Learning Context-Aware Convolutional Filters for Text Processing

Dinghan Shen[1], **Martin Renqiang Min**[2], **Yitong Li**[1], **Lawrence Carin**[1]
[1] Duke University [2] NEC Laboratories America
dinghan.shen@duke.edu, renqiang@nec-labs.com, yitong.li@duke.edu, lcarin@duke.edu

Abstract

Convolutional neural networks (CNNs) have recently emerged as a popular building block for natural language processing (NLP). Despite their success, most existing CNN models employed in NLP share the same learned (and static) set of filters for all input sentences. In this paper, we consider an approach of using a small *meta network* to learn context-aware convolutional filters for text processing. The role of meta network is to abstract the contextual information of a sentence or document into a set of *input-aware* filters. We further generalize this framework to model sentence pairs, where a *bidirectional* filter generation mechanism is introduced to encapsulate co-dependent sentence representations. In our benchmarks on four different tasks, including ontology classification, sentiment analysis, answer sentence selection, and paraphrase identification, our proposed model, a modified CNN with context-aware filters, consistently outperforms the standard CNN and attention-based CNN baselines. By visualizing the learned *context-aware* filters, we further validate and rationalize the effectiveness of proposed framework.

1 Introduction

In the last few years, convolutional neural networks (CNNs) have demonstrated remarkable progress in various natural language processing applications (Collobert et al., 2011), including sentence/document classification (Kim, 2014; Zhang et al., 2015; Wang et al., 2018), text sequence matching (Hu et al., 2014; Yin et al., 2016; Shen et al., 2017), generic text representations (Gan et al., 2016; Tang et al., 2018), language modeling (Dauphin et al., 2017), machine translation (Gehring et al., 2017) and abstractive sentence summarization (Gehring et al., 2017). CNNs are typically applied to tasks where feature extraction and a corresponding supervised task are approached jointly (LeCun et al., 1998). As an encoder network for text, CNNs typically convolve a set of filters, of window size n, with an input-sentence embedding matrix obtained via *word2vec* (Mikolov et al., 2013) or Glove (Pennington et al., 2014). Different filter sizes n may be used within the same model, exploiting meaningful semantic features from different n-gram fragments.

The learned weights of CNN filters, in most cases, are assumed to be fixed regardless of the input text. As a result, the rich contextual information inherent in natural language sequences may not be fully captured. As demonstrated in Cohen and Singer (1999), the context of a word tends to greatly influence its contribution to the final supervised tasks. This observation is consistent with the following intuition: when reading different *types* of documents, *e.g.*, academic papers or newspaper articles, people tend to adopt distinct strategies for better and more effective understanding, leveraging the fact that the same words or phrases may have different meaning or imply different things, depending on context.

Several research efforts have sought to incorporate *contextual information* into CNNs to adaptively extract text representations. One common strategy is the *attention mechanism*, which is typically employed on top of a CNN (or Long Short-Term Memory (LSTM)) layer to guide the extraction of semantic features. For the embedding of a single sentence, Lin et al. (2017) proposed a self-attentive model that attends to different parts of a sentence and combines them into multiple vector representations. However, their model needs considerably more parameters to achieve performance gains over traditional CNNs. To match sentence pairs, Yin et al. (2016) introduced an attention-based CNN model, which re-weights the convolution inputs or outputs, to extract interdepen-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1839–1848
Brussels, Belgium, October 31 - November 4, 2018.

dent sentence representations. Wang et al. (2016); Wang and Jiang (2017) explore a *compare and aggregate* framework to directly capture the word-by-word matching between two paired sentences. However, these approaches suffer from the problem of high matching complexity, since a similarity matrix between pairwise words needs to be computed, and thus it is computationally inefficient or even prohibitive when applied to long sentences (Mou et al., 2016).

In this paper, we propose a generic approach to learn *context-aware* convolutional filters for natural language understanding. In contrast to traditional CNNs, the convolution operation in our framework does not have a fixed set of filters, and thus provides the network with stronger modeling flexibility and capacity. Specifically, we introduce a *meta network* to generate a set of context-aware filters, conditioned on specific input sentences; these filters are adaptively applied to either the same (Section 3.2) or different (Section 3.3) text sequences. In this manner, the learned filters vary from sentence to sentence and allow for more fine-grained feature abstraction.

Moreover, since the generated filters in our framework can adapt to different conditional information available (labels or paired sentences), they can be naturally generalized to model sentence pairs. In this regard, we propose a novel *bidirectional* filter generation mechanism to allow interactions between sentence pairs while constructing *context-aware* representations.

We investigate the effectiveness of our Adaptive Context-sensitive CNN (ACNN) framework on several text processing tasks: ontology classification, sentiment analysis, answer sentence selection and paraphrase identification. We show that the proposed methods consistently outperforms the standard CNN and attention-based CNN baselines. Our work provides a new perspective on how to incorporate contextual information into text representations, which can be combined with more sophisticated structures to achieve even better performance in the future.

2 Related Work

Learning deep text representations has attracted much attention recently, since they can potentially benefit a wide range of NLP applications (Collobert et al., 2011; Kim, 2014; Wang et al., 2017a; Shen et al., 2018a; Tang and de Sa, 2018; Zhang et al., 2018). CNNs have been extensively investigated as the encoder networks of natural language. Our work is in line with previous efforts on improving the adaptivity and flexibility of convolutional neural networks (Jeon and Kim, 2017; De Brabandere et al., 2016). Jeon and Kim (2017) proposed to enhance the transformation modeling capacity of CNNs by adaptively learning the filter shapes through backpropagation. De Brabandere et al. (2016) introduced an architecture to generate the future frames conditioned on given image(s), by adapting the CNN filter weights to the motion within previous video frames. Although CNNs have been widely adopted in a large number of NLP applications, improving the adaptivity of vanilla CNN modules has been considerably less studied. To the best of our knowledge, the work reported in this paper is the first attempt to develop more flexible and adjustable CNN architecture for modeling sentences.

Our use of a meta network to generate parameters for another network is directly inspired by the recent success of hypernetworks for text-generation tasks (Ha et al., 2017), and dynamic parameter-prediction for video-frame generation (De Brabandere et al., 2016). In contrast to these works that focus on generation problems, our model is based on context-aware CNN filters and is aimed at abstracting more informative and predictive sentence features. Most similar to our work, Liu et al. (2017) designed a meta network to generate compositional functions over tree-structured neural networks for encapsulating sentence features. However, their model is only suitable for encoding individual sentences, while our framework can be readily generalized to capture the interactions between sentence pairs. Moreover, our framework is based on CNN models, which is advantageous due to fewer parameters and highly parallelizable computations relative to sequential-based models.

3 Model

3.1 Basic CNN for text representations

The CNN architectures in (Kim, 2014; Collobert et al., 2011) are typically utilized for extracting sentence representations, by a composition of a convolutional layer and a *max-pooling* operation over all resulting feature maps. Let the words of a sentence of length T (padded where necessary) be $x_1, x_2, \ldots, x_T$. The sentence can be represented

as a matrix $\boldsymbol{X} \in \mathbb{R}^{d \times T}$, where each column represents a d-dimensional embedding of the corresponding word.

In the convolutional layer, a set of filters with weights $\boldsymbol{W} \in \mathbb{R}^{K \times h \times d}$ is convolved with every window of h words within the sentence, *i.e.*, $\{x_{1:h}, x_{2:h+1}, \ldots, x_{T-h+1:T}\}$, where K is the number of output feature maps (and filters). In this manner, feature maps $\boldsymbol{p}$ for these h-gram text fragments are generated as:

$$\boldsymbol{p}_i = f(\boldsymbol{W} \times x_{i:i+h-1} + b) \tag{1}$$

where $i = 1, 2, ..., T - h + 1$ and $\times$ denotes the convolution operator at the ith shift location. Parameter $b \in \mathbb{R}^K$ is the bias term and $f(\cdot)$ is a non-linear function, implemented as a rectified linear unit (ReLU) in our experiments. The output feature maps of the convolutional layer, *i.e.*, $\boldsymbol{p} \in \mathbb{R}^{K \times (T-h+1)}$ are then passed to the pooling layer, which takes the maximum value in every row of $\boldsymbol{p}$, forming a K-dimensional vector, $\boldsymbol{z}$. This operation attempts to keep the most salient feature detected by every filter and discard the information from less fundamental text fragments. Moreover, the *max-over-time* nature of the pooling operation (Collobert et al., 2011) guarantees that the size of the obtained representation is independent of the sentence length.

Note that in basic CNN sentence encoders, filter weights are the same for different inputs, which may be suboptimal for feature extraction (De Brabandere et al., 2016), especially in the case where conditional information is available.

3.2 Learning context-sensitive filters

The proposed architecture to learn context-sensitive filters is composed of two principal modules: (*i*) a filter generation module, which produces a set of filters conditioned on the input sentence; and (*ii*) an adaptive convolution module, which applies the generated filters to an input sentence (this sentence may be either the same as or different from the first input, as discussed further in Section 3.3). The two modules are jointly differentiable, and the overall architecture can be trained in an end-to-end manner. Since the generated filters are sample-specific, our ACNN feature extractor for text tends to have stronger predictive power than a basic CNN encoder. The general ACNN framework is shown schematically in Figure 1.

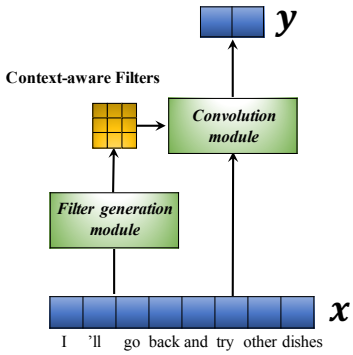


Figure 1: The general ACNN framework. Notably, the input sentences to filter generating module and convolution module could be different (see Section 3.3).

Filter generation module Instead of utilizing fixed filter weights $\boldsymbol{W}$ for different inputs (as (1)), our model generates a set of filters conditioned on the input sentence $\boldsymbol{X}$. Given an input $\boldsymbol{X}$, the filter-generation module can be implemented, in principle, as any deep (differentiable) architecture. However, in order to handle input sentences of variable length common in natural language, we design a generic filter generation module to produce filters with a predefined size.

First, the input $\boldsymbol{X}$ is encapsulated into a fixed-length vector (code) $\boldsymbol{z}$ with the dimension of l, via a basic CNN model, where one convolutional layer is employed along with the pooling operation (as described in Section 3.1). On top of this hidden representation $\boldsymbol{z}$, a deconvolutional layer, which performs transposed operations of convolutions (Radford et al., 2016), is further applied to produce a unique set of filters for $\boldsymbol{X}$ (as illustrated in Figure 1):

$$\boldsymbol{z} = \text{CNN}(\boldsymbol{X}; \boldsymbol{\theta}_e), \tag{2}$$

$$\boldsymbol{f} = \text{DCNN}(\boldsymbol{z}; \boldsymbol{\theta}_d)\,, \tag{3}$$

where $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$ are the learned parameters in each layer of the filter-generating module, respectively. Specifically, we convolve $\boldsymbol{z}$ with a filter of size (f_s, l, k_x, k_y), where f_s is the number of generated filters and the kernel size is (k_x, k_y). The output will be a tensor of shape (f_s, k_x, k_y). Since the dimension of hidden representation $\boldsymbol{z}$ is independent of input-sentence length, this framework guarantees that the generated filters are of the same shape and size for every sentence. Intuitively, the encoding part of filter generation module abstracts the information from sentence $\boldsymbol{X}$ into $\boldsymbol{z}$. Based on this representation, the deconvolutional up-sampling layer determines a set of fixed-size, fine-grained filters $\boldsymbol{f}$ for the specific input.

Adaptive convolution module The adaptive convolution module takes as inputs the generated filters $\boldsymbol{f}$ and an input sentence. This sentence and the input to the filter-generation module may be identical (as in Figure 1) or different (as in Figure 2). With the sample-specific filters, the input sentence is adaptively encoded, again, via a basic CNN architecture as in Section 3.1, *i.e.*, one convolutional and one pooling layer. Notably, there are no additional parameters in the adaptive convolution module (no bias term is employed).

Our ACNN framework can be seen as a generalization of the basic CNN, which can be represented as an ACNN by setting the outputs of the filter-generation module to a constant, regardless of the contextual information from input sentence(s). Because of the learning-to-learn (Thrun and Pratt, 2012) nature of the proposed ACNN framework, it tends to have greater representational power than the basic CNN.

3.3 Extension to text sequence matching

Considering the ability of our ACNN framework to generate context-aware filters, it can be naturally generalized to the task of text sequence matching. In this section, we will describe the proposed Adaptive Question Answering (AdaQA) model in the context of answer sentence selection task. Note that the corresponding model can be readily adapted to other sentence matching problems as well (see Section 5.2).

Given a factual question q (associated with a list of candidate answers $\{a_1, a_2, \ldots, a_m\}$ and their corresponding labels $\boldsymbol{y} = \{y_1, y_2, \ldots, y_m\}$), the goal of the model is to identify the correct answers from the set of candidates. For $i = 1, 2, \ldots, m$, if a_i correctly answers q, then $y_i = 1$, and otherwise $y_i = 0$. Therefore, the task can be cast as a classification problem where, given an unlabeled question-answer pair (q_i, a_i), we seek to predict the judgement y_i.

Conventionally, a question q and an answer a are independently encoded by two basic CNNs to fixed-length vector representations, denoted $\boldsymbol{h}_q$ and $\boldsymbol{h}_a$, respectively. They are then directly employed to predict the judgement $\boldsymbol{y}$. This strategy could be suboptimal, since no communication (information sharing) occurs between the question-answer pair until the top prediction layer. Intuitively, while the model is inferring the representation for a question, if the meaning of the answer is taken into account, those features that are relevant for final prediction are more likely to be extracted. So motivated, we propose an adaptive CNN-based question-answer (AdaQA) model for this problem. The AdaQA model can be divided into three modules: filter generation, adaptive convolution, and matching modules, as depicted schematically in Figure 2. Assume there is a question-answer pair to be matched, represented by word-embedding matrices, *i.e.* $\boldsymbol{Q} \in \mathbb{R}^{T_q \times d}$ and $\boldsymbol{A} \in \mathbb{R}^{T_a \times d}$, where d is the embedding dimension and T_q and T_a are respective sentence lengths. First, they are passed to two filter-generation modules, to produce two sets of filters that encapsulate features of the corresponding input sentences. Similar to the setup in Section 3.2, we also employ a two-step process to produce the filters. For a question $\boldsymbol{Q}$, the generating process is:

$$\boldsymbol{z}_q = \text{CNN}(\boldsymbol{Q}; \boldsymbol{\theta}_e^q), \tag{4}$$

$$\boldsymbol{f}_q = \text{DCNN}(\boldsymbol{z}_q; \boldsymbol{\theta}_d^q) \tag{5}$$

where CNN and DCNN denote the basic CNN unit and deconvolution layer, respectively, as discussed in Section 2.1. Parameters $\boldsymbol{\theta}_e^q$ and $\boldsymbol{\theta}_d^q$ are to be learned. The same process can be utilized to produce encodings $\boldsymbol{z}_a$ and filters $\boldsymbol{f}_a$ for the answer input, $\boldsymbol{A}$, with parameters $\boldsymbol{\theta}_e^a$ and $\boldsymbol{\theta}_d^a$, respectively.

The two sets of filter weights are then passed to adaptive convolution modules, along with $\boldsymbol{Q}$ and $\boldsymbol{A}$, to obtain the extracted question and answer embeddings. That is, the question embedding is convolved with the filters produced by the answer and *vise versa* (ψ_q and ψ_a are the bias terms to be learned). The key idea is to abstract information from the answer (or question) that is pertinent to the corresponding question (or answer).

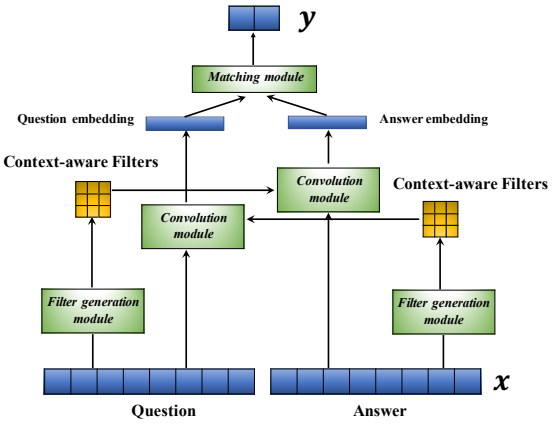


Figure 2: Schematic description of Adaptive Question Answering (AdaQA) model.

Compared to a Siamese CNN architecture (Bromley et al., 1994), our model selectively encapsulates the most important features for judgement prediction, removing less vital information. We then employ the question and answer representations $\boldsymbol{h}_q \in \mathbb{R}^{n_h}$, $\boldsymbol{h}_a \in \mathbb{R}^{n_h}$ as inputs to the matching module (where n_h is the dimension of question/answer embeddings). Following Mou et al. (2016), the matching function is defined as:

$$\boldsymbol{t} = [\boldsymbol{h}_q; \boldsymbol{h}_a; \boldsymbol{h}_q - \boldsymbol{h}_a; \boldsymbol{h}_q \odot \boldsymbol{h}_a] \quad (6)$$

$$p(\boldsymbol{y} = 1|\boldsymbol{h}_q, \boldsymbol{h}_a) = \text{MLP}(\boldsymbol{t}; \boldsymbol{\eta}') \quad (7)$$

where $-$ and $\odot$ denote an element-wise subtraction and element-wise product, respectively. $[\boldsymbol{h}_a; \boldsymbol{h}_b]$ indicates that $\boldsymbol{h}_a$ and $\boldsymbol{h}_b$ are stacked as column vectors. The resulting matching vector $\boldsymbol{t} \in \mathbb{R}^{4n_h}$ is then sent through an MLP layer (with sigmoid activation function and parameters $\boldsymbol{\eta}'$ to be learned) to model the desired conditional distribution $p(y_i = 1|\boldsymbol{h}_q, \boldsymbol{h}_a)$.

Notably, we share the weights of filter generating networks for both the question and answer, so that the model adaptivity for answer selection can be improved without an excessive increase in the number of parameters. All three modules in AdaQA model are jointly trained end-to-end. Note that the AdaQA model proposed can be readily adapted to other sentence matching tasks, such as paraphrase identification (see Section 5.2).

3.4 Connections to attention mechanism

The adaptive *context-aware* filter generation mechanism proposed here bears close resemblance to attention mechanism (Yin et al., 2016; Bahdanau et al., 2015; Xiong et al., 2017) widely adopted in the NLP community, in the sense that both methods intend to incorporate rich *contextual information* into text representations. However, attention is typically operated on top of the hidden units preprocessed by CNN or LSTM layers, and assigns different weights to each unit according to a *context vector*. By contrast, in our *context-aware* filter generation mechanism, the contextual information is inherently encoded into the convolutional filters, which directly interact with the input sentence during the convolution encoding operation. Notably, according to our experiments, the proposed filter generation module can be readily combined with (standard) attention mechanisms to further enhance the modeling expressiveness of CNN encoder.

Dataset	# train/ test	average #w	vocabulary
Yelp P.	560k/ 38k	138	25,709
DBpedia	560k/ 70k	56	21,666
WikiQA	20,360/ 2,352	7/ 26	10,000
SelQA	66,438/ 19,435	8/ 24	20,000
Quora	390k/ 5,000	13/ 13	20,000

Table 1: Dataset statistics.

4 Experimental Setup

Datasets We investigate the effectiveness of the proposed ACNN framework on both document classification and text sequence matching tasks. Specifically, we consider two large-scale document classification datasets: Yelp Reviews Polarity, and DBPedia ontology datasets (Zhang et al., 2015). For Yelp reviews, we seek to predict a binary label (positive or negative) regarding one review about a restaurant. DBpedia is extracted from Wikipedia by crowd-sourcing and is categorized into 14 non-overlapping ontology classes, including *Company*, *Athlete*, *Natural Place*, *etc*. We sample 15% of the training data as the validation set, to select hyperparameters for our models and perform early stopping. For sentence matching, we evaluate the AdaQA model on two datasets for open-domain question answering: WikiQA (Yang et al., 2015) and SelQA (Jurczyk et al., 2016). Given a question, the task is to rank the corresponding candidate answers, which, in the case of WikiQA, are sentences extracted from the summary section of a related Wikipedia article. To facilitate comparison with existing results (Yin et al., 2016; Yang et al., 2015; Shen et al., 2018b), we truncate the candidate answers to a maximum length of 40 tokens for all experiments on the WikiQA dataset. We also consider the task of paraphrase identification with the Quora Question Pairs dataset, with the same data splits as in (Wang et al., 2017b). A summary of all datasets is presented in Table 1.

Training Details For the document classification experiments, we randomly initialize the word embeddings uniformly within $[-0.001, 0.001]$ and update them during training. For the generated filters, we set the window size as $h = 5$, with $K = 100$ feature maps (the dimension of z is set as 100). For direct comparison, we employ the same filter shape/size settings as in our basic CNN implementation. A one-layer architecture is utilized for both the CNN baseline and the ACNN model, since we did not observe significant

performance gains with a multilayer architecture. The minibatch size is set as 128, and a dropout rate of 0.2 is utilized on the embedding layer. We observed that a larger dropout rate (*e.g.*, 0.5) will hurt performance on document classifications and make training significantly slower.

For the sentence matching tasks, we initialized the word embeddings with 50-dimensional Glove (Pennington et al., 2014) word vectors pretrained from Wikipedia 2014 and Gigaword 5 (Pennington et al., 2014) for all model variants. As for the filters, we set the window size as $h = 5$, with $K = 300$ feature maps. As described in Section 3.3, the vector $\boldsymbol{t}$, output from the matching module, is fed to the prediction layer, implemented as a one-layer MLP followed by the sigmoid function. We use Adam (Kingma and Ba, 2014) to train the models, with a learning rate of 3×10^{-4}. Dropout (Srivastava et al., 2014), with a rate of 0.5, is employed on the word embedding layer. The hyperparameters are selected by choosing the best model on the validation set. All models are implemented with TensorFlow (Abadi et al., 2016) and are trained using one NVIDIA GeForce GTX TITAN X GPU with 12GB memory.

Baselines For document classification, we consider several baseline models: (i) ngrams (Zhang et al., 2015), a bag-of-means method based on TFIDF representations built by choosing the 500,000 most frequent n-grams (up to 5-grams) from the training set and use their corresponding counts as features; (ii) small/large word CNN (Zhang et al., 2015): 6 layer word-based convolutional networks, with 256/1024 features at each layer, denoted as small/large, respectively; (iii) deep CNN (Conneau et al., 2016): deep convolutional neural networks with 9/17/29 layers. To evaluate the effectiveness of proposed AdaQA model, we compare it with several CNN-based sequence matching baselines, including Vanilla CNN (Jurczyk et al., 2016; Santos et al., 2017), attentive pooling networks (dos Santos et al., 2016), and ABCNN (Yin et al., 2016) (where an attention mechanism is employed over the two sentence representations).

Evaluation Metrics For document categorization and paraphrase identification tasks, we employ the percentage of correct predictions on the test set to evaluate and compare different models.

Model	Yelp P.	DBpedia
CNN-based Baseline Models		
ngrams*	4.36	1.37
ngrams TFIDF*	4.56	1.31
Small word CNN*	5.54	1.85
Large word CNN*	4.89	1.72
Self-attentive Embedding ‡	3.92	1.14
Deep CNN (9 layer)†	4.88	1.35
Deep CNN (17 layer)†	4.50	1.40
Deep CNN (29 layer)†	4.28	1.29
Our Implementations		
S-CNN	14.48	22.35
S-ACNN	**6.41**	**5.16**
M-CNN	4.58	1.66
M-ACNN	**3.89**	**1.07**

Table 2: Test error rates on document classification tasks (in percentages). S-*model* indicates that the *model* has one single convolutional filter, while M-*model* indicates that the *model* has multiple convolutional filters. Results marked with $*$ are reported by (Zhang et al., 2015), † are reported by (Conneau et al., 2016), and ‡ are reported by (Lin et al., 2017).

For the answer sentence selection task, mean average precision (MAP) and mean reciprocal rank (MRR) are utilized as the corresponding evaluation metrics.

5 Experimental Results

5.1 Document Classification

To explicitly explore whether our ACNN model can leverage the input-aware filter weights for better sentence representation, we perform a comparison between the basic CNN and ACNN models with only a *single* filter, which are denoted as S-CNN, S-ACNN, respectively (this setting may not yield best overall performance, since only a single filter is used, but it allows us to isolate the impact of adaptivity). As illustrated in Table 2, S-ACNN significantly outperforms S-CNN on both datasets, demonstrating the advantage of the filter-generation module in our ACNN framework. As a result, with only one convolutional filter and thus very limited modeling capacity, our S-ACNN model tends to be much more expressive than the basic CNN model, due to the flexibility of applying different filters to different sentences.

We further experiment on both ACNN and CNN models with *multiple* filters. The corresponding document categorization accuracies are presented in Table 2. Although we only use one convolution layer for our ACNN model, it already outperforms other CNN baseline methods with much deeper architectures. Moreover, our method ex-

Model	MAP	MRR
CNN-based Baseline Models		
bigram CNN + *Cnt**	0.6520	0.6652
Attentive Pooling Network	0.6886	0.6957
ABCNN	0.6921	0.7127
Our Implementations		
CNN	0.6752	0.6890
ACNN (self-adaptive)	0.6897	0.7032
AdaQA (one-way)	0.7005	0.7161
AdaQA (two-way)	0.7107	0.7304
AdaQA (two-way) + att.	**0.7325**	**0.7428**

Table 3: Results of our models on WikiQA dataset, compared with previous CNN-based methods.

Model	MAP	MRR
CNN-based Baseline Models		
CNN: *baseline**	0.8320	0.8420
CNN: *average* + *word**	0.8400	0.8494
CNN: *aver* + *emb**	0.8466	0.8568
CNN: *hinge_loss*‡	0.8758	0.8812
CNN-DAN‡	0.8655	0.8730
Our Implementations		
CNN	0.8644	0.8720
ACNN (self-adaptive)	0.8739	0.8801
AdaQA (one-way)	0.8823	0.8889
AdaQA (two-way)	0.8914	0.8983
AdaQA (two-way) + att.	**0.9021**	**0.9103**

Table 4: Results of our models on SelQA dataset, compared with previous CNN-based methods. Results marked with $*$ are from (Jurczyk et al., 2016), and marked with ‡ are from (Santos et al., 2017).

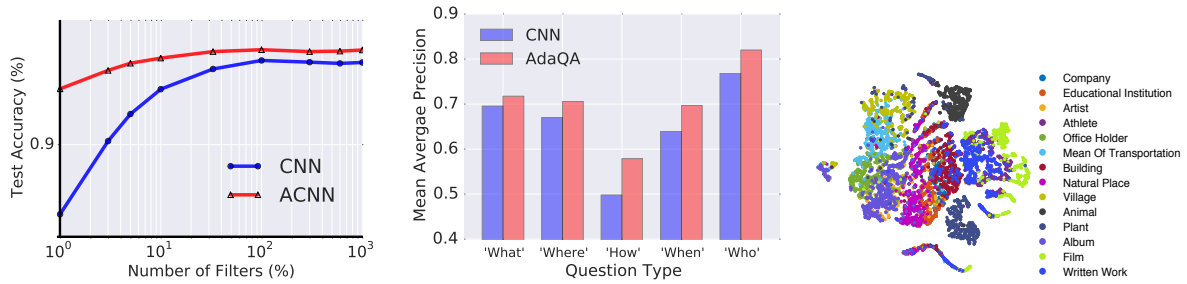
hibits higher accuracy than n-grams, which is a very strong baseline as shown in (Zhang et al., 2015). We attribute the superior performance of the ACNN framework to its stronger (adaptive) feature-extraction ability. Moreover, our M-ACNN also achieves slightly better performance than self-attentive sentence embeddings proposed in Lin et al. (2017), which requires significant more parameters than our method.

Effect of number of filters To further demonstrate that the performance gains in document categorization experiments originates from the improved adaptivity of our ACNN framework, we implement the basic CNN model with different numbers of filter sizes, ranging from 1 to 1000. As illustrated in Figure 3(a), when the filter size is larger than 100, the test accuracy of the standard CNN model does not show any noticeable improvement with more filters. More importantly, even with a filter size of 1000, the classification accuracy of the CNN is worse than that of the ACNN model with the filter number restricted to 100. Given these observations, we believe that the boosted categorization accuracy does come from the improved flexibility and thus better feature extraction of our ACNN framework.

5.2 Answer Sentence Selection

To elucidate the role of different parts (modules) in our AdaQA model, we implement several model variants for comparison: (*i*) a "vanilla" CNN model that independently encodes two sentence representations for matching; (*ii*) a self-adaptive ACNN-based model where the question/answer sentence generates adaptive filters only to convolve with the input itself; (*iii*) a one-way ACNN model where only the answer sentence representation is extracted with adaptive filters, which are generated conditioned on the question; (*iv*) a two-way AdaQA model as described in Section 2.4, where both sentences are adaptively encoded, with filters generated conditioned on the other sequence; (*v*) considering that the proposed filter generation mechanism is complementary to the attention layer typically employed in sequence matching tasks (see Section 3.4), we experiment with another model variant that combines the proposed *context-aware* filter generation mechanism with the multi-perspective attention layer introduced in (Wang et al., 2017b).

Tables 3 and 4 show experimental results of our models on WikiQA and SelQA datasets, along with other state-of-the-art methods. Note that the self-adaptive ACNN model variant, which generates filters only for the input itself (without any interactions before the top matching module), slightly outperforms the vanilla CNN Siamese model. Combined with the results in document categorization experiments, we believe that our ACNN framework, in its simplest form, can be utilized as a powerful feature extractor for transforming natural language sentences into fixed-length vectors. More importantly, our two-way AdaQA model exhibits superior results compared with the one-way variant as well as other CNN-based baseline models on the WikiQA dataset. This observation indicates that the *bidirectional* filter generation mechanism is strongly associated with the performance gains. While combined with the multi-perspective attention layers, adopted after the ACNN encoding layer, our two-way AdaQA model achieves even better performance. This suggests that the proposed strategy is complemen-



(a) Effect of filter number (b) Different question types (c) t-SNE visualization

Figure 3: Comprehensive study of the proposed ACNN framework, including (a) the number of filters (Yelp dataset), and (b) performance *vs* question types (WikiQA dataset), and (c) t-SNE visualization of learned filter weights (DBpedia dataset).

Model	Accuracy
Siamese-CNN	0.7960
Multi-Perspective-CNN	0. 8138
AdaQA (two-way)	0.8516
AdaQA (two-way) + att.	**0.8794**

Table 5: Results on the Quora Question Pairs dataset.

tary, in terms of the incorporation of rich contextual information, to the standard attention mechanism. The same trend is also observed on the SelQA dataset (as shown in Table 4), which is a much larger dataset than WikiQA.

Notably, our model yields significantly better results than an attentive pooling network and ABCNN (attention-based CNN) baselines. We attribute the improvement to two potential advantages of our AdaQA model: (*i*) for the two previous baseline methods, the interaction between question and answer takes place either before or after convolution. However, in our AdaQA model, the communication between two sentences is inherent in the convolution operation, and thus can provide the abstracted features with more flexibility; (*ii*) the *bidirectional* filter generation mechanism in our AdaQA model generates co-dependent representations for the question and candidate answer, which could enable the model to recover from initial local maxima corresponding to incorrect predictions (Xiong et al., 2017).

Paragraph Identification Considering that the proposed AdaQA model can be readily generalized to other text sequence matching problems, we further evaluate the proposed framework on the paraphrase identification task with the Quora question pairs dataset. To ensure a fair comparison, we employ the same data splits as in (Wang et al., 2017b). As illustrated in Table 5, our two-way AdaQA model again exhibits superior performances compared with basic CNN models (as reported in (Wang et al., 2017b)).

5.3 Discussion

Reasoning ability To associate the improved answer sentence selection results with the reasoning capabilities of our AdaQA model, we further categorize the questions in the WikiQA test set into 5 types containing: 'What', 'Where', 'How', 'When' or 'Who'. We then calculate the MAP scores of the basic CNN and our AdaQA model on different question types. Similar to the findings in (Miao et al., 2016), we observe that the 'How' question is the hardest to answer, with the lowest MAP scores. However, our AdaQA model improves most over the basic CNN on the 'How' type question, see Figure 3(b). Further comparing our results with NASM in (Miao et al., 2016), our AdaQA model (with a MAP score of 0.579) outperforms their reported 'How' question MAP scores (0.524) by a large margin, indicating that the adaptive convolutional filter-generation mechanism improves the model's ability to read and reason over natural language sentences.

Filter visualization To better understand what information has been encoded into our *context-aware* filters, we visualize one of the filters for sentences within the test set (on the DBpedia dataset) with t-SNE. The corresponding results are shown in Figure 3(c). It can be observed that the filters for documents with the same label (ontology) are grouped into clusters, indicating that for different types of document, ACNN has leveraged distinct convolutional filters for better feature extraction.

6 Conclusions

We presented a *context-aware* convolutional filter-generation mechanism, introducing a meta network to adaptively produce a set of input-aware filters. In this manner, the filter weights vary from sample to sample, providing the CNN encoder network with more modeling flexibility and capacity.

This framework is further generalized to model question-answer sentence pairs, leveraging a two-way feature abstraction process. We evaluate our models on several document-categorization and sentence matching benchmarks, and they consistently outperform the standard CNN and attention-based CNN baselines, demonstrating the effectiveness of our framework.

Acknowledgments This research was supported in part by DARPA, DOE, NIH, ONR and NSF.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *NIPS*, pages 737–744.

William W Cohen and Yoram Singer. 1999. Context-sensitive learning methods for text categorization. *TOIS*, 17(2):141–173.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12(Aug):2493–2537.

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

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. *ICML*.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. 2016. Dynamic filter networks. In *NIPS*.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2016. Learning generic sentence representations using convolutional neural networks. *arXiv preprint arXiv:1611.07897*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *ICML*.

David Ha, Andrew Dai, and Quoc V Le. 2017. Hypernetworks. *ICLR*.

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

Yunho Jeon and Junmo Kim. 2017. Active convolution: Learning the shape of convolution for image classification. *CVPR*.

Tomasz Jurczyk, Michael Zhai, and Jinho D Choi. 2016. Selqa: A new benchmark for selection-based question answering. In *ICTAI*, pages 820–827. IEEE.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *EMNLP*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *ICLR*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Dynamic compositional neural networks over tree structure. *IJCAI*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICML*, pages 1727–1736.

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

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*.

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

Cicero Nogueira dos Santos, Kahini Wadhawan, and Bowen Zhou. 2017. Learning loss functions for semi-supervised learning via discriminative adversarial networks. *arXiv preprint arXiv:1707.02198*.

Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Lawrence Carin, and Ricardo Henao. 2018a. Nash: Toward end-to-end neural architecture for generative semantic hashing. In *ACL*.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018b. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *ACL*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2017. Deconvolutional latent-variable model for text sequence matching. *arXiv preprint arXiv:1709.07109*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia Sa. 2018. Speeding up context-based sentence representation learning with non-autoregressive convolutional decoding. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 69–78.

Shuai Tang and Virginia R de Sa. 2018. Multi-view sentence representation learning. *arXiv preprint arXiv:1805.07443*.

Sebastian Thrun and Lorien Pratt. 2012. *Learning to learn*. Springer Science & Business Media.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. *ICLR*.

Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. 2017a. Topic compositional neural language model. *arXiv preprint arXiv:1712.09783*.

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

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. *COLING*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. *ICLR*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*, pages 2013–2018.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *TACL*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.

Xinyuan Zhang, Yitong Li, Dinghan Shen, and Lawrence Carin. 2018. Diffusion maps for textual network embedding. *arXiv preprint arXiv:1805.09906*.