Paper 1 | Convolutional Neural Networks for Sentence Classification by Yoon Kim

Abstract

1. Simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks
2. Learning task-specific vectors through fine-tuning further improves performance
3. Proposed a simple modification to allow for the use of both task-specific and static vectors

Introduction

1. Word vectors are essentially feature extractors that encode semantic features of words in their dimensions. In such dense representations, semantically close words are likewise close in Euclidean or cosine distance, in the lower dimensional vector space
2. CNN, although originally invented for computer vision, has proven to be effective for NLP, having achieved great results in **semantic parsing**, **search query retrieval**, **sentence modelling**, and other traditional NLP tasks

Model

a) Model Variations

1. CNN-rand
    - Baseline model where all words are randomly initialised and then modified during training
2. CNN-static
    - A model with pre-trained vectors from word2vec. All words, including unknown ones that are randomly initialised, are kept static and only the other parameters of the model are learned
3. CNN-non-static
    - Same as CNN-static but pre-trained vectors are fine-tuned for each task
4. CNN-multichannel
    - A model with two sets of word vectors. Each set of vectors is treated as a 'channel'. Each filter is applied to both channels and the results are added to calculate features
    - Gradients are only backpropagated through one of the channels
    - Both channels are initialised with word2vec

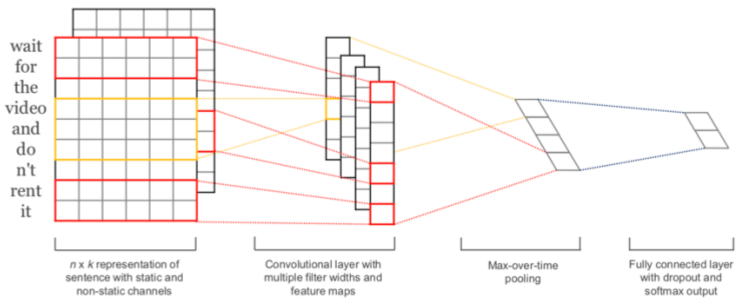

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

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

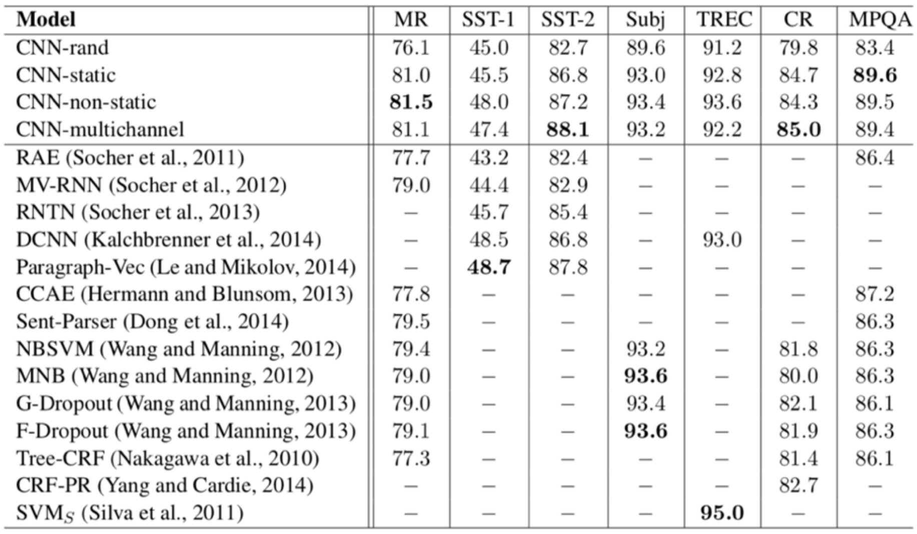
b) Regularisation

- Dropout with a constraint on l_2-norms of weight vectors
- Dropout prevents co-adaptation of hidden units by randomly dropping out (setting to zero) a probability p of the hidden units during forward propagation
- At test time, the learned weight vectors are scaled by p such that w_hat = pw and w_hat is used to score unseen sentences
- Constrain l_2-norms of weight vectors by rescaling w to have $||w||_2 = s$ whenever $||w||_2 > s$ after a gradient descent step

c) Hyperparameters (chosen via grid search on SST-2 dev set)

- Rectified linear units (ReLU)
- Filter windows (h) of 3, 4, 5
- 100 filters for each window (therefore 100 feature maps for each filter window)
- Dropout rate (p) = 0.5
- l_2 constraint (s) = 3

Results



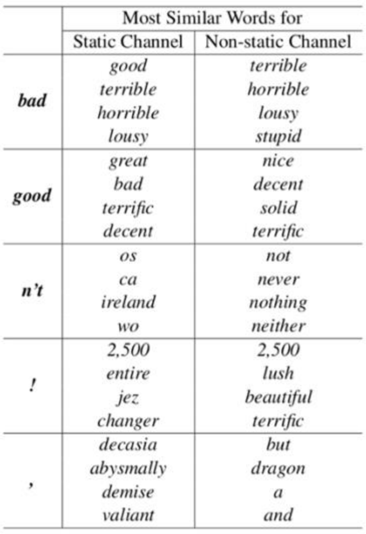
Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (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**	–	–

a) Multichannel vs Single Channel Models

- Initially hope that multichannel architecture would prevent overfitting (by ensuring that the learned vectors do not deviate too far from original values) but the results were mixed

b) Static vs Non-static representations

- Both the single and multichannel models are able to fine-tune the non-static channel to make it more specific to the task at-hand
- For randomly initialised tokens (words not in the set of pre-trained vectors), fine-tuning allows them to learn more meaningful representations. For example, look at exclamation marks and commas



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

c) Further observations

- Dropout proved to be a good regulariser, allowing to use a larger than necessary network. Consistently added 2%-4% relative performance
- Obtained slight improvements by sampling each dimension from U[-a,a] where a was chosen such that the randomly initialised vectors have the same variance as the pre-trained vectors

Conclusion

1. This paper described a series of experiments with CNN built on top of word2vec
2. The results add to the well-established evidence that unsupervised pre-training of **word vectors is an important ingredient in deep learning for NLP**