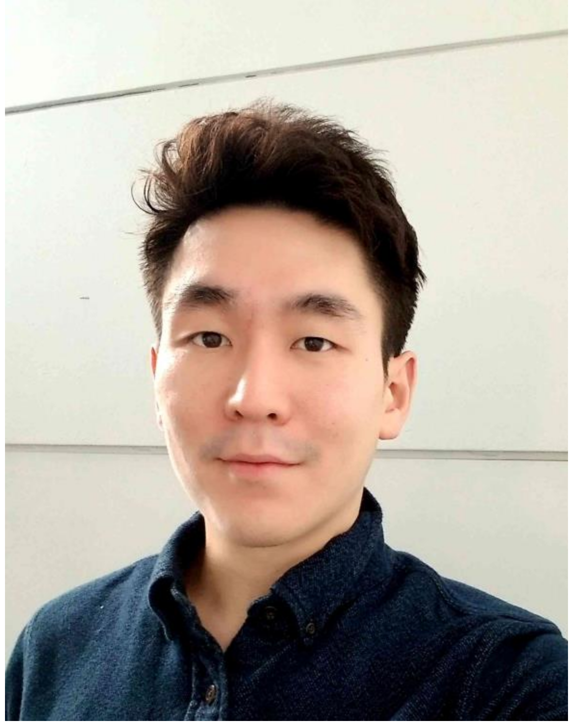
Convolutional Neural Networks for Sentence Classification, Yoon Kim(2014)

집현전 초급반 유영재
(u_yeongjae@naver.com)

INDEX

Author



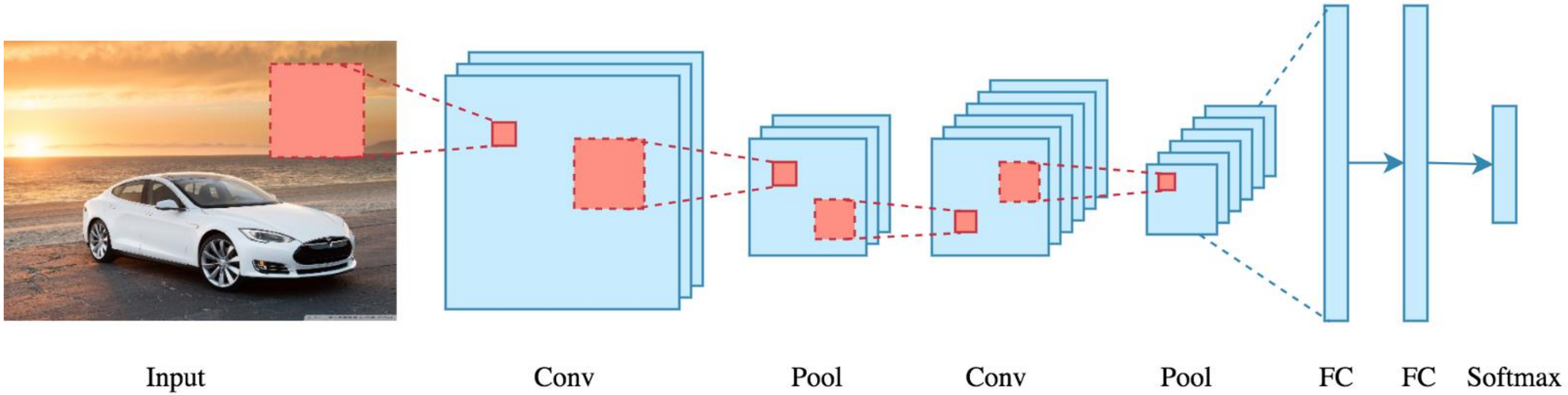
Yoon Kim

- Research scientist at MIT-IBM Watson AI Lab
- Assistant Professor at MIT EECS department

CNN for Sentence Classification

- Paper - arxiv.org/pdf/1408.5882
- Code - github.com/yoonkim/CNN_sentence

Convolutional Neural Networks



Abstract

We report on a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks. We show that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks. Learning task-specific vectors through fine-tuning offers further gains in performance. We additionally propose a simple modification to the architecture to allow for the use of both task-specific and static vectors. The CNN models discussed herein improve upon the state of the art on 4 out of 7 tasks, which include sentiment analysis and question classification.

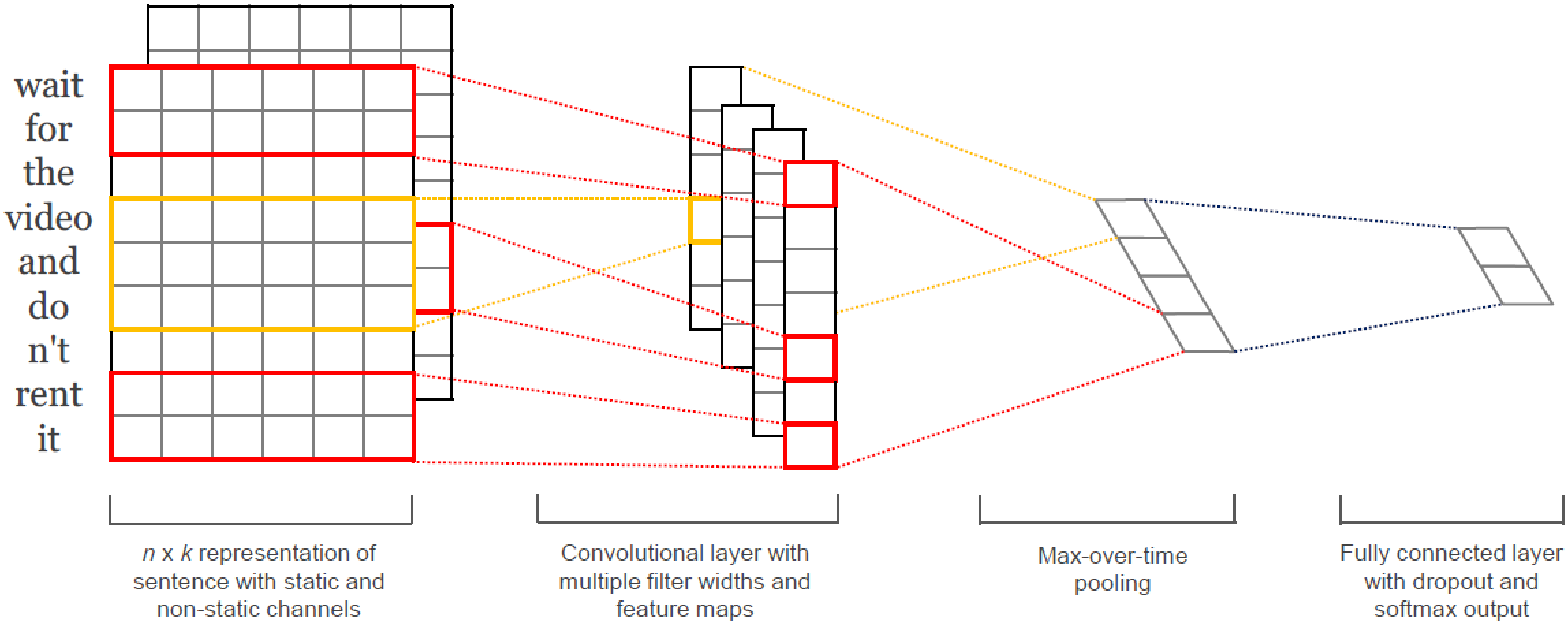
1. pre-trained word vectors와 CNN을 활용하여 sentence-level classification task를 수행

2. fine-tuning을 통해 word vectors를 task-specific 하게하여 성능을 향상

3. task specific word vectors와 static word vectors를 활용한 아키텍쳐를 제안

4. 실험에 사용된 7개 중 4개의 benchmark dataset에 대하여 SOTA를 기록

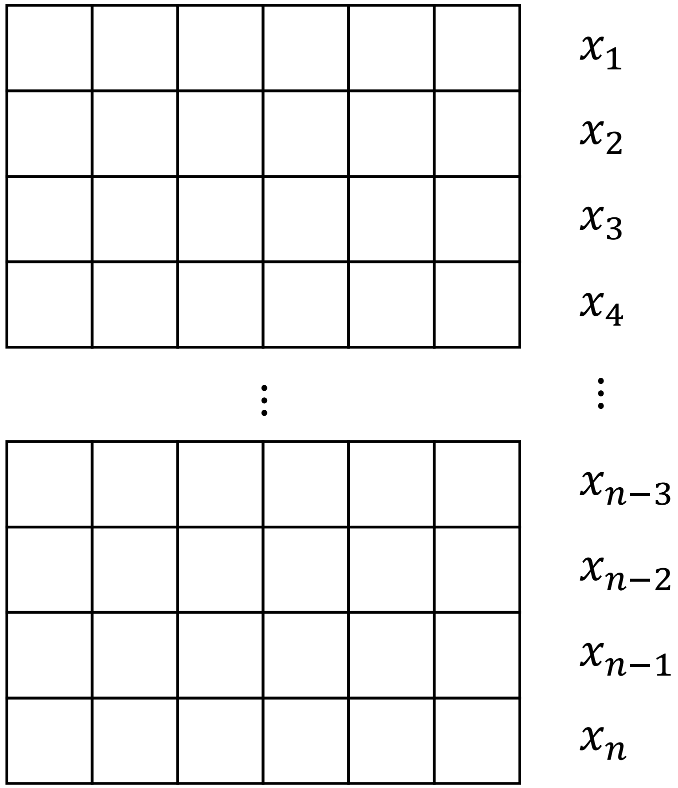
Model

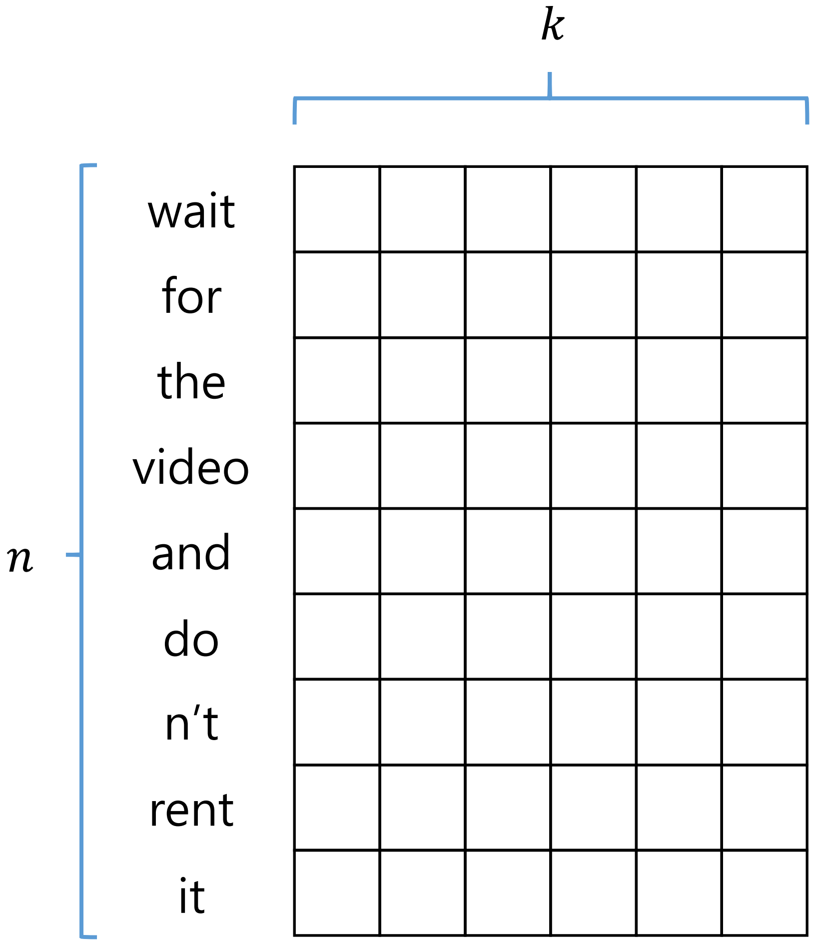


Convolutional Neural Networks for Sentence Classification, Yoon Kim(2014)

Model

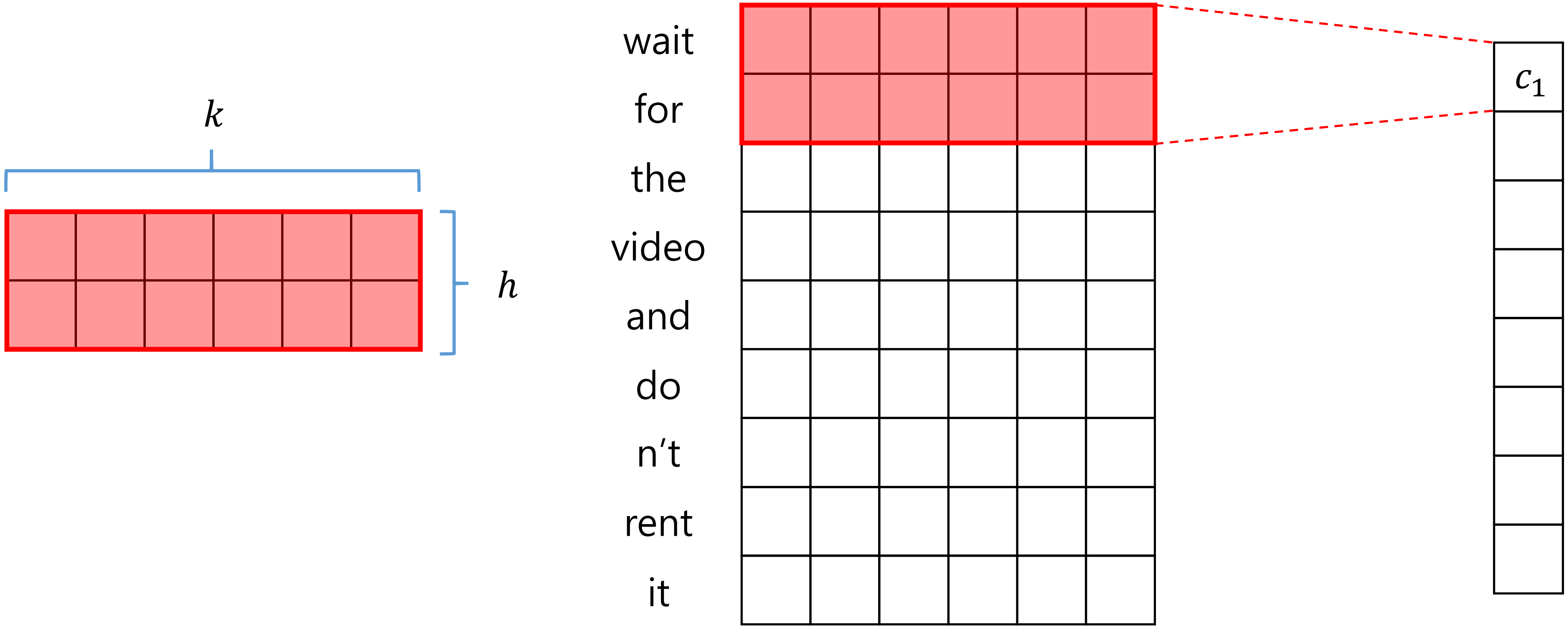
$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n,$$





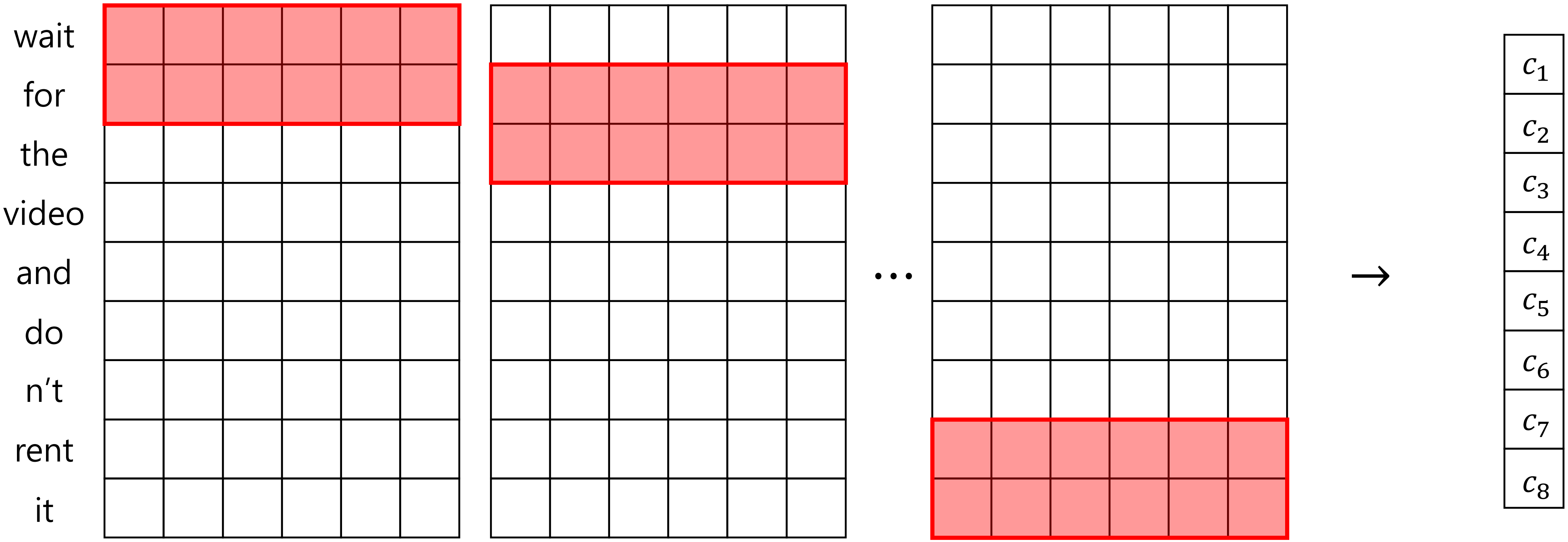
Model

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



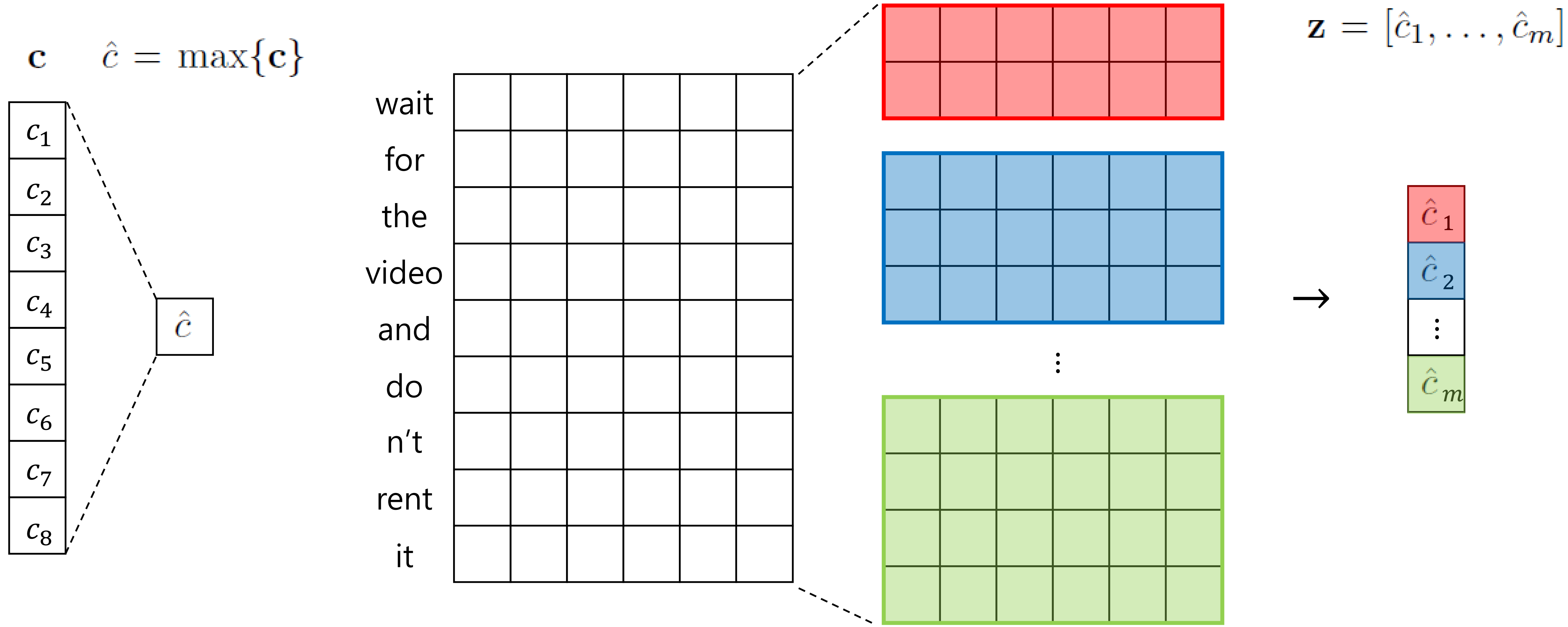
Model

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



Convolutional Neural Networks for Sentence Classification, Yoon Kim(2014)

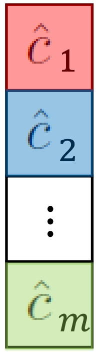
Model



$\mathbf{c}$
$\hat{c} = \max\{\mathbf{c}\}$
c_1
c_2
c_3
c_4
c_5
c_6
c_7
c_8
$\hat{c}$
wait
for
the
video
and
do
n't
rent
it
$\mathbf{z} = [\hat{c}_1, \ldots, \hat{c}_m]$
$\hat{c}_1$
$\hat{c}_2$
$\hat{c}_m$

Model

$$\mathbf{z} = [\hat{c}_1, \ldots, \hat{c}_m]$$



$$y = \mathbf{w} \cdot (\mathbf{z} \circ \mathbf{r}) + b,$$

where $\circ$ is the element-wise multiplication operator and $\mathbf{r} \in \mathbb{R}^m$ is a 'masking' vector of Bernoulli random variables with probability p of being 1. Gradients are backpropagated only through the unmasked units. At test time, the learned weight vectors are scaled by p such that $\hat{\mathbf{w}} = p\mathbf{w}$, and $\hat{\mathbf{w}}$ is used (without dropout) to score unseen sentences. We additionally constrain l_2-norms of the weight vectors by rescaling $\mathbf{w}$ to have $||\mathbf{w}||_2 = s$ whenever $||\mathbf{w}||_2 > s$ after a gradient descent step.

Experiments

Data	c	l	N	$\|V\|$	$\|V_{pre}\|$	*Test*
MR	2	20	10662	18765	16448	CV
SST-1	5	18	11855	17836	16262	2210
SST-2	2	19	9613	16185	14838	1821
Subj	2	23	10000	21323	17913	CV
TREC	6	10	5952	9592	9125	500
CR	2	19	3775	5340	5046	CV
MPQA	2	3	10606	6246	6083	CV

Table 1: Summary statistics for the datasets after tokenization. c: Number of target classes. l: Average sentence length. N: Dataset size. $|V|$: Vocabulary size. $|V_{pre}|$: Number of words present in the set of pre-trained word vectors. *Test*: Test set size (CV means there was no standard train/test split and thus 10-fold CV was used).

For all datasets we use: rectified linear units, filter windows (h) of 3, 4, 5 with 100 feature maps each, dropout rate (p) of 0.5, l_2 constraint (s) of 3, and mini-batch size of 50. These values were chosen via a grid search on the SST-2 dev set.

2011; Socher et al., 2011; Iyyer et al., 2014). We use the publicly available `word2vec` vectors that were trained on 100 billion words from Google News. The vectors have dimensionality of 300 and were trained using the continuous bag-of-words architecture (Mikolov et al., 2013). Words not present in the set of pre-trained words are initialized randomly.

Experiments

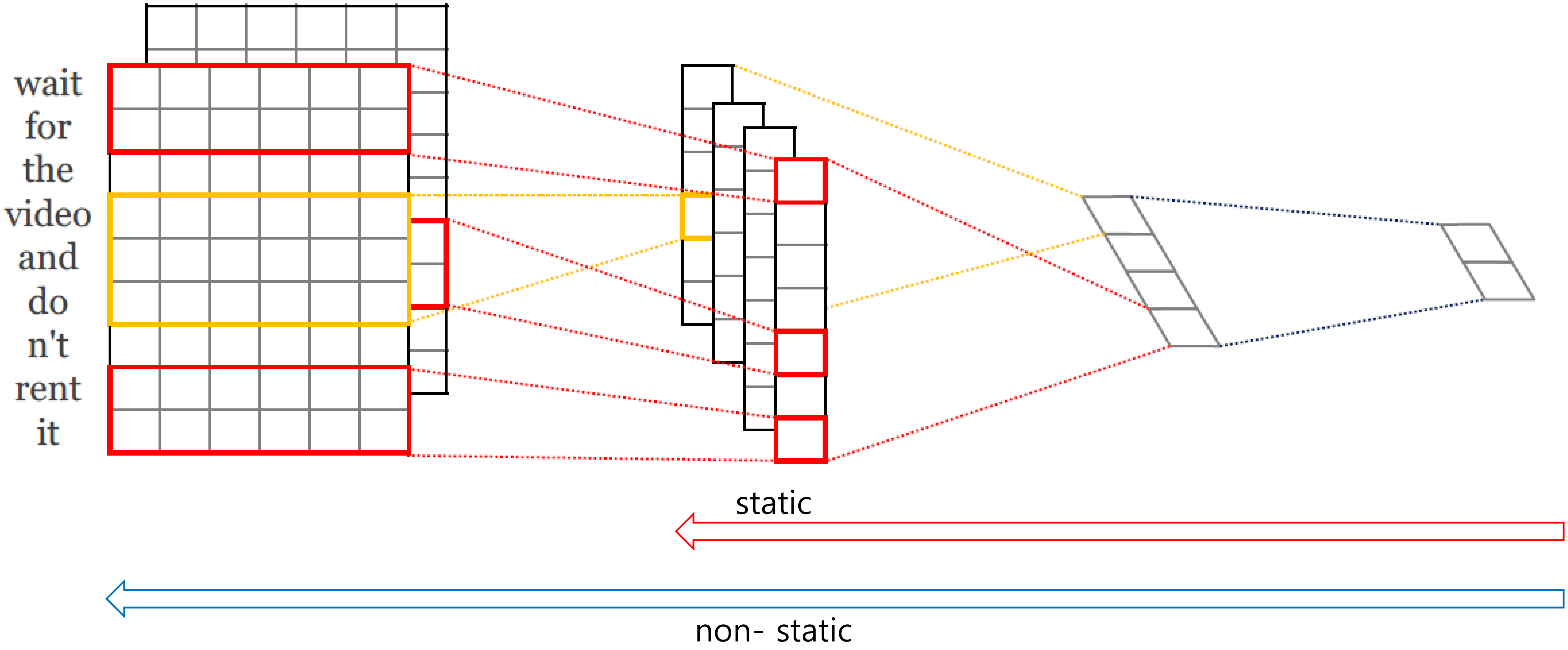
- **CNN-rand**: Our baseline model where all words are randomly initialized and then modified during training. → **Random Initialization & Update**

- **CNN-static**: A model with pre-trained vectors from `word2vec`. All words—including the unknown ones that are randomly initialized—are kept static and only the other parameters of the model are learned. → **Pre-trained Word2Vec**

- **CNN-non-static**: Same as above but the pre-trained vectors are fine-tuned for each task. → **Pre-trained Word2Vec & Fine-tuning**

- **CNN-multichannel**: A model with two sets of word vectors. Each set of vectors is treated as a 'channel' and each filter is applied → **Pre-trained Word2Vec + Pre-trained Word2Vec & Fine-tuning**

Experiments



Result

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**	–	–

Result

4.1 Multichannel vs. Single Channel Models

We had initially hoped that the multichannel architecture would prevent overfitting (by ensuring that the learned vectors do not deviate too far from the original values) and thus work better than the single channel model, especially on smaller datasets. The results, however, are mixed, and further work on regularizing the fine-tuning process is warranted. For instance, instead of using an additional channel for the non-static portion, one could maintain a single channel but employ extra dimensions that are allowed to be modified during training.

4.2 Static vs. Non-static Representations

As is the case with the single channel non-static model, the multichannel model is able to fine-tune the non-static channel to make it more specific to the task-at-hand. For example, *good* is most similar to *bad* in `word2vec`, presumably because they are (almost) syntactically equivalent. But for vectors in the non-static channel that were fine-tuned on the SST-2 dataset, this is no longer the case (table 3). Similarly, *good* is arguably closer to *nice* than it is to *great* for expressing sentiment, and this is indeed reflected in the learned vectors.

For (randomly initialized) tokens not in the set of pre-trained vectors, fine-tuning allows them to learn more meaningful representations: the network learns that exclamation marks are associated with effusive expressions and that commas are conjunctive (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*

References

- DSBA_정의석, [Paper Review] Convolutional Neural Networks for Sentence Classification
- 곽근봉, [slideshare] convolutional-neural-networks-for-sentence-classification

Thank you