Convolutional Neural Networks for Sentence Classification

Yoon Kim, New York University

AI Robotics NLP STUDY

Dahae Jeon

Convolutional Neural Networks for Sentence Classification

이 논문의 핵심

- Sentence Classification Using Word Vector and CNN

- RNN과 마찬가지로 context 정보를 보존

- CNN을 사용하기 전에 Word Vector를 word2vec으로 학습해서 사용하니까 성능 개선에 효과적이다.

Model Architecture

- Google News 1000억개를 word2vec 모델로 학습시켜 **word vector**를 구축하여 사용

- **n** : 단어 갯수
- **k** : 각 단어의 차원

  word2vec으로 표현된 단어 vector
- **h** : window size

  한번에 단어를 몇 개씩 볼 것인지 정함

- 한개의 문장은 **n * k** matrix

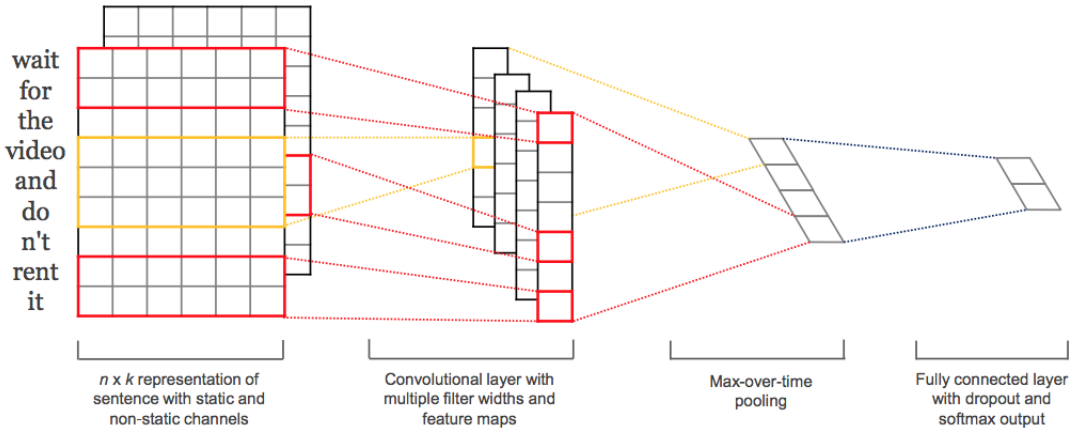


Figure 1: Model architecture with two channels for an example sentence.

Model Architecture

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$$

⊕ : concatenation operator

i 번째 단어의 word embedding vector 를 x_i

Word embedding vector들을 이어 붙여서 matrix 형태의 input을 만듦

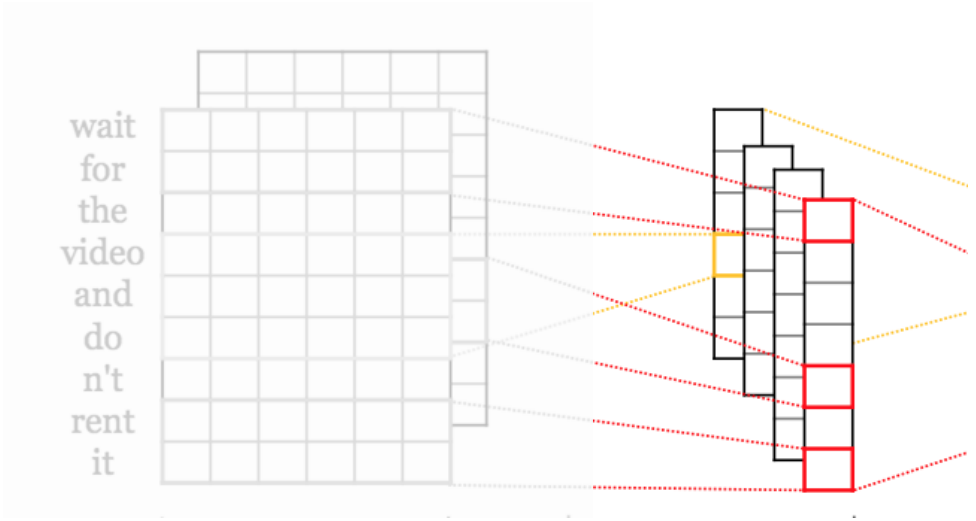
$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$$

- Convolution 연산
- 다양한 필터 사이즈로 Convolution Layer 만듦 (Filter Size: 3, 4, 5)
- 3가지 feature map 생성
- 필터사이즈를 여러 개 사용하면 다양한 길이의 단어 조합 패턴을 찾는 것이 가능
- W: filter weight
- f: 비선형 활성함수 사용 (ex: ReLU)

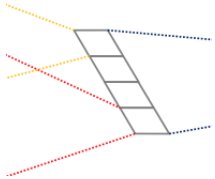
Model Architecture

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}]$$

- 모든 word window에 filter 적용하여 feature map 계산
- 한 개 문장에서 여러 개의 값을 구할 수 있다.
- Filter windows (h) of 3, 4, 5 with 100 feature maps each



$$\hat{c} = \max\{\mathbf{c}\}$$



- Feature map을 max-overtime pooling.
- 각 filter 별로 가장 중요한 특성을 추출.
- 문장마다 단어 개수가 다르고, 그러면 문장마다 feature map의 개수가 달라지는데 모든 문장마다 하나의 값을 갖게끔 feature map vector 중 가장 큰 값 하나만 사용하는 것.

Model Architecture

Fully Connected Layer와 Softmax Layer

- **Dropout, softmax 적용**
- 학습과정에는 dropout 사용. Test에서는 사용하지 않음.

- **"dropout rate (p) of 0.5, l2 constraint (s) of 3, and mini-batch size of 50."**
  - Dropout rate = 0.5
  - L2 regularization = 3
  - mini-batch size = 50
  - epoch = 25

- Static Layer / Non Static Layer 둘다 적용

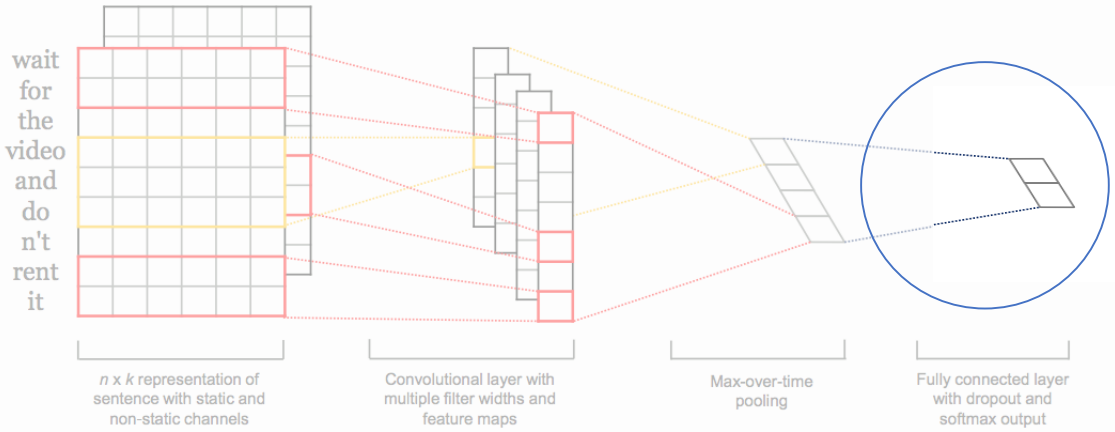
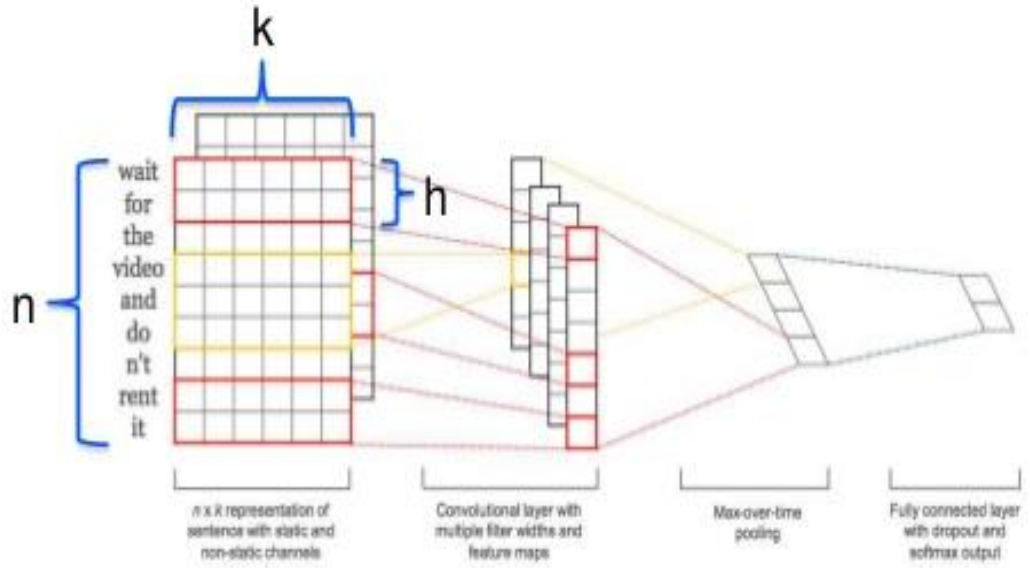
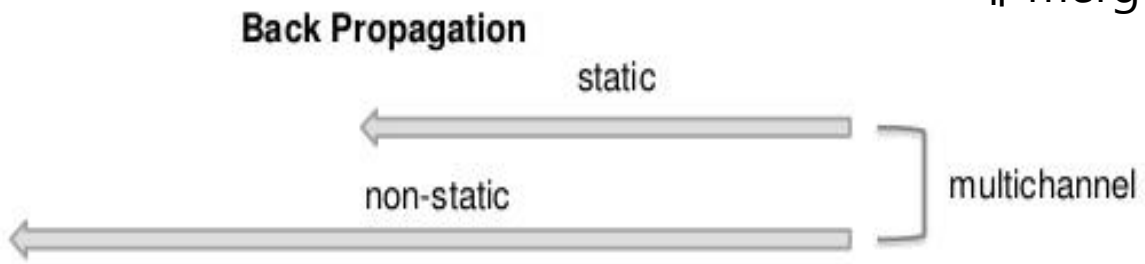


Figure 1: Model architecture with two channels for an example sentence.

Model Architecture

Static Layer / Non Static Layer





Static Layer
- Word vector를 Input으로 취급
- 이미 학습된 word vector를 사용
- Input으로 취급되므로 변하지 않음
- 위치만 보기 때문에 'bad'와 'good'이 비슷한 단어인 것으로 나옴

Non Static Layer
- Word vector도 학습 대상으로 함

Multichannel
- Static, Non Static 각각 채널 구성하여 맨 마지막에 merge 하여 학습

https://www.slideshare.net/keunbongkwak/convolutional-neural-networks-for-sentence-classification

Result

Page 5, Table 3

	Most Similar Words for	
	Static Channel	Non-static Channel
bad	*good* *terrible* *horrible* *lousy*	*terrible* *horrible* *lousy* *stupid*
good	*great* *bad* *terrific* *decent*	*nice* *decent* *solid* *terrific*
n't	*os* *ca* *ireland* *wo*	*not* *never* *nothing* *neither*
!	*2,500* *entire* *jez* *changer*	*2,500* *lush* *beautiful* *terrific*
,	*decasia* *abysmally* *demise* *valiant*	*but* *dragon* *a* *and*

Table 3: Top 4 neighboring words—based on cosine similarity—for vectors in the static channel (left) and fine-tuned vectors in the non-static channel (right) from the multichannel model on the SST-2 dataset after training.

- Non Static으로 학습 시킨 것이 단어의 의미는 더 잘 파악
- Static만 사용한 경우 good과 bad가 비슷한 의미의 단어가 됨
- Non static의 경우 bad와 유사한 의미의 단어가 terrible, horrible 등으로 알맞게 나옴.

Result

Page 4, Table 2

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

- CNN-rand
Word2vec 사용 안하니까 성능 가장 안좋음

- CNN-static
성능 좋은 편 word2vec 효과 있음

- CNN-non-static
Static에 비해 유의미한 효과 있는지는 모름

- CNN-multichannel
둘다 사용하니 향상 있으나 static만 사용한 것에 비해서 유의미한 효과가 있는지는 의문

참고 및 인용 문헌

Review

- https://www.slideshare.net/keunbongkwak/convolutional-neural-networks-for-sentence-classification
- http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/
- https://jisoo-coding.tistory.com/9
- http://taewan.kim/post/cnn/
- https://elapser.github.io/nlp/2018/02/05/Convolutional-Neural-Networks-for-Sentence-Classification.html
- https://youtu.be/IRB2vXSet2E?list=PLlMkM4tgfjnJhhd4wn5aj8fVTYJwIpWkS
- https://jamiekang.github.io/2017/06/12/cnn-for-sentence-classification/

Yoon Kim

- http://cbnaodkpfinfiipjblikofhlhlcickei/src/pdfviewer/web/viewer.html?file=http://www.people.fas.harvard.edu/~yoonkim/data/sent-cnn-slides.pdf
- http://cbnaodkpfinfiipjblikofhlhlcickei/src/pdfviewer/web/viewer.html?file=http://www.people.fas.harvard.edu/~yoonkim/data/sent-cnn.pdf