Convolutional Neural Networks for Sentence Classification

2019.01.12

김보섭

Agenda

1. **Abstract**
2. **Introduction**
3. **Model**
4. **Datasets and Experimental Setup**
5. **Results and Discussion**
6. **Conclusion**

Abstract

pre-trained word vector와 간단한 convolution neural network를 이용하여, 다양한 sentence-level classification에서 좋은 성능을 보여줌

Convolutional Neural Networks for Sentence Classification

Yoon Kim
New York University
yhk255@nyu.edu

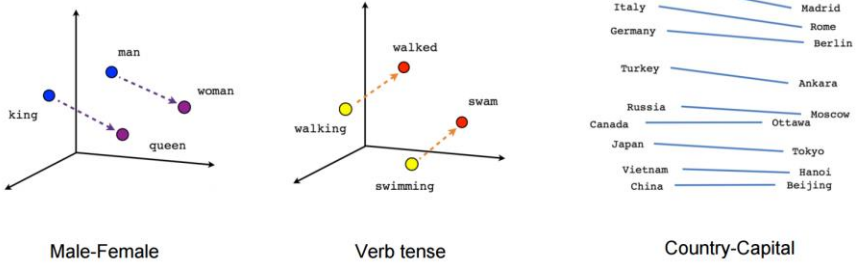
Abstract

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.

Introduction

semantic 잘 encoding하는 word vector를 universal feature extractor로 활용하고, 이를 convolution neural network로 학습하는 강건한 model을 제안

word vector

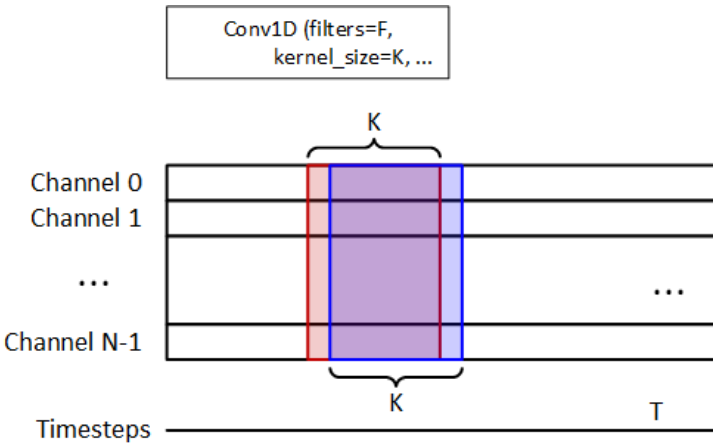


Male-Female Verb tense Country-Capital

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

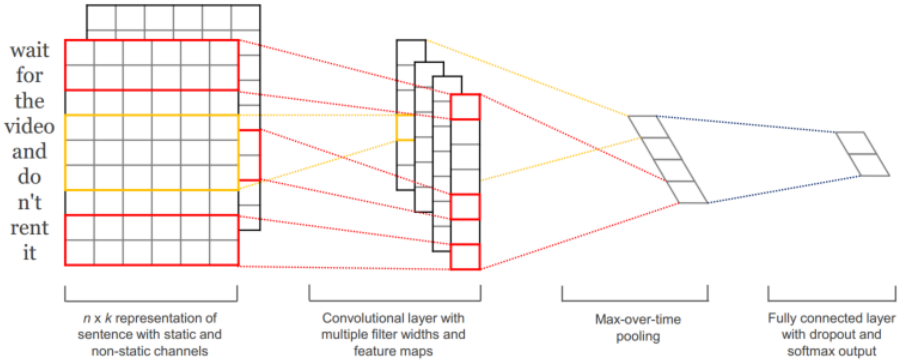
1d convolution



https://www.tensorflow.org/tutorials/representation/word2vec
Distributed Representations of Words and Phrases and their Compositionality (https://arxiv.org/pdf/1310.4546.pdf)
https://medium.com/@jon.froiland/convolutional-neural-networks-for-sequence-processing-part-1-420dd9b500

Model : CNN-multichannel

pre-trained word vector를 이용, training 유무에 따라 static channel, non-static channel로 representation하고, 이를 input으로 model을 학습



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

Input (sentence length : n)

$$x_{1:n} = x_1 \oplus x_2 \oplus \cdots \oplus x_n, x_{1:n} \in \mathbb{R}^{nk}, x_i \in \mathbb{R}^k$$

Convolution layer (filter width : h)

convolution operation (each filter)

$w_{conv} \in \mathbb{R}^{hk}, b_{conv} \in \mathbb{R}$

$$c_i^{static} = f\left(w_{conv} \cdot x_{i:i+h-1}^{static} + b_{conv}\right)$$
$$c_i^{non-static} = f\left(w_{conv} \cdot x_{i:i+h-1}^{non-static} + b_{conv}\right)$$
$$c_i = c_i^{static} + c_i^{non-static}$$
$$c = [c_1, c_2, \ldots, c_{n-h+1}], c \in \mathbb{R}^{n-h+1}, c \in \mathbb{R}$$

max-overtime-pooling operation (each feature map)

$$\hat{c} = \max\{c\}, \hat{c} \in \mathbb{R}$$

Fully connected layer (before using m filters, k classes)

$\dim(\mathbf{w}_{dense}) = (k, m), b_{dense} \in \mathbb{R}^m, r \in \mathbb{R}^m$

r : masking vector of Bernoulli random variables with probability p of being 1 (drop out)

$$z = [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_m], z \in \mathbb{R}^m$$

$$y = \mathbf{w}_{dense}(z \circ r) + b_{dense}, \mathbf{w}_{dense} = \begin{bmatrix} w_1^T \\ \vdots \\ w_k^T \end{bmatrix}$$

$$prob = \text{softmax}(y), prob \in \mathbb{R}^m$$

Train and Test

At train time, we additionally constrain $l_2 - norms$ of the weight vectors by rescaling w_{dense} to have $\|w_{dense}\|_2 = s$ whenever $\|w_{dense}\|_2 > s$ after a gradient descent step. At test time (without drop out),

$$\hat{\mathbf{w}}_{dense} = p\mathbf{w}_{dense} \text{ (scaled by p)}$$

Model : Variation

CNN-multichannel model 말고도, 실험을 위해 아래와 같은 variation들을 학습하고 비교함

- CNN-rand
  - single channel
  - 모든 word vector에 대해서 random initialization
  - word vector가 training에 포함됨
- CNN-static
  - Single channel
  - 기본적으로 pre-trained word vector로 initialization을 하고, pre-trained word vector set에 없는 word의 경우 random initialization
  - word vector가 training에 포함되지않음
- CNN-non-static
  - single channel
  - 기본적으로 pre-trained word vector로 initialization을 하고, pre-trained word vector set에 없는 word의 경우 random initialization
  - word vector가 training에 포함됨
- CNN-multichannel
  - double channel (static channel, non-static channel)
  - word vector를 initialization하는 방법을 동일하게 각 channel에 적용
  - non-static channel이 training에 포함됨

Datasets and Experimental Setup

다양한 데이터셋에 같은 구조를 지닌 모형으로 실험하고, pre-trained word vector로는 continuous bag of words 방식으로 학습된 word vector 활용

3.1 Hyperparameters and Training

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.

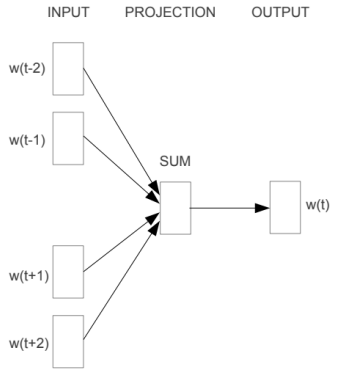
We do not otherwise perform any dataset-specific tuning other than early stopping on dev sets. For datasets without a standard dev set we randomly select 10% of the training data as the dev set. Training is done through stochastic gradient descent over shuffled mini-batches with the Adadelta update rule (Zeiler, 2012).

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).

3.2 Pre-trained Word Vectors

Initializing word vectors with those obtained from an unsupervised neural language model is a popular method to improve performance in the absence of a large supervised training set (Collobert et al., 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.



CBOW

Efficient Estimation of Word Representations in Vector Space (https://arxiv.org/pdf/1301.3781.pdf)

Results and Discussion (1/2)

pre-trained word vector를 사용만해도 성능이 증가, fine-tuning하는 경우에는 word vector가 task-specific하게 semantic을 잘 encoding함

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**	–	–

Table 2: Results of our CNN models against other methods. **RAE**: Recursive Autoencoders with pre-trained word vectors from Wikipedia (Socher et al., 2011). **MV-RNN**: Matrix-Vector Recursive Neural Network with parse trees (Socher et al., 2012). **RNTN**: Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013). **DCNN**: Dynamic Convolutional Neural Network with k-max pooling (Kalchbrenner et al., 2014). **Paragraph-Vec**: Logistic regression on top of paragraph vectors (Le and Mikolov, 2014). **CCAE**: Combinatorial Category Autoencoders with combinatorial category grammar operators (Hermann and Blunsom, 2013). **Sent-Parser**: Sentiment analysis-specific parser (Dong et al., 2014). **NBSVM, MNB**: Naive Bayes SVM and Multinomial Naive Bayes with uni-bigrams from Wang and Manning (2012). **G-Dropout, F-Dropout**: Gaussian Dropout and Fast Dropout from Wang and Manning (2013). **Tree-CRF**: Dependency tree with Conditional Random Fields (Nakagawa et al., 2010). **CRF-PR**: Conditional Random Fields with Posterior Regularization (Yang and Cardie, 2014). **SVM_S**: SVM with uni-bi-trigrams, wh word, head word, POS, parser, hypernyms, and 60 hand-coded rules as features from Silva et al. (2011).

	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*

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.

Sentiment analysis dataset

Results and Discussion (2/2)

pre-trained word vector가 존재하지 않는 word의 경우, pre-trained word vector의 각 dimension의 variance를 반영하여 initialization 하는 것이 좋음

- Dropout proved to be such a good regularizer that it was fine to use a larger than necessary network and simply let dropout regularize it. Dropout consistently added 2%–4% relative performance.
- When randomly initializing words not in `word2vec`, we obtained slight improvements by sampling each dimension from $U[-a, a]$ where a was chosen such that the randomly initialized vectors have the same variance as the pre-trained ones. It would be interesting to see if employing more sophisticated methods to mirror the distribution of pre-trained vectors in the initialization process gives further improvements.

Conclusion

5 Conclusion

In the present work we have described a series of experiments with convolutional neural networks built on top of `word2vec`. Despite little tuning of hyperparameters, a simple CNN with one layer of convolution performs remarkably well. Our results add to the well-established evidence that unsupervised pre-training of word vectors is an important ingredient in deep learning for NLP.

Q & A

감사합니다.