settings

Datasets Publicly available datasets from Zhang et al. (2015) are used to evaluate our models. These datasets contain various domains and sizes, corresponding to sentiment analysis, news classification, question answering, and ontology extraction, which are summarized in Table 2.

Model Settings For data preprocessing, all the texts of datasets are tokenized by NLTKs tokenizer (Loper and Bird, 2002). For model hyperparameters, notations are following in 3.2 and 3.3. We adopt CNN, GRU and Attention for Encoder1. In CNN (Encoder1), we use filter windows (h) of $[3, 5, 7]$ with 128 feature maps (k) each. The hidden unit number is 128 in GRU (Encoder1) and Attention (Encoder1). For Encoder2, we conduct experiments based on two types of local feature extractor, corresponding to CNN and DRNN respectively. In CNN (Encoder2), window sizes of filters are of $[3, 5, 7]$ with 128 feature maps each. In DRNN (Encoder2), all the dimensions of hidden states are set to 300. Other settings are shown in Table 3, all trainable parameters including embeddings of words are initialized randomly without any pre-trained techniques (Mikolov et al., 2013; Peters et al., 2018; Devlin et al., 2018).

Training and Validation For each dataset, we randomly split the full training corpus into training and validation set, where the validation size is the same as the corresponding test size. Then the validation set is fixed for all models for fair comparison. The reported test accuracy is evaluated in the model which has lowest validation error.

AdaDelta (Zeiler, 2012) with $\rho = 0.95$ and $\epsilon = 1e - 6$ is chosen to optimize all the trainable parameters. Gradient norm clipping is employed to avoid the gradient explosion problem. L2 normalization is used in all models which include RNN structures. The batch size is set to 64 for Yelp P. and Yelp F. while 128 for other datasets. We train all the models using early stopping with timedelay 10.

4.2 Experimental Results and Analysis

Table 4 is the summary of the experimental results. We use underscores to represent the best published models, and bold the best records. Best models in our proposed architecture beat previous state-of-the-art models on all eight text classification benchmarks.

For published models, best results are achieved almost all by local feature driven models including Region-emb, VDCNN and DRNN. Self-Attention model SANet performs well, but does not achieve advantageous results as in sequence to sequence

Model	Yelp P.	Yelp F.	Amz. P.	Amz. F.	AG	Sogou	Yah. A.	DBP
bigram-FastText (Joulin et al., 2016)	95.7	63.9	94.6	60.2	92.5	96.8	72.3	98.6
Region-emb (Qiao et al., 2018)	96.2	64.5	95.3	60.8	92.8	97.3	73.4	98.9
SANet(big) (Letarte et al., 2018)	95.2	64.0	95.5	61.3	92.6	-	74.1	98.8
LSTM (Zhang et al., 2015)	94.7	58.2	93.9	59.4	86.1	95.2	70.8	98.6
D-LSTM (Yogatama et al., 2017)	92.6	59.6	-	-	92.1	94.9	73.7	98.7
char-CNN (Zhang et al., 2015)	94.7	62.0	94.5	59.6	87.2	95.1	71.2	98.3
VDCNN (Conneau et al., 2016)	95.7	64.7	95.7	63.0	91.3	96.8	73.4	98.7
CNN (Kim, 2014)	95.8	64.7	95.2	60.9	91.9	97.1	72.6	98.8
Encoder1-CNN-S (Ours)	96.5	66.2	95.9	**63.3**	92.5	**97.5**	74.5	98.8
Encoder1-CNN-A (Ours)	96.5	66.5	**96.0**	**63.3**	93.0	**97.5**	74.6	98.9
DRNN (Wang, 2018)	96.3	66.4	95.6	63.0	92.9	96.9	74.3	98.9
Encoder1-DRNN-S (Ours)	96.6	66.8	**96.0**	63.2	93.0	97.2	74.8	**99.0**
Encoder1-DRNN-A (Ours)	**96.7**	**67.0**	**96.0**	63.1	**93.2**	97.3	**75.0**	**99.0**

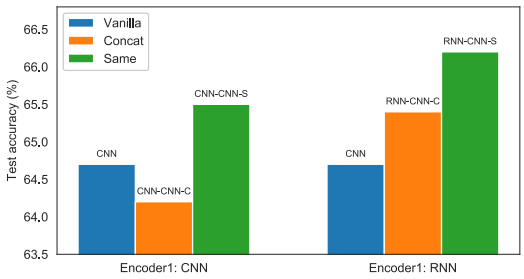
Table 4: Test accuracy [%] on several Datasets. We use Encoder1-Encoder2-Mode to represent our architecture, where S indicates mode SAME while A indicates ATTEND. For each specialized Encoder2, we test three different Encoder1. For the sake of brevity, *Encoder1* here does not specifically refer to a certain model but the best performance of the three combinations. Detailed experiments results for all combinations of Encoder1-Encoder2-Mode are further reported and compared in Table 5. For compared previous models, first block lists n-grams based models including bigram-FastText (Joulin et al., 2016) and region embedding (Qiao et al., 2018). Self-attention Networks SANet (Letarte et al., 2018) is reported in the second block. RNN based models LSTM (Zhang et al., 2015), D-LSTM (Yogatama et al., 2017) and CNN based models char-CNN (Zhang et al., 2015) and VDCNN (Conneau et al., 2016) are listed in third and forth block respectively. Strong local feature driven models CNN (Kim, 2014) and DRNN (Wang, 2018) are chosen as base model and directly compared with our architecture in last two blocks.

scenes, neither do RNN based methods. We argue that it is because key phrases and word order play an important role in text classification.

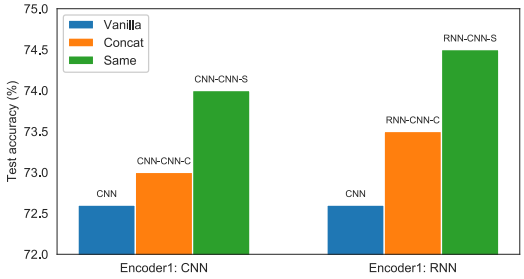
For our models, the experimental results show that enhanced local extractors with global encoder outperform vanilla local models by a advantageous margin. When CNN is chosen as local extractor, the performance gains are particularly significant for relatively difficult tasks such as Amz. F.(+2.4%) and Yah. A.(+2.0%). Encoder1-CNN performs even better than VDCNN with 29 convolutional layers. The results are satisfying considering that our CNN used as local extractor here is a shallow model with only one layer. Moreover, complicated VDCNN performs best among published models on larger datasets Amz. P.(95.7%) and Amz. F.(63.0%) but not as expected on smaller AG(91.3%), while our Encoder1-CNN has stable superior performance on all datasets. When DRNN is chosen as local extractor, the bonus from the global encoder is not so big like CNN, but still considerable and stable. Encoder1-DRNN beats DRNN on all datasets with a highest gain up to 0.7%.

To better analyze the impact of specific Encoder1 (global encoder) and different Interaction Modes on architecture performance, Table 5 details all combinations results of Encoder1-Encoder2-Mode on three datasets. We find the local extractor benefits quite a lot for any introduced global encoders. Overall, RNN and Attention based global encoders perform well-matched for local extractor, and both of them often perform better than CNN based global encoder. For example, Attention-CNN wins CNN-CNN 1.0% on Yelp F. and RNN-DRNN wins CNN-DRNN 0.4% on Yah. A. This is intuitive since RNN and Attention are more appropriate in capturing global information compared with CNN, which is critical for local extractor. The structures which specialize in modeling long-term dependency are more recommended as the global encoder.

For two Interaction Modes, we find ATTEND performs slightly better than SAME up to 0.4%, which can verify the differentiated motivation. Encoder1 (global encoder) can be viewed as a "reference book" about the whole text. Two Modes utilize the information from different perspectives to



(a) Experiments on Yelp F.



(b) Experiments on Yah. A.

Figure 2: Model ablation experiments. "Vanilla" is the traditional CNN (Kim, 2014). "Concat" simply concatenates the output of Encoder1 and Encoder2 and directly classifies according to the concatenated result. "Same" indicates the Interaction Mode in our architecture.

Model	Yelp P.	Yelp F.	Yah. A.
CNN	95.8	64.7	72.6
CNN-CNN-S	96.0	**65.5**	73.9
CNN-CNN-A	**96.3**	**65.5**	**74.1**
RNN-CNN-S	**96.5**	66.2	74.5
RNN-CNN-A	**96.5**	**66.4**	**74.6**
Attention-CNN-S	96.4	66.2	**74.2**
Attention-CNN-A	**96.5**	**66.5**	74.1
DRNN	96.3	66.4	74.3
CNN-DRNN-S	96.5	66.6	**74.6**
CNN-DRNN-A	**96.7**	**67.0**	74.4
RNN-DRNN-S	**96.6**	66.6	74.7
RNN-DRNN-A	**96.6**	**66.9**	**75.0**
Attention-DRNN-S	**96.6**	66.8	**74.9**
Attention-DRNN-A	**96.6**	**67.0**	**74.9**

Table 5: Effect of Encoder1 and Interaction Mode.

assist the local extractor. SAME Mode selects the most important information of global encoder and provides same guidance for each window in Encoder2, while the ATTEND Mode tends to make use of the "reference" with purpose based on different local contexts as if we refer to a reference book with initiative questions.

4.3 Model Ablation

In addition to introducing another encoder into vanilla local feature driven models, the greatest novelty of our architecture lies in that the global encoding is used to generate local features directly. Based on this motivation, the local features have the global awareness from the very beginning. To verify that our novel architecture makes key contribution to the performance improvement, we carry out model ablation experiments.

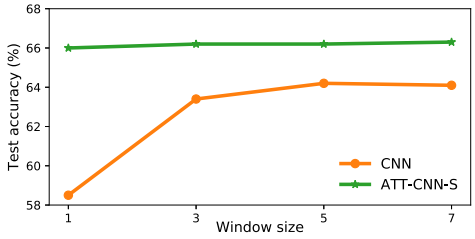
Without loss of generality, we use CNN as local extractor here and validate on Yelp F. and Yah. A. datasets. The results are illustrated in Figure 2.

Firstly, we list the results of Vanilla CNN, which is regarded as the most primitive state. Secondly, another additional encoder is introduced but they both process inputs independently and then their output representations are concatenated for classification. We call it "Concat", abbreviated as "C". For example, RNN-CNN-C stands for concatenating another RNN. Finally, we upgrade the way to use the introduced encoder as our proposed architecture. Here we list Mode SAME, abbreviated as "S".

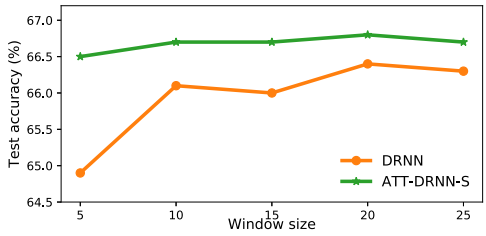
We find CNN-CNN-C loses 0.5% on Yelp P. but wins 0.4% on Yah. A. compared with vanilla CNN. CNN-CNN-C can be viewed as doubling convolution filters and we can observe that introducing more parameters does not always perform better. Meanwhile, RNN-CNN-C wins vanilla CNN 0.7% on Yelp P. and 0.9% on Yah. A. It makes sense since the classifier could use features from CNN and RNN simultaneously and different model structures complement each other for classification. In particular, our architecture performs best for both cases. CNN-CNN-S wins CNN-CNN-C 1.3% and 0.8%, and RNN-CNN-S wins RNN-CNN-C 0.6% and 1.0% on Yelp P. and Yah. A. respectively. In fact, CNN-CNN-S does not introduce new model structure or complicated operations and the number of parameters are almost the same. We attribute the great improvement to our novel mechanism where the global representation conduces to the local extraction.

4.4 Effect of Window Size

As an important hyperparameter, window size determines how much information can be seen in a specific window and often requires carefully tun-



(a) The impact of window size on CNN.



(b) The impact of window size on DRNN.

Figure 3: Window size experiments on Yelp F. ATT is the abbreviation of Attention.

Model	Sentence Samples
CNN	*"Commission backs 5bn British Energy deal "" British Energy, the nuclear generator yesterday welcomed a decision by the European commission to approve a government-backed 5bn rescue plan ."* **World ✗**
ATT-CNN	*"Commission backs 5bn British Energy deal "" British Energy, the nuclear generator, yesterday welcomed a decision by the European commission to approve a government-backed 5bn rescue plan ."* **Business ✓**
CNN	*The mac and cheese sticks were amazing ... highly recommend them . Overall, for the high price price pay here, I would rather be across the casino with at least a great view of the fountains.* **Positive ✗**
ATT-CNN	*The mac and cheese sticks were amazing ... highly recommend them . Overall, for the high price you pay here, I would rather be across the casino with at least a great view of the fountains.* **Negative ✓**

Table 6: Visualization of chosen samples on AG News and Yelp Review Polarity dataset. We use SAME Interaction Mode in ATT-CNN where ATT is the abbreviation of Attention.

ing in traditional method. Small window sizes may result in the loss of some critical information whereas large windows result in an enormous parameter space, which could be difficult to train (Lai et al., 2015). In this section, we analyze the impact of different window sizes on model performances.

As shown in Figure 3, both CNN and DRNN are very sensitive to window size, the optimal window size in DRNN can be much larger than CNN due to the sequential memory in RNN structure. Tuning these models is often challenging. In contrast, our Encoder1-Encoder2 architecture is insensitive to the parameter and achieves stable satisfied performance in various window sizes. We believe it is because the local extraction has been enhanced by global information and not strictly dependent on large windows to capture long range information.

4.5 Case Study and Visualization

To investigate how our architecture makes a difference in details, we visualize the attending phrases by the neural model in Table 6. Qualitatively, we display the contribution of phrases in Encoder2 to classification via max-pooling. The most important phrases are highlighted red where the intensity of the color indicates the contribution. Meanwhile, we use waves to roughly indicate the key phrases with high attention scores in Encoder1. Detailed visualization techniques have been introduced in Li et al. (2015).

The first two lines compare CNN with our Attention-CNN on an example from AG News. CNN wrongly captures key phrases *British Energy* and *the nuclear generator* and thus misclassifies the example into *World*. In contrast, our Attention-CNN is able to correctly classify it into *Business*. The Encoder1 firstly captures the global description by *Energy deal*, *nuclear*, and *rescue plan*. Informed with these global information, Encoder2 reduces its attention to *nuclear*, which implies label *World* while captures key phrases *British Energy deal* and *5bn rescue plan*. Accordingly the model makes a correct prediction labeled as *Business*. For the second example, the global representations include phrases *high price* and conjunction *Overall*, making Encoder2 activate *I would rather* while reduce its sensitivity to *highly recommend them* compared with CNN.

In short, the global representations learned by Encoder1 provide a brief overall grasp of the whole text, which includes both semantic and structure information. It effectively helps Encoder2 capture better instance specific local features and improve model performance.

4.6 Comparison to Pre-trained Models

As shown above, our paper mainly focuses on fully-supervised domain where all model parameters are trained from scratch. Alternatively, substantial work has shown that pre-trained models are beneficial for various NLP tasks. Typically, they first pre-train neural networks on large-scale unlabeled text corpora, and then finetune the models or representations on downstream tasks.

One kind of pre-trained models is the word embeddings, such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). More recently, by utilizing larger-scale unsupervised corpus and deeper architecture, pre-trained language models have shown to be effective in learning common language representations and have achieved great success. Among them, OpenAI GPT (Radford et al., 2018), BERT (Devlin et al., 2018), XLNet (Yang et al., 2019) and ERNIE 2.0 (Sun et al., 2019) are the most remarkable examples.

In this section, we generalize our architecture to semi-supervised domain which is equipped with pre-trained word embeddings and then compare with popular pre-trained based models. Specifically, we use GloVe vectors[2] with 300 dimensions to initialize the word embeddings in our architecture. BERT_{BASE}[3] and ERNIE 2.0_{BASE}[4] with 12-layer Transformer (Vaswani et al., 2017) are chosen for comparison. Here we report best model for each specialized Encoder2 with SAME Mode. Results on three datasets are listed in Table 7.

Overall, our architecture can be further boosted a lot by utilizing pre-trained word embeddings. For example, *Encoder1*-DRNN-S obtains a new score of 76.2%(+1.4%) on Yah. A. and *Encoder1*-CNN-S gets 94.1%(+1.6%) on AG. Vanilla local extractors also achieve better performance as expected in most instances while our models are still much better than them. *Encoder1*-CNN-S outperforms CNN by 0.9%, 1.8% and 2.0% on three datasets respectively, and *Encoder1*-DRNN-S outperforms DRNN by 0.4%, 0.6% and 0.7%. It shows that our architecture is well generalized and compatible with pre-training techniques.

It is interesting to compare with stronger pre-trained models. Although we obtain close scores on AG, BERT and ERNIE 2.0 indeed achieve more advanced results on others and the latter performs best on all three datasets. Despite their superb accuracy, we argue that the huge models are resource-hungry in practice. Lightweight models still have advantages under some circumstances such as limited memory, longer text data to be processed and requirements of faster inference time.

Model	Yelp F.	AG	Yah. A.
CNN(n/y)	64.7/64.5	91.9/92.3	72.6/73.7
Encoder1-CNN-S(n/y)	66.2/66.6	92.5/94.1	74.5/75.7
DRNN(n/y)	66.4/66.8	92.9/93.6	74.3/75.5
Encoder1-DRNN-S(n/y)	66.8/67.2	93.0/94.2	74.8/76.2
BERT_{BASE}	67.9	94.2	76.4
ERNIE 2.0_{BASE}	**69.1**	**94.3**	**77.0**

Table 7: Semi-supervised generalization of our architecture and comparison with popular pre-trained models. Here "n" and "y" stand for initializing word embeddings randomly and with pre-trained GloVe vectors separately.

5 Conclusion

In this work, we demonstrate the local feature extraction can be significantly enhanced with global information. Instead of traditionally exploiting deeper and complicated operations in upper neural layers, our work innovatively provides another lightweight way for improving the ability of neural model. Specifically, we propose a novel architecture named Encoder1-Encoder2 with two Interaction Modes for their interacting. The architecture has high flexibility and our best models achieve new state-of-the-art performance in fully-supervised setting on all benchmark datasets. We also find that our architecture is insensitive to window size and enjoy a better robustness. In future work, we plan to validate its effectiveness for multi-label classification. Besides, we are interested in incorporating more powerful unsupervised methods into our architecture.

Acknowledgments

This work is supported by the Natural Science Foundation of China (No. 61533018). We gratefully thank the anonymous reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

[2]http://nlp.stanford.edu/projects/glove
[3]https://github.com/google-research/bert
[4]https://github.com/PaddlePaddle/ERNIE

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

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

Meng Joo Er, Yong Zhang, Ning Wang, and Mahardhika Pratama. 2016. Attention pooling-based convolutional neural network for sentence modelling. *Information Sciences*, 373:388–403.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rie Johnson and Tong Zhang. 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)*, volume 1, pages 562–570.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

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

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. 2010. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE.

Gaël Letarte, Frédérik Paradis, Philippe Giguère, and François Laviolette. 2018. Importance of self-attention for sentiment analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 267–275.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

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

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*.

Chao Qiao, Bo Huang, Guocheng Niu, Daren Li, Daxiang Dong, Wei He, Dianhai Yu, and Hua Wu. 2018. Anew method of region embedding for text classification. In *International Conference on Learning Representations*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yangyang Shi, Kaisheng Yao, Le Tian, and Daxin Jiang. 2016. Deep lstm based feature mapping for query classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1501–1511.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. Ernie 2.0: A continual pre-training framework for language understanding. *arXiv preprint arXiv:1907.12412*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Baoxin Wang. 2018. Disconnected recurrent neural networks for text categorization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2311–2320.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Jianyu Zhao, Zhiqiang Zhan, Qichuan Yang, Yang Zhang, Changjian Hu, Zhensheng Li, Liuxin Zhang, and Zhiqiang He. 2018. Adaptive learning of local semantic and global structure representations for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2033–2043.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.