Convolutional Neural Network for Senctence Classification

Yoon Kim, 2014

윤훈상, 양광민, 양태일

Yoon Kim

Yoon Kim

I am a fourth-year PhD student in Computer Science at Harvard University, working on machine learning and natural language processing. My advisor is Alexander Rush, and I am part of the HarvardNLP group. I am supported by a Google Fellowship.

Previously I was a master's student in Data Science at New York University, where I worked with David Sontag, and prior to that I obtained a master's degree in Statistics from Columbia University. My bachelor's degree is from Cornell University, in Mathematics and Economics.

[CV] [Google Scholar] [LinkedIn]

Agenda

오늘 발표는 소개하기
많이 6 page라
프랜차로 시간이 짧은 수밖에
없을듯 발표는 바로끝.

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

1. Abstract

- Pre-trained word vector + (Simple) Convolution Neural Network

==> High Performance on sentence-level classification

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.

Sentence Classification

> 감정 분류(Sentiment Analysis)

예시)

- 이번 아이폰의 카메라 성능은 정말 좋은 것 같아 – 긍정
- 이 레스토랑의 음식은 정말 실망스러웠어 – 부정

> 주제 분류

예시)

- 유승민의 자신감, 19대 대선 예비후보 등록 – 정치
- 손흥민 없는 슈틸리케호, 중국전 공격 조합은? – 스포츠

2. Introduction

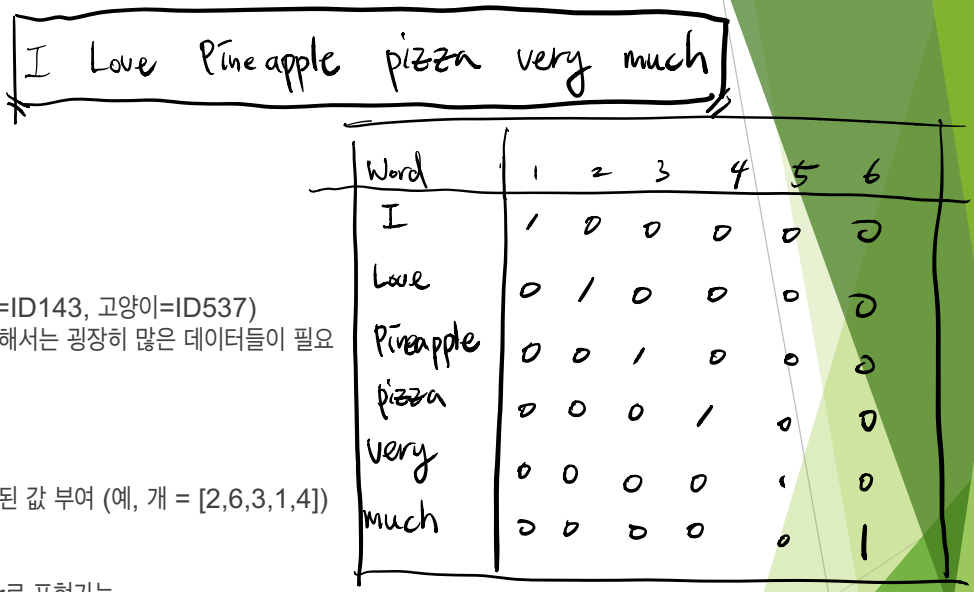
- Deep learning Model이 CV, NLP 등에서 다양하게 사용되고 있다.

- Word embedding(Bengio, 2003, Mikolov 2013)에서부터
- Classification(Collobert, 2011) 에 이르면서 다양하게...

- 이 논문에서는 Pre-trained Word Vector를 연구에 포함시킨 것을 큰 Novelty로 삼고 있다.

--> Sparse Vectors (to) Dense Vectors

2. Introduction



> Sparse Vector

- 사전을 만들어서 ID를 부여하자
- 간단하고 적용하기 쉬움
- 단어들과의 관계를 나타내지 못함 (예, 개=ID143, 고양이=ID537)
- 모든 단어가 다르기 때문에, 학습시키기 위해서는 굉장히 많은 데이터들이 필요

> Dense Vector

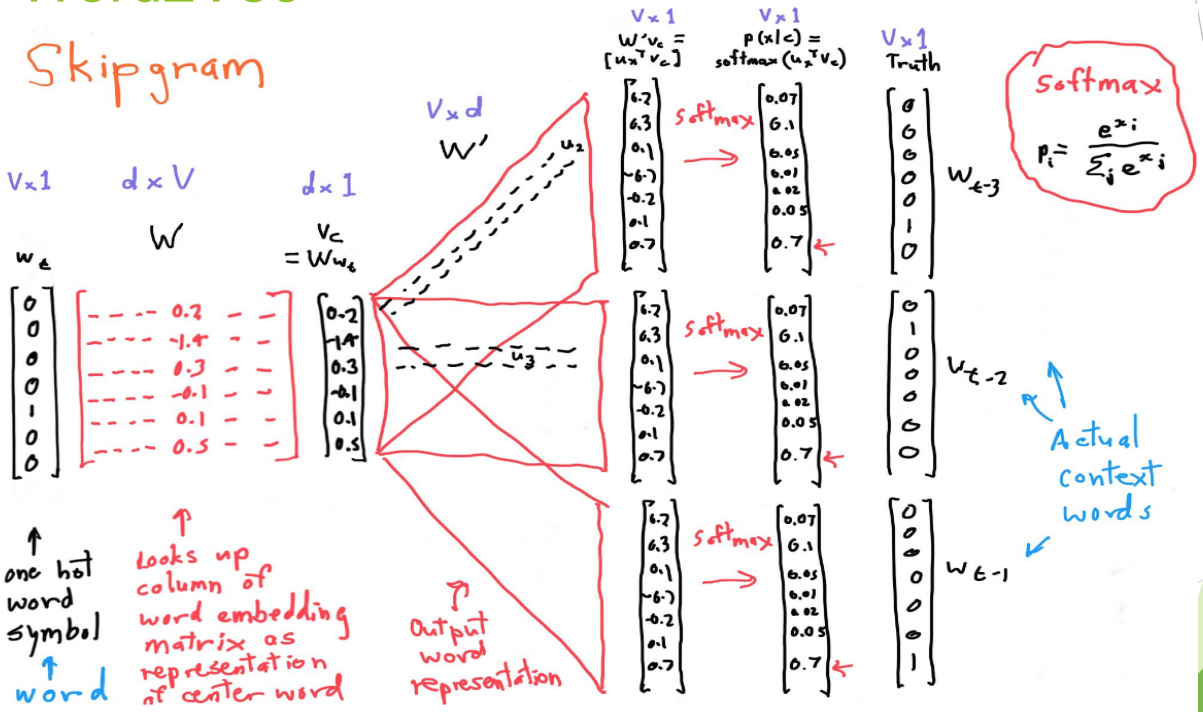
- 각 단어마다 Vector 값을 부여하자
- 단어들의 특징을 표현할 수 있도록 수치로 된 값 부여 (예, 개 = [2,6,3,1,4])

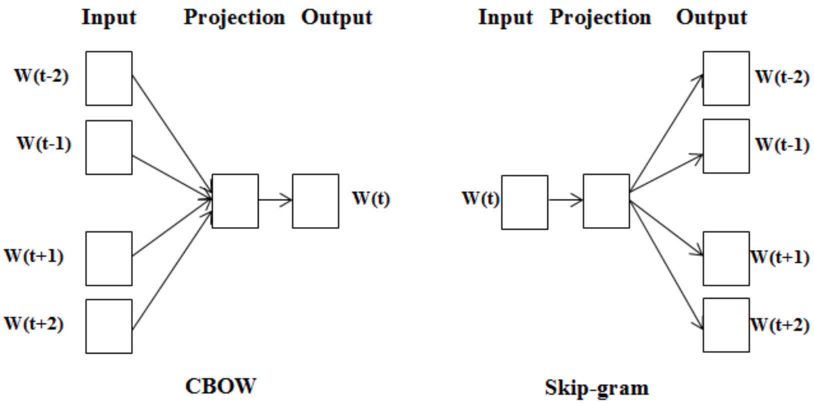
- 단어간의 관계 표현 가능

- 단어를 discrete한 symbol이 아닌 vector로 표현가능

- Sparse vector(1-hot) --> 저차원의 dense vector로 매핑

Word2Vec


Skipgram
V×1
d×V
W
d×1
$v_c = Wx_c$
x_c
V×d
W'
u_2
u_3
V×1
$W'v_c = [u_x^T v_c]$
V×1
$p(x|c) = \text{softmax}(u_x^T v_c)$
V×1
Truth
Softmax
$p_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$
W_{t-3}
W_{t-2}
W_{t-1}
Actual context words
one hot word symbol
word
looks up column of word embedding matrix as representation of center word
Output word representation



CBOW: 주변단어 → 중심단어

Skip-gram: 중심단어 → 주변단어

- yesterday was really [...] day
- [...] [..] [...] delight [..]

태일's 뇌피셜:

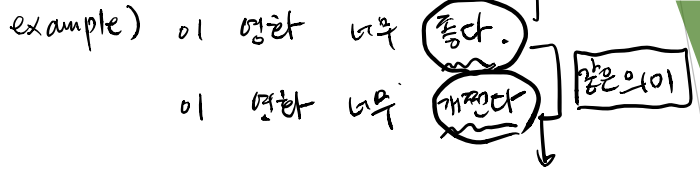
문장의 유인가를 분류함에 있어서
정확하고, 최다빈도를 찾아야 하기 때문에

https://www.youtube.com/watch?v=sY4YyacSsLc

Here is my oversimplified and rather naive understanding of the difference:

As we know, **CBOW** is learning to predict the word by the context. Or maximize the probability of the target word by looking at the context. And this happens to be a problem for rare words. For example, given the context `yesterday was really [...] day` CBOW model will tell you that most probably the word is `beautiful` or `nice` . Words like `delightful` will get much less attention of the model, because it is designed to predict the most probable word. Rare words will be smoothed over a lot of examples with more frequent words.

On the other hand, the **skip-gram** is designed to predict the context. Given the word `delightful` it must understand it and tell us, that there is huge probability, the context is `yesterday was really [...] day` , or some other relevant context. With **skip-gram** the word `delightful` will not try to compete with word `beautiful` but instead, `delightful+context` pairs will be treated as new observations. Because of this, **skip-gram** will need more data so it will learn to understand even rare words.



2. Introduction

- 본 연구에서는 이 Word vector를 Mikolov가 직접 만든 100 billion word vector로 사용하겠다.
- Razavian (2014)의 논문과 철학을 공유하는데 그 철학이란:

  '기존에 세워져 있는 것을 활용하는 것!'

  'Transfer Learning의 모태가 아닐까...'

3. Model: CNN-multichannel

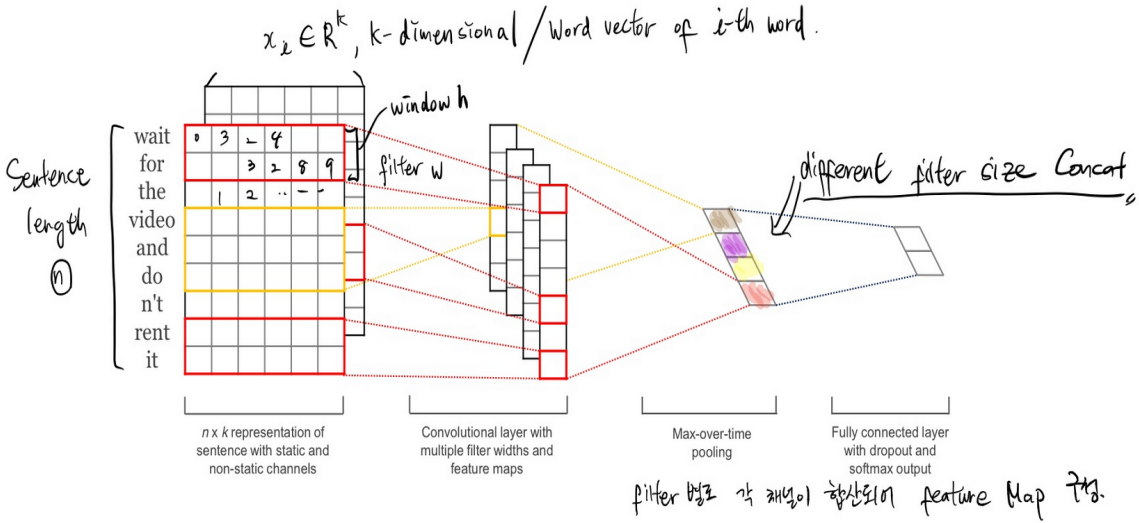


Figure 1: Model architecture with two channels for an example sentence.

3. Model: CNN-multichannel

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),

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

3. Model: CNN-multichannel

Parameters Used:

- ReLU Used
- filter windows(h) : 3,4,5
- 100 Feature maps each
- Dropout rate(p) : 0.5
- L2 Constraint(s) : 3
- mini-batch size : 50

==> All these chosen via SST-2 dev set

Word Vectors From:

Publicly available 'word2vec', trained on 100bilion words from Google News.

Vectors have 300 dimension ==> Trained with CBOW(Continuous Bag-of-words)

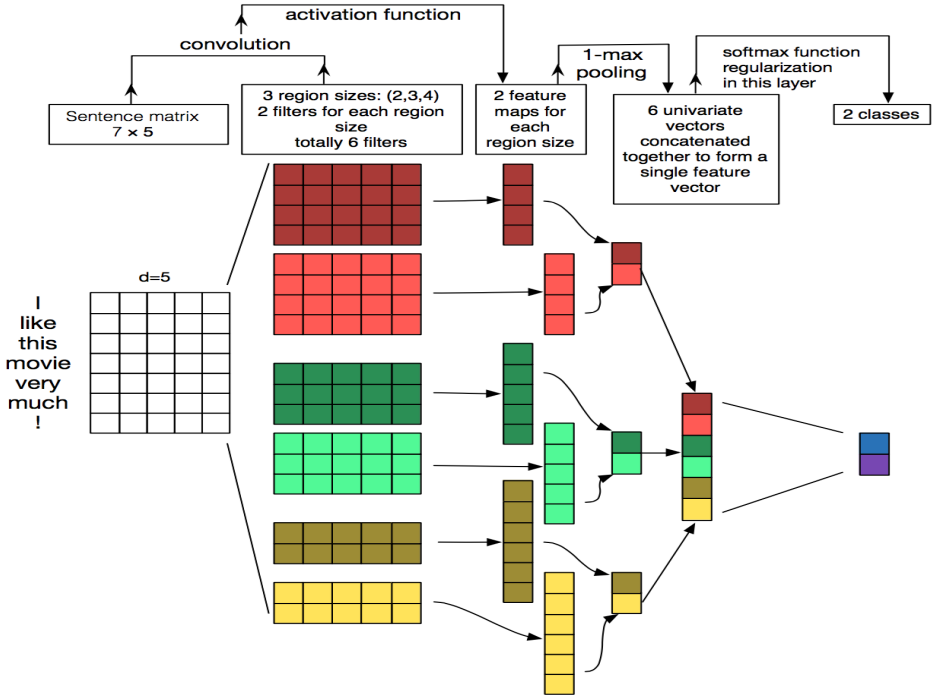
3.1 Hyperparameters and Training

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).

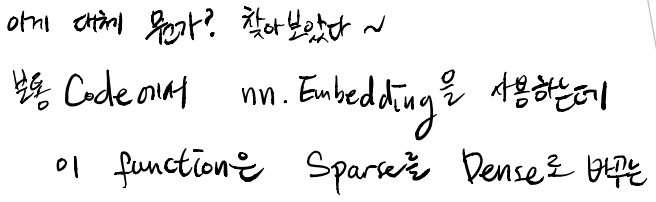
3.2 Pre-trained Word Vectors

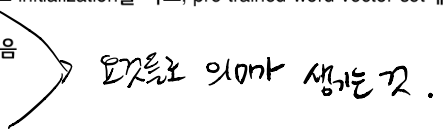
Initializing word vectors with those obtained from an unsupervised neural language model is a popular method to improve performance in the absence of a large supervised training set (Collobert et al., 2011; Socher et al., 2011; Iyyer et al., 2014). We use the publicly available `word2vec` vectors that 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.


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

3. Model: Variation

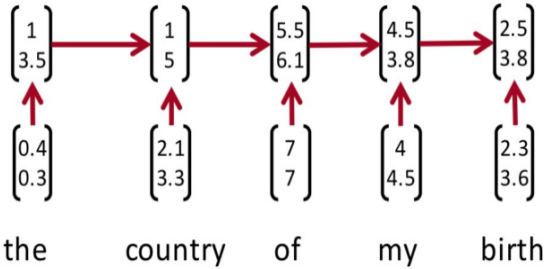
- CNN-rand
  - single channel
  - 모든 word vector에 대해서 random initialization
  - word vector가 training에 포함됨
- CNN-static
  - Single channel
  - 기본적으로 pre-trained word vector로 initialization을 하고, pre-trained word vector set에 없는 word의 경우 random initialization
  - word vector가 training에 포함되지않음
- CNN-non-static
  - 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





Why CNN than RNN?

- Recurrent neural nets cannot capture phrases without prefix context
- Often capture too much of last words in final vector



- Softmax is often only at the last step

Why CNN than RNN?

- Main CNN idea:
- What if we compute vectors for every possible phrase?

- Example: "the country of my birth" computes vectors for:
  - the country, country of, of my, my birth, the country of, country of my, of my birth, the country of my, country of my birth

- Regardless of whether phrase is grammatical
- Not very linguistically or cognitively plausible

- Then group them afterwards (more soon)

http://web.stanford.edu/class/cs224n/archive/WWW_1617/index.html lecture13

4. Datasets and Experimental Setup

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).

Even a simple model with static vectors (CNN-static) performs remarkably well

Fine tuning the pre trained vectors for each task gives still further improvements (CNN-non-static).

5. Results and Discussion

	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.

For (randomly initialized) tokens not in the set of pre-trained vectors, fine-tuning allows them to learn more meaningful representations.

Dropout -> 2%~4% 성능향상

Wikipedia(2011) Word Vector 사용
->word2vec성능이 더 뛰어남

Adagrad를 사용해도 비슷한 결과가 나오지만 더 느림.

Word2vec에 없는 단어 초기화 할 때에 같은 학습되어 있는 벡터의 분산에서 랜덤 추출함.

6. 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.

간단한 CNN구조를 이용하였음에도 word2vec를 이용한 학습이 성능이 뛰어나다는 것을 증명할 수 있는 실험이었다.

유사 연구 (한국어 NLP)

컨볼루션 신경망 기반 대용량 텍스트 데이터 분류 기술

조휘열[01] 김진화[2] 윤상웅[3] 김경민[1] 장병탁[1,2,3]
서울대학교 공과대학 컴퓨터공학과[1]
서울대학교 인문대학 협동과정 인지과학전공[2]
서울대학교 자연대학 협동과정 뇌과학전공[3]
{hyjo, jhkim, swyoon, kmkim, btzhang}@bi.snu.ac.kr

word2vec을 신경망의 단어 임베딩 초기화에 적용해보았으나, 기존에 보고된 바와 달리 큰 도움이 되지 않는 것을 관찰했다.
이러한 결과들을 통해 딥 러닝 기반 한국어 문서 분류 시스템의 발전에 기여할 수 있을 것으로 기대된다.

유사 연구 (한국어 NLP)

Model	Accuracy (Top-1,3,5)		
MNB	0.641	0.911	0.958
SVM	0.795	0.960	0.991
CNN	**0.856**	**0.986**	**0.997**

표 2 모델 별 대주제 실험 정확도

Model	Accuracy (Top-1,3,5)		
MNB	0.399	0.679	0.794
SVM	0.614	0.851	0.906
CNN	**0.700**	**0.920**	**0.962**

표 3 모델 별 소주제 실험 정확도

Repre.	Accuracy (Top-1,3,5)		
Rand	0.856	**0.986**	**0.997**
W2V	**0.857**	0.985	**0.997**

표 4 Representation 별 대주제 실험 정확도

Repre.	Accuracy (Top-1,3,5)		
Rand	**0.700**	0.920	**0.962**
W2V	0.696	**0.921**	**0.962**

표 5 Representation 별 소주제 실험 정확도

CNN모델이 MNB나 SVM같은 모델보다 정확도가 높음을 확인할 수 있다. 다만 같은 CNN모델 안에서 랜덤으로 값을 지정하였을 때와 word2vec으로 값을 지정하였을 시에 큰 성능차이가 발생하지 않았다.

유사 연구 (한국어 NLP)

본 연구에서는 문서마다 처음 n개의 단어를 보며 빈도수를 기준으로 상위 m개까지 단어를 사전에 저장하도록 하였다. 사전에 저장된 단어는 랜덤 표현 또는 Word2Vec 표현을 하였다. 그 외에 사전에 저장되지 않은 단어는 기타단어로서 모두 동일한 표현을 가졌다.

차이점 1

이 전에 보았던 논문은 word2vec의 단어 사전에 없는 단어들을 각 차원의 분산 범위 안에서 임의 추출하였으나 해당 연구에서는 사전에 없는 단어들을 기타 단어로 동일하게 표현하였다.

유사 연구 (한국어 NLP)

차이점 2

이전의 논문에서 word2vec을 구축하기 위해 사용한 단어 1000억
해당 논문에서 word2vec을 구축하기 위해 사용한 단어 623,303

word2vec를 구축할 때 사용한 단어 개수 차이가 크므로 한국어에서는 word2vec이 효과가 없다고 하기에는 근거가 부족하다.

Q&A

감사합니다!