

# Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers

- <https://arxiv.org/abs/1602.00367> (<https://arxiv.org/abs/1602.00367>)

## Abstract

### 연구의의

기존 문서분류는 Word-level에서 이루어졌습니다. 이 연구는 Character-level input 에 대하여, encoding 과정에서 CNN과 RNN을 이용한 NN 아키텍처를 구성했습니다.

### 기존 Character level 연구와의 차이점

RNN layer 의 사용은 훨씬 적은 파라미터를 활용하여 성능을 만들어 냈습니다.

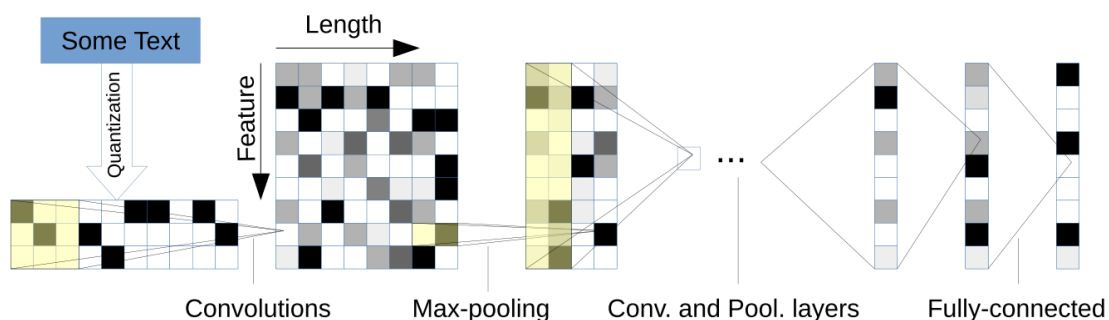
## Introduction

### Word level의 한계점

- Statistically inefficient
  - 각각의 토큰들은 분리되고 동일한 수의 파라미터를 가지게 됩니다. 하지만 많은 단어들은 같은 root, prefix, suffix를 공유합니다.
- out-of-vocabulary word
  - 단어장 외에 있는 단어들은 token으로 인식이 안됩니다. Word-level에서는 unknown token으로 분류하게 됩니다. 하지만 이러한 문제는 훈련된 모델을 다른 도메인에서 사용할 시에, 문제가 됩니다.
- Missing spell
  - social network 같은곳에 있는 축약어나, 오타등을 잘 못잡게 됩니다.
- reference
  - <https://www.edwith.org/deeppnl/lecture/29225/> (<https://www.edwith.org/deeppnl/lecture/29225/>)

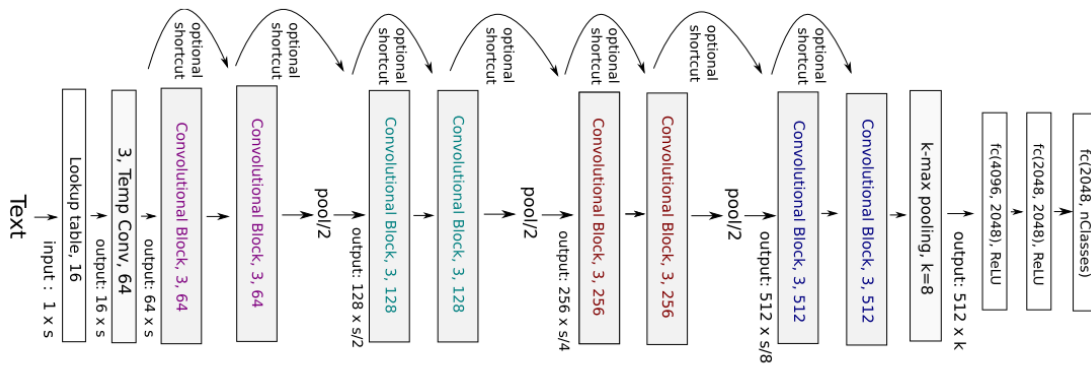
### 기존의 Character level 방법론

- Character-level Convolutional Networks for Text Classification (Zhang et al., 2015)
  - 다중 convolution layer를 구성하고, max-pooling을 하였다. 각 layer는 우선적으로 feature를 small 부터 뽑아낸다. 마지막 컨볼루션 레이어는 벡터의 모양을 flatten하게 activation 시킨다. 그 다음에 fully connected layer로 연결한다.
  - 한계점 : 각 convolution layer의 Receptive field가 작아서(3,7), 네트워크의 layer가 input sequence의 long-term dependency를 고려하여 커져야합니다.



- Very deep convolutional network

- 6개의 convolution layer 와 2개의 fully-connected layer의 연결로 구성하였다.
- 한계점 : 파라미터 수의 급격한 증가가 있다.



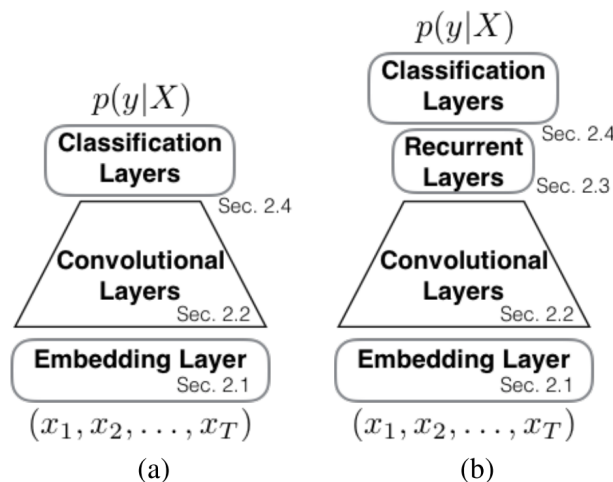
## 이 논문의 포인트

위에서 언급한 Character-level sequence의 비효율을 해결하기위해서 이 논문에서는 CNN과 RNN의 결합을 소개한다.이 결합 모델은 single recurrent layer 와 CNN 으로 구성되어 있다. RNN의 경우 GRU또는 LSTM의 사용을 통해 long-term dependency를 해결한다. 또한 CNN이 더 적은 컨볼루션 레이어를 사용하게 한다.

## Evaluation

Character-level Convolutional Networks for Text Classification 에서 제시한 8개의 large-scale document classification task를 해보고자한다. 8개의 문서분류 문제를 풀어보고, Character-level convolutional networks for text classification와 성능 비교해보자고 한다. 더 적은 모델을 통해서, 비슷한 성능을 내는것을 목표로 한다.

## Basic Building Blocks: Neural Network Layers



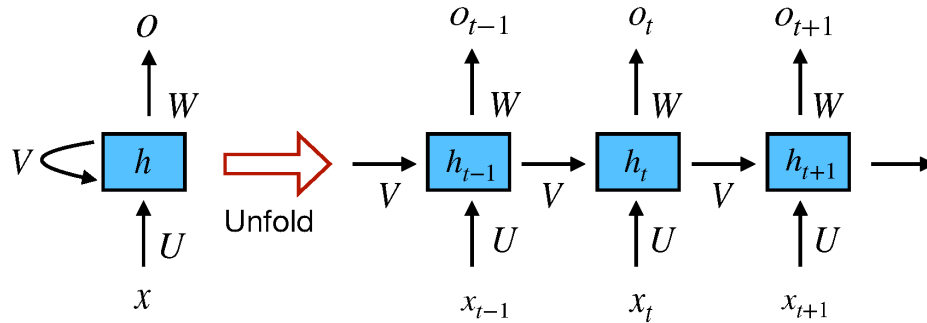
### Embedding Layer

- $e_t = Wx_t$
- 각 문서들은 one-hot vector의 sequence로 나타냅니다.
- 학습된 weight matrix가 W입니다.

### Convolution Layer

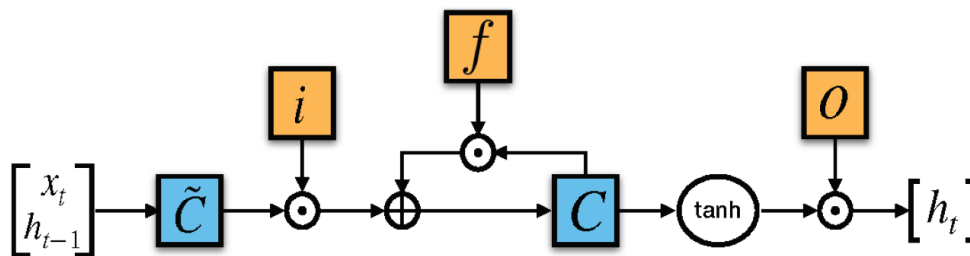
### Recurrent Layer

- 이전 hidden step의 input을 받아서 다음번 step의 output으로 내보냅니다.
- Recursive function
- $h_t = f(x_t, h_{t-1})$
- $h_t = \tanh(W_x x_t + U_h h_{t-1})$ ,
- NLP 에서 RNN은 문장의 정보를 시간의 순서에 따라 압축 할 수 있습니다.

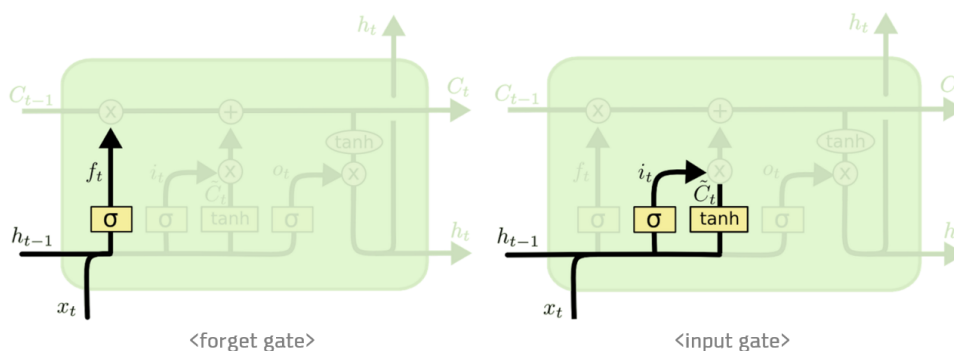


## LSTM & GRU

- RNN은 문장이 많이 길어질 수록 고정된 메모리에 압축된 정보를 담아야 하기 때문에, 앞에서 학습한 정보를 잊습니다. 이는 곧 정보의 손실을 뜻합니다.
- LSTM 은 이를 해결하기 위해 cell state를 RNN에 추가합니다
- input,output, forget gates and candidate memory cell 됩니다.
- $f_t = \sigma(W_f x_t + U_f h_{t-1})$ 
  - ‘과거 정보를 잊기’를 위한 게이트입니다.  $h_{t-1}$  과  $x_t$ 를 받아 시그모이드를 취해준 값
- $\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1})$
- $i_t = \sigma(W_i x_t + U_i h_{t-1})$ 
  - ‘현재 정보를 기억하기’ 위한 게이트입니다.
- $o_t = \sigma(W_o x_t + U_o h_{t-1})$

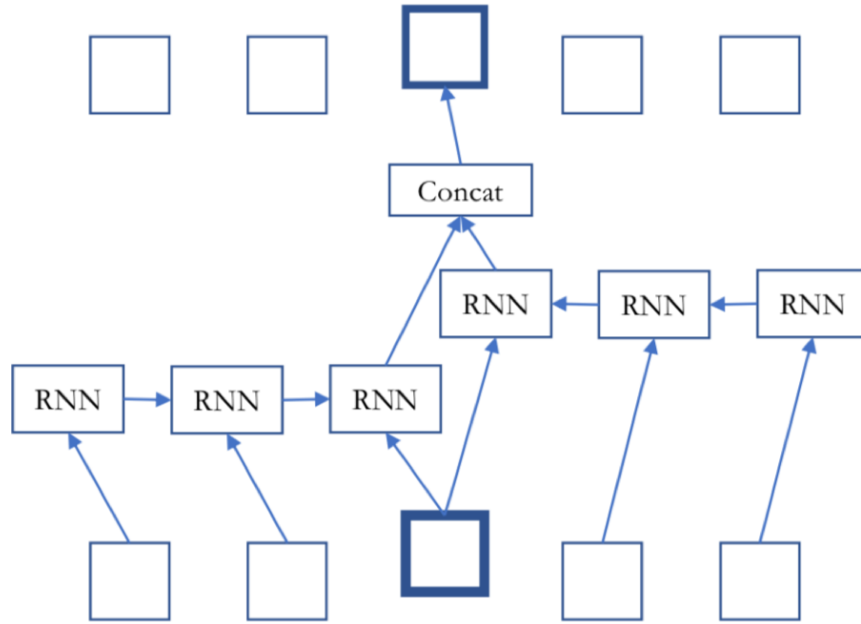


(1) Long Short-Term Memory



## Bidirectional Recurrent Layer

- 또한 토큰을 순차적으로 하나씩 읽어야 하기 때문에, 훈련 할때 속도가 기타 네트워크 보다 느립니다.
- Bidirection RNN 은 양쪽 방향으로 전파시킨다음에 concat 합니다.



## Classification Layer

- Classification 을 위해 logistic regression 을 사용하였습니다.
- 위 Convolution layer 와 Recuurent layer 의 결과가 문장의 길이  $D$  에 따른 다양한 길이를 가진 sequence를 반환해도 fixed-dimension의 vector를 얻을수 있습니다.

### reference

- <https://nlpoverview.com/#4> (<https://nlpoverview.com/#4>)
- <https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/> (<https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/>)
- <https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/> (<https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/>)

## Model Setting

Model	Embedding Layer Sec. 2.1		Convolutional Layer Sec. 2.2				Recurrent Layer Sec. 2.3	
	$ V $	$d$	$d'$	$r$	$r'$	$\phi$	$d'$	
C2R1DD	96	8	$D$	5,3	2,2	ReLU	$D$	
C3R1DD				5,5,3	2,2,2			
C4R1DD				5,5,3,3	2,2,2,2			
C5R1DD				5,5,3,3,3	2,2,2,1,2			

**Table 2:** Different architectures tested in this paper.

## Model Description

- Vocabulary 은 96개의 단어로, 대소문자, 숫자, !, 공백까지 포함합니다.

- 단어 embedding size는 8 입니다.
- Convolution layer의 경우 2~5개를 배치합니다.
- $d' = 128$  짜리 filter를 적용합니다. AG, Yahoo의 경우 1024 filter로 실험을 추가해봅니다.
  - Filter 는 detect하고자 하는 feature에 대한 내용(128 char = 문장길이?)
- Receptive field size는  $r$ 로 설정하며, depth에 따라 5,3이 됩니다.
  - Receptive Field는 Filter size로 진행하면서 detect되는 실제 값
- Max pooling size  $r'$ 은 2로 합니다.
- ReLU 씁니다.
- Recurrent layer의 경우, single layer에 bidirectional LSTM을 모든 모델에 적용합니다. Hidden state는 128의 차원을 가집니다.
- Dropout 을 사용합니다.
  - Convolution layer 와 Recurrent Layer 모두 사용합니다.
- AdaDelta with  $\rho = 0.95$ ,  $\epsilon = 10^{-5}$
- batch size of 128.
- Gradient 가 L2 norm에서 5보다 커질때, rescale 합니다.
- Early stopping 합니다.
  - patience value를 도입합니다.
  - 현재 상태에서 가장낮은 validation loss가 0.5보다 낮을시, patience 를 2배로 확장합니다.
  - epoch가 patience value 보다 커지면 학습을 멈춥니다.

## Experiment Setting

Data set	Classes	Task	Training size	Test size
AG's news	4	news categorization	120,000	7,600
Sogou news	5	news categorization	450,000	60,000
DBPedia	14	ontology classification	560,000	70,000
Yelp review polarity	2	sentiment analysis	560,000	38,000
Yelp review full	5	sentiment analysis	650,000	50,000
Yahoo! Answers	10	question type classification	1,400,000	60,000
Amazon review polarity	2	sentiment analysis	3,600,000	400,000
Amazon review full	5	sentiment analysis	3,000,000	650,000

**Table 1:** Data sets summary.

- Task 200,000 ~ 4,000,000개의 문서를 sentiment analysis, ontology classification, Question type classification, news categorization 등을 준비했다.

## Result

Data set	# Ex. # Cl.		Our Model			(Zhang et al., 2015)		
			Network	# Params	Error (%)	Network	# Params	Error (%)
AG	120k	4	C2R1D1024	20M	8.39/ <b>8.64</b>	C6F2D1024	27M	-/9.85
Sogou	450k	5	C3R1D128	.4M	4.82/ <b>4.83</b>	C6F2D1024*	27M	-/4.88
DBPedia	560k	14	C2R1D128	.3M	1.46/ <b>1.43</b>	C6F2D1024	27M	-/1.66
Yelp P.	560k	2	C2R1D128	.3M	5.50/5.51	C6F2D1024	27M	-/ <b>5.25</b>
Yelp F.	650k	5	C2R1D128	.3M	38.00/ <b>38.18</b>	C6F2D1024	27M	-/38.40
Yahoo A.	1.4M	10	C2R1D1024	20M	28.62/ <b>28.26</b>	C6F2D1024*	27M	-/29.55
Amazon P.	3.6M	2	C3R1D128	.4M	5.64/5.87	C6F2D256*	2.7M	-/ <b>5.50</b>
Amazon F.	3.0M	5	C3R1D128	.4M	40.30/40.77	C6F2D256*	2.7M	-/ <b>40.53</b>

- $CCRRFFDD$  refers to a network with  $C$  convolutional layers,  $R$  recurrent layers,  $F$  fully-connected layers and  $D$  dimensional feature vectors
- Classes 의 수가 증가할수록, 우리의 모델은 더 좋은 퍼포먼스를 낼 수 있습니다.
- 제안한 모델은 더 적은 pooling layers는 더 적은 pooling layer를 사용하면서 구체적인 정보를 더 보존합니다.
- 적은 데이터 사이즈에서 더 잘 작동합니다.
- 컨볼루션 레이어는 2~3개일때 가장 잘 작동합니다
- 필터갯수의 증가는 파라미터 갯수 증가에 비해 성능개선이 약합니다

## Conclusion

제안한 모형은 class 숫자 증가에 따라, 데이터셋이 적을수록, 그리고 컨볼루션 레이어가 2~3개일때 가장 잘 작동했습니다.

In [ ]: