CNN for Text Categorization

Chang Liu   Ning Zhang

Target

- Study CNN on Text Categorization
- Exploit the 1D structure(word order) of text data for prediction
- Deploy the bag-of-word conversion in the convolutional layer
- Combine multiple convolutional layers is explored for accuracy prediction

CNN for Image

- convolution layer, pooling layer
- activation function σ
- Weight **W**
- Bias **b**

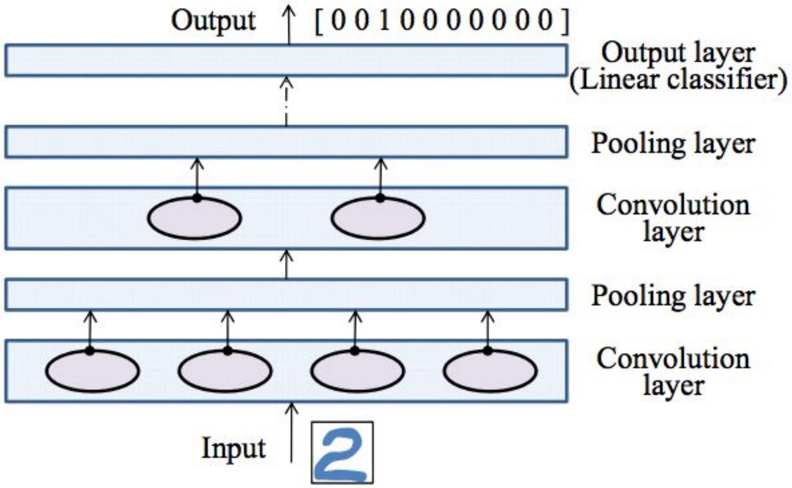


Figure 1: Convolutional neural network.

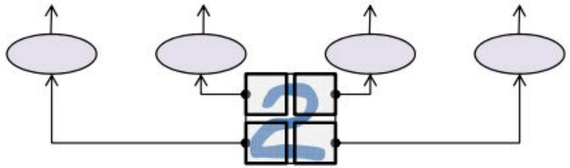


Figure 2: Convolution layer for image. Each computation unit (oval) computes a non-linear function $\boldsymbol{\sigma}(\mathbf{W} \cdot \mathbf{r}_\ell(\mathbf{x}) + \mathbf{b})$ of a small region $\mathbf{r}_\ell(\mathbf{x})$ of input image $\mathbf{x}$, where weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$ are shared by all the units in the same layer.

CNN for Text

- **seq-CNN**: straightforward adaption of CNN from image to text
- **bow-CNN**: employ bow conversion in convolution layer
- seq-CNN is better in sentiment classification(IMDB)
- bow-CNN is better in topic classification(RCV1)

methods	**IMDB**	**Elec**	**RCV1**
SVM bow3(30K)	10.14	9.16	10.68
SVM bow1(all)	11.36	11.71	10.76
SVM bow2(all)	9.74	9.05	10.59
SVM bow3(all)	9.42	8.71	10.69
NN bow3(all)	9.17	8.48	10.67
NB-LM bow3(all)	8.13	8.11	13.97
bow-CNN	8.66	8.39	9.33
seq-CNN	8.39	7.64	9.96
seq2-CNN	8.04	7.48	-
seq2-bow$n$-CNN	**7.67**	**7.14**	-

Table 2: Baseline method with error rate(%).

seq-CNN

- Document **D**=(w_1, w_2, ...) with vocabulary **V**
- treat each word as a pixel
- treat **D** as an image of **|D|x1** with **|V|** channels
- Example
  - V = {“don’t”, “hate”, “I”, “it”, “love”}
  - D = “I love it”
  - vector x for D: x=[00100|00001|00010]

seq-CNN

- represent region with concatenation of pixels
- **p|V|** dimensional region vector, where p is fixed in advance region size
- Example:
    - p=2, stride=1
    - “I love”, “love it”

$$\mathbf{r}_0(\mathbf{x}) = \left[\begin{array}{c} 0 \\ 0 \\ \mathbf{1} \\ 0 \\ 0 \\ — \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \end{array}\right] \begin{array}{c} \text{don't} \\ \text{hate} \\ \textbf{I} \\ \text{it} \\ \text{love} \\ \\ \text{don't} \\ \text{hate} \\ \text{I} \\ \text{it} \\ \textbf{love} \end{array} \qquad \mathbf{r}_1(\mathbf{x}) = \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ — \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \end{array}\right] \begin{array}{c} \text{don't} \\ \text{hate} \\ \text{I} \\ \text{it} \\ \textbf{love} \\ \\ \text{don't} \\ \text{hate} \\ \text{I} \\ \textbf{it} \\ \text{love} \end{array}$$

bow-CNN

- **p** is large, r(x) is high-dimensional.
- p|V| is too large, then mod becomes very complex
- Use bow conversion to represent region vector
- **p|V|** ----> **|V|** dimensions

$$\mathbf{r}_0(\mathbf{x}) = \begin{array}{c} \left[\begin{array}{c} 0 \\ 0 \\ \mathbf{1} \\ 0 \\ \mathbf{1} \end{array}\right] \end{array} \begin{array}{c} \text{don't} \\ \text{hate} \\ \textbf{I} \\ \text{it} \\ \textbf{love} \end{array} \qquad \mathbf{r}_1(\mathbf{x}) = \begin{array}{c} \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{array}\right] \end{array} \begin{array}{c} \text{don't} \\ \text{hate} \\ \text{I} \\ \textbf{it} \\ \textbf{love} \end{array}$$

Baseline

- **seq2-bown-CNN** achieve the best accuracy!
- seq vs bow
- layer vs more layers

methods	IMDB	Elec	RCV1
SVM bow3(30K)	10.14	9.16	10.68
SVM bow1(all)	11.36	11.71	10.76
SVM bow2(all)	9.74	9.05	10.59
SVM bow3(all)	9.42	8.71	10.69
NN bow3(all)	9.17	8.48	10.67
NB-LM bow3(all)	8.13	8.11	13.97
bow-CNN	8.66	8.39	9.33
seq-CNN	8.39	7.64	9.96
seq2-CNN	8.04	7.48	-
seq2-bow$n$-CNN	**7.67**	**7.14**	-

Table 2: Baseline method with error rate(%).

More convolutional Layers

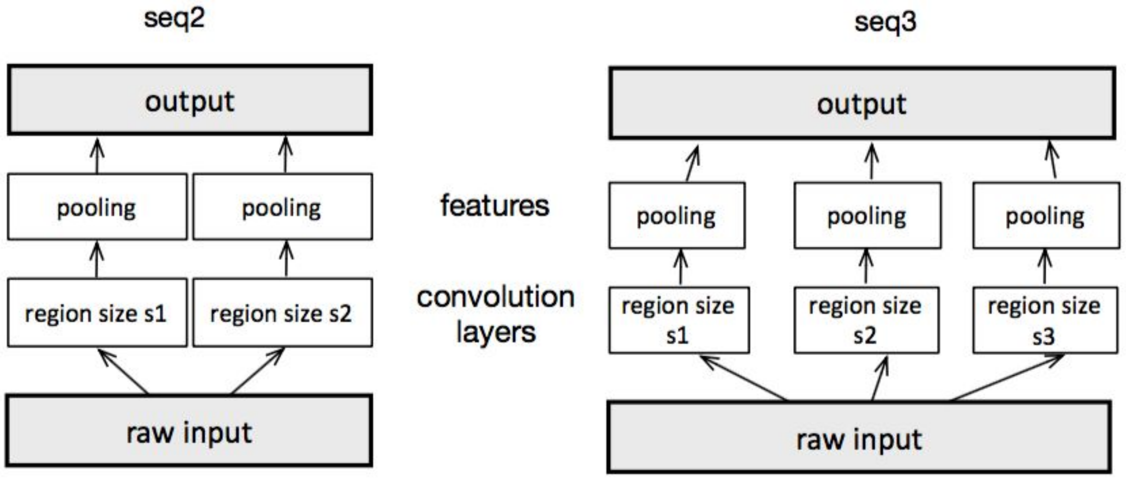


Figure 7: architecture in seq2 vs seq3

More convolutional Layers

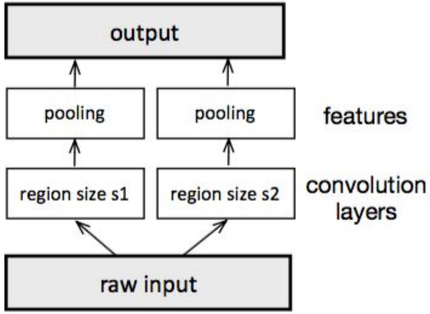
- In most cases, not working well.
- With bow feature layers, it slightly works better.
- Simply increasing the convolutional will not work.

methods	IMDB	Elec	RCV1
NB-LM bow3(all)	8.13	8.11	13.97
seq-CNN	8.39	7.64	9.96
seq2-CNN	8.04	7.48	-
seq3-CNN	**8.12**	**7.54**	-
seq2-bown-CNN	7.67	7.14	-
seq3-bown-CNN	**7.61**	**7.2**	-
seq2-CNN-3k	**7.88**	-	-
seq3-CNN-3k	**8.04**	-	-

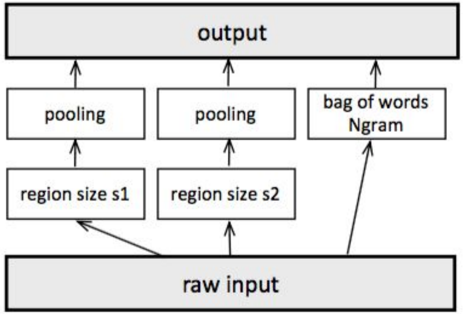
Table 3: error rate(%) with different convolutional layers

More bow feature layers

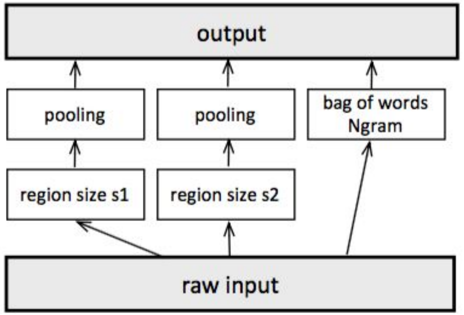
seq2_CNN



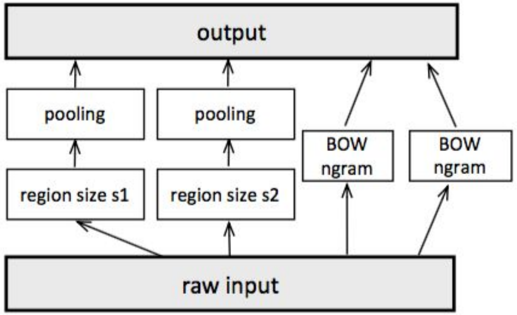
seq2_bown-CNN



seq2_bown-CNN



seq2_bown_bown-CNN



More bow feature layers

- Not so obvious using more bown layer
- still better than seq3-CNN

methods	**IMDB**	**Elec**	**RCV1**
NB-LM bow3(all)	8.13	8.11	13.97
bow-CNN	8.66	8.39	9.33
seq-CNN	8.39	7.64	9.96
seq2-CNN	8.04	7.48	-
seq2-bow$n$-CNN	**7.67**	**7.14**	-
seq2-bow$n$-bow$n$-CNN	**7.72**	-	-
seq3-CNN	**8.12**	7.54	-

More powerful feature

- Using 1,2,3,4-gram instead of 1,2,3-gram
- Works much better in IMDB
- IMDB vs. Elec (positive/negative with star rating)

methods	**IMDB**	**Elec**	**RCV1**
NB-LM bow3(all)	8.13	8.11	13.97
seq2-bow$n$-CNN	**7.67**	**7.14**	-
seq2-bow$n$-4CNN	**7.48**	**7.2**	-

Conclusion

- Using convolutional layer to present 1D word order can obviously improve the accurate prediction.
- Under the 2-conv with bow feature layer, exploiting more convolutional layers will not affect the prediction, word order should not be overrated.
- Still bow feature conversion in convolutional layer can help to better predict if combined with the previous structure.