Convolutional Neural Networks for Sentence Classification

EMNLP 2014

Yoon Kim
New York University

발표: 곽[illegible]봉

Paper

Convolutional Neural Network for Sentence Classification

EMNLP 2014

Yoon Kim, New York University

Idea: **문장 분류(Sentence classification)**문제에 1) **word vector**와 아주 간단한 2) **CNN**을 이용하여 전통적/복잡한 모델 수준의 성능을 보인 모델

https://arxiv.org/pdf/1408.5882.pdf

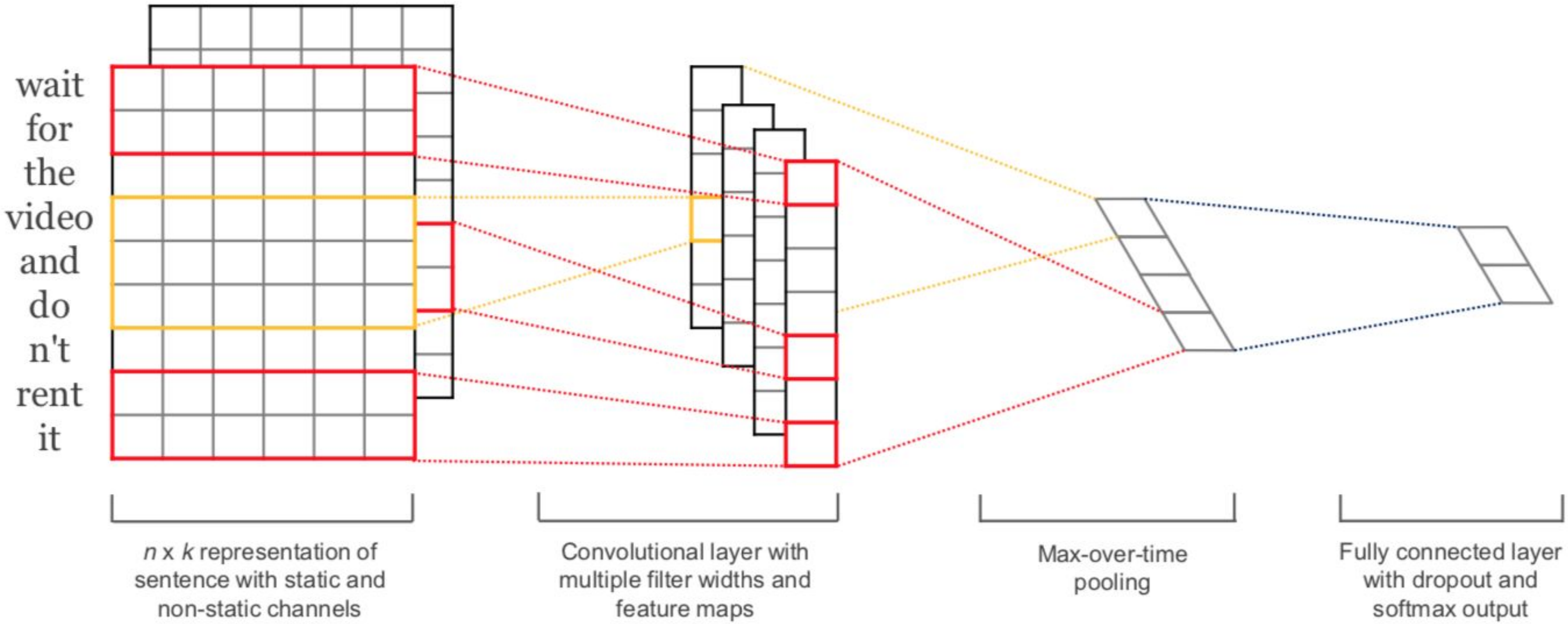


Figure 1: Model architecture with two channels for an example sentence.

Abstract

Word vector와 CNN을 활용한 문장 분류

이미 트레이닝된 word vector를 활용

Word2vec를 활용하여 단어를 vector화 함

Simple한 CNN 구조 사용

3가지 filter를 가진 단순한 Convolutional Layer

높은 정확도

7개의 벤치마크 중 4곳에서 가장 높은 정확도

문장 분류(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

어떻게 단어를 계산할까?

Word Representations(Embedding)

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

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

어떻게 단어에 Vector값을 줄까?

Word2Vec

문장에서 나오는 단어들의 위치로 학습시키자!

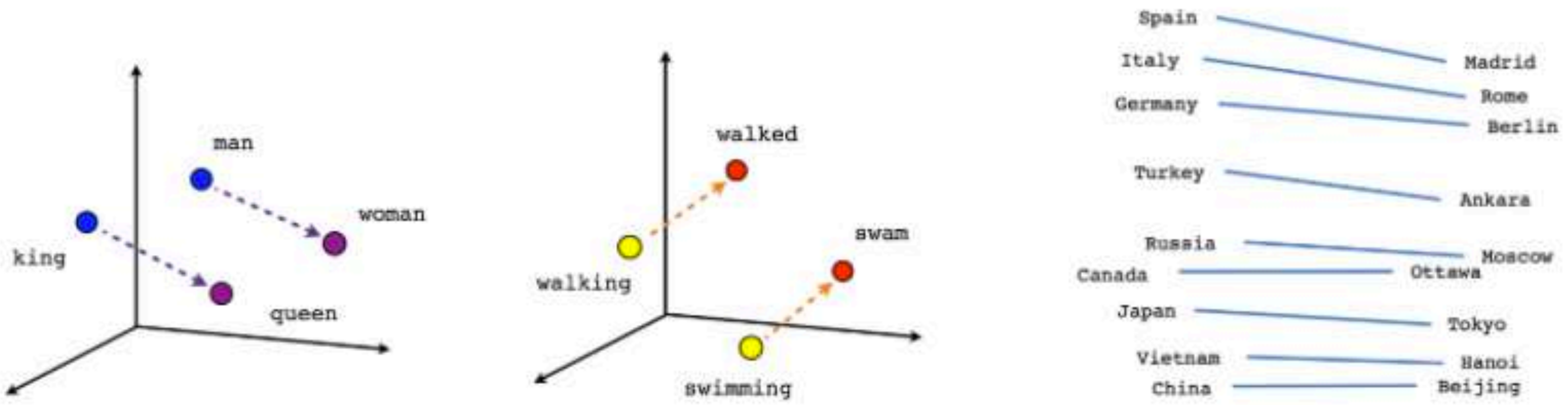
the quick brown fox jumped over the lazy dog

([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox)

어떻게 단어에 Vector값을 줄까?

Word2Vec

그랬더니 특정 방향들이 의미를 담고 있었어!



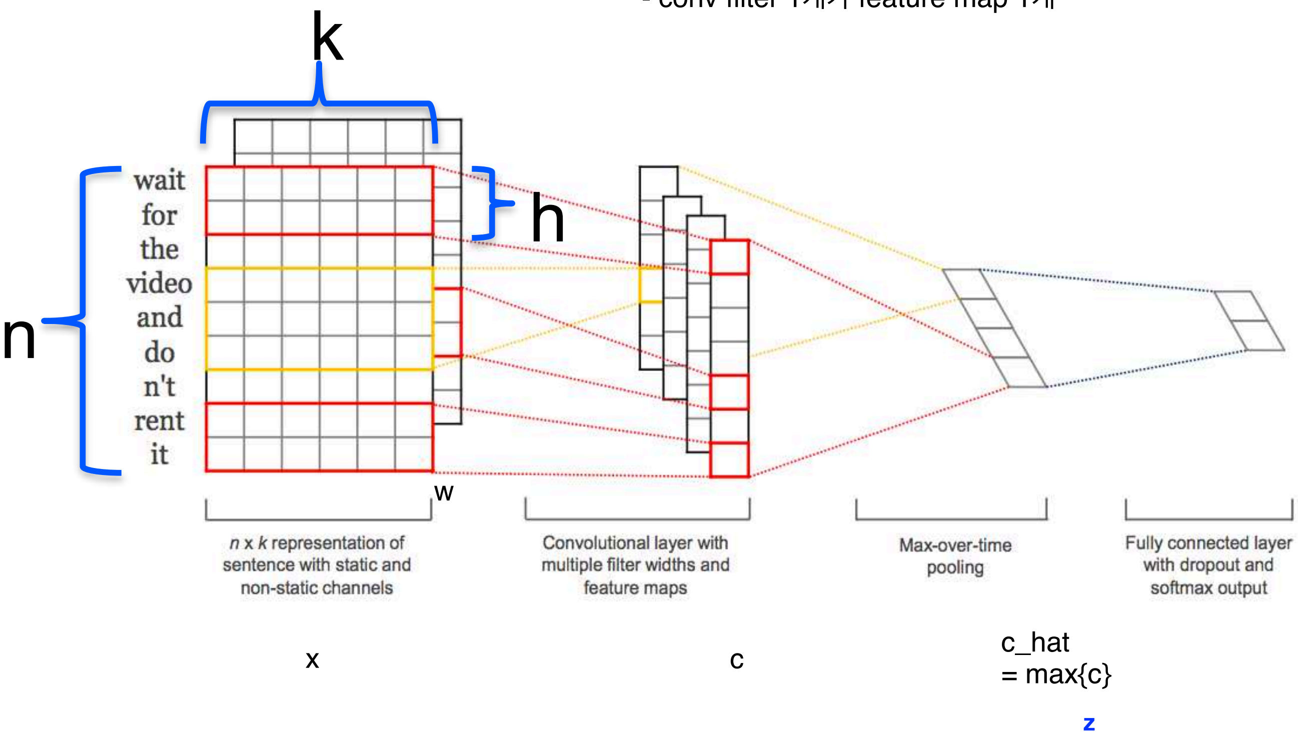
Male-Female Verb tense Country-Capital

https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/tutorials/word2vec/

CNN과 Word Vector를 이용한 문장 분류

Model Architecture

- 여러개의 feature map
- h에 따라서, feature map간의 길이가 다름
- 같은 h의 filter도 여러 개일 수 있다
- conv filter 1개가 feature map 1개



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

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b). \tag{2}$$

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}], \tag{3}$$

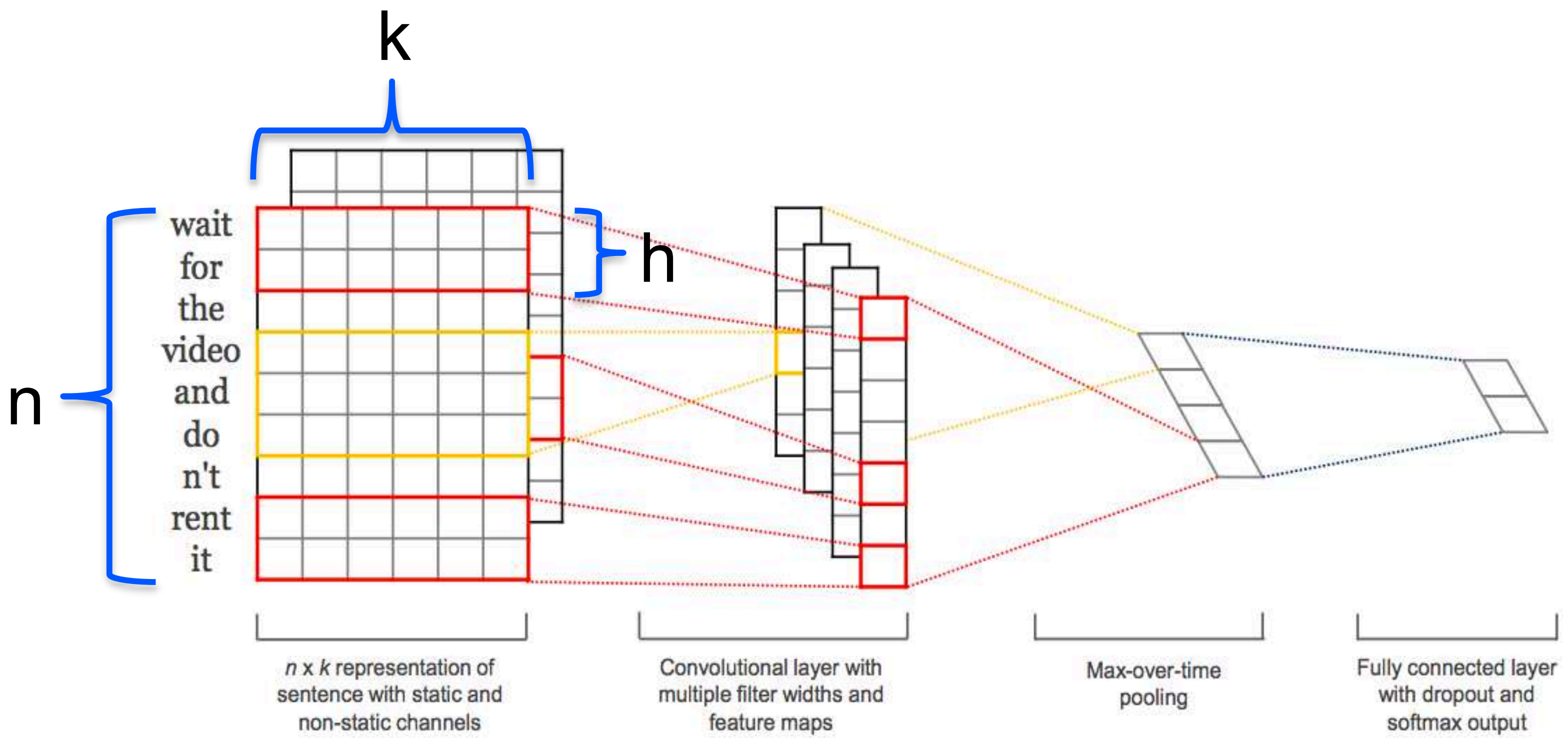
y = w · z + b (4)
대신에

y = w · (z ○ r) + b, (5)
r: a ‘masking’ vector of Bernoulli random variables

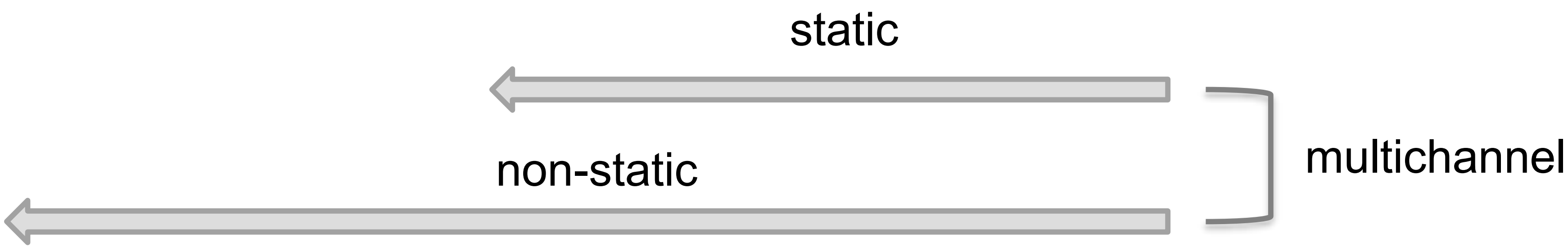
n : 문장에 나오는 단어의 갯수 k : Word Vector의 차원 h : 필터 윈도우 사이즈

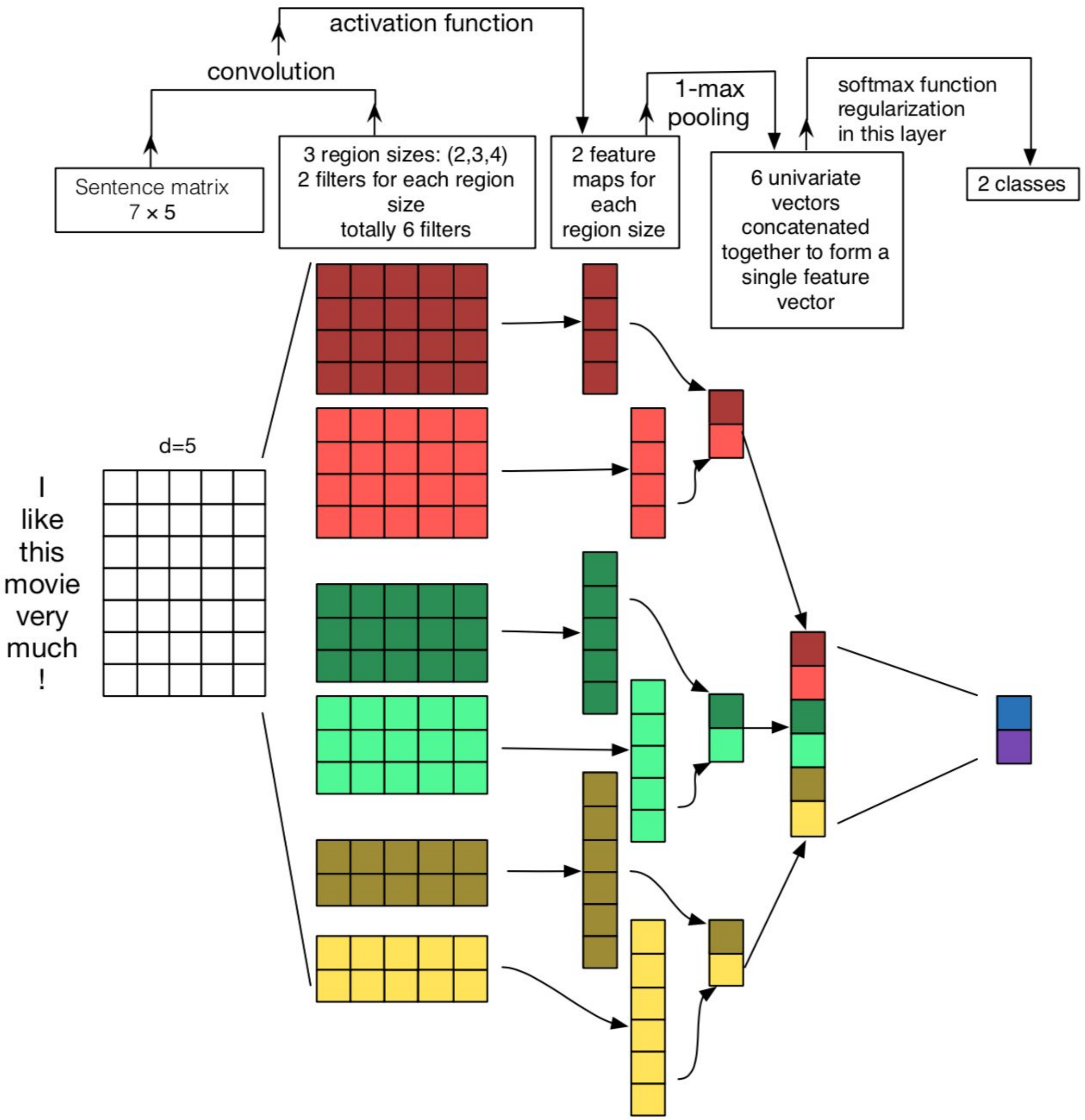
Page2 Figure1 (Yoon Kim, Convolutional Neural Networks for Sentence Classification, EMNLP 2014)

Static, Non-static, Multichannel



Back Propagation

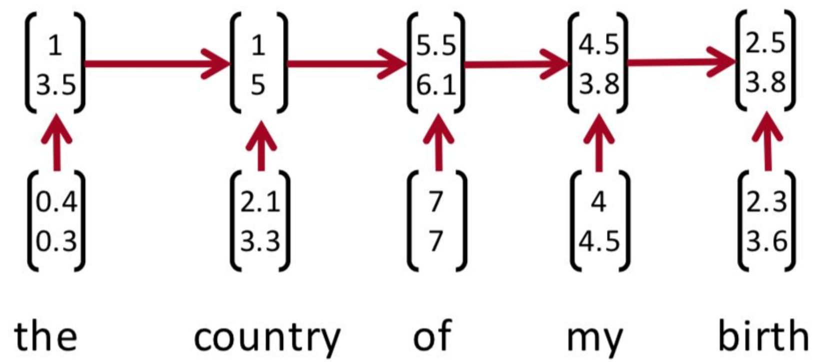




https://arxiv.org/pdf/1510.03820.pdf

From RNNs to CNNs

- 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

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

From RNNs to CNNs

- 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

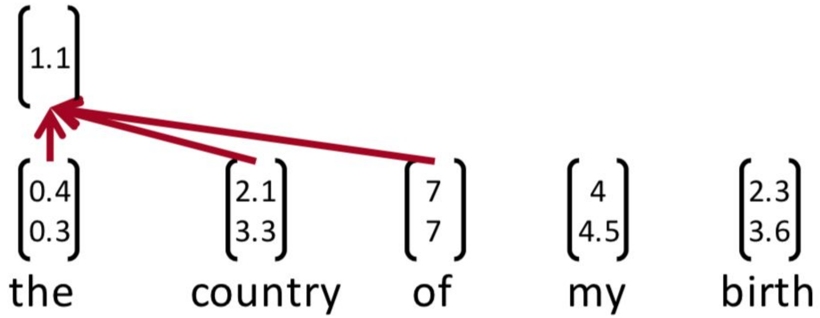
- Regardless of whether phrase is grammatical
- Not very linguistically or cognitively plausible

- Then group them afterwards (more soon)

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

Single Layer CNN

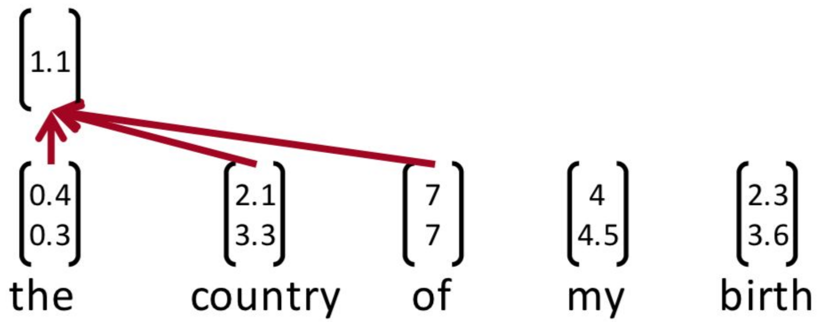
- A simple variant using one convolutional layer and **pooling**
- Based on Collobert and Weston (2011) and Kim (2014)
  "Convolutional Neural Networks for Sentence Classification"
- Word vectors: $\mathbf{x}_i \in \mathbb{R}^k$
- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$ (vectors concatenated)
- Concatenation of words in range: $\mathbf{x}_{i:i+j}$
- Convolutional filter: $\mathbf{w} \in \mathbb{R}^{hk}$ (goes over window of h words)
- Could be 2 (as before) higher, e.g. 3:



Single layer CNN

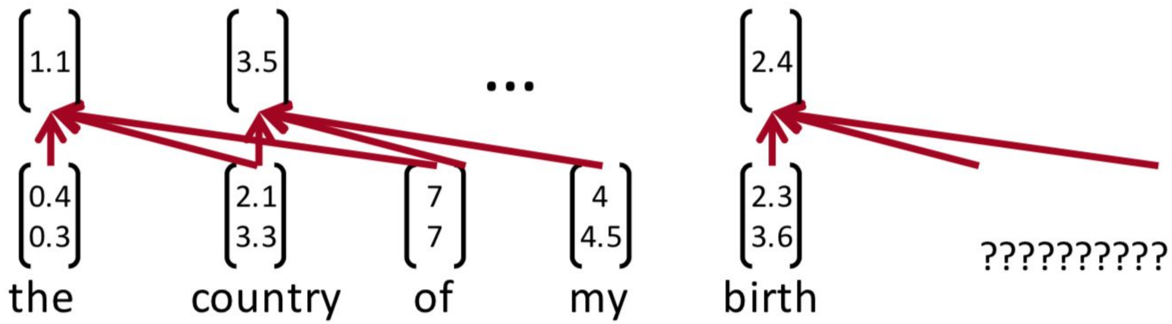
- Convolutional filter: $\mathbf{w} \in \mathbb{R}^{hk}$ (goes over window of h words)
- Note, filter is vector!
- Window size h could be 2 (as before) or higher, e.g. 3:
- To compute feature for CNN layer:

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



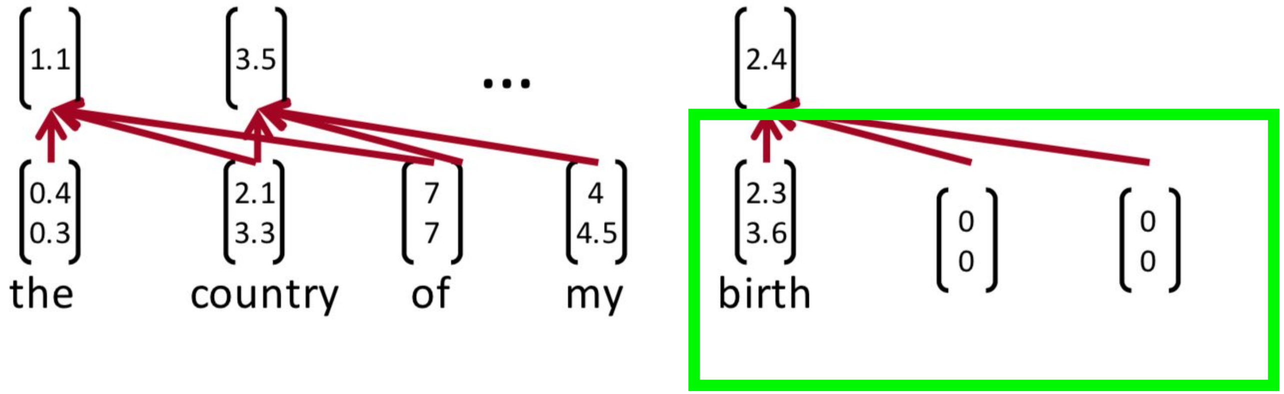
Single layer CNN

- Filter w is applied to all possible windows (concatenated vectors)
- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$
- All possible windows of length h: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$



Single layer CNN

- Filter w is applied to all possible windows (concatenated vectors)
- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$
- All possible windows of length h: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$



문장의 길이가 다르므로
zero-padding

Single layer CNN: Pooling layer

- New building block: Pooling
- In particular: max-over-time pooling layer
- Idea: capture most important activation (maximum over time)

- From feature map $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$

- Pooled single number: $\hat{c} = \max\{\mathbf{c}\}$

- But we want more features!

Solution: Multiple filters

- Use multiple filter weights w

- Useful to have different window sizes h

- Because of max pooling $\hat{c} = \max\{\mathbf{c}\}$, length of **c** irrelevant

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

- So we can have some filters that look at unigrams, bigrams, tri-grams, 4-grams, etc.

Multi-channel idea

- Initialize with pre-trained word vectors (word2vec or Glove)
- Start with two copies
- Backprop into only one set, keep other “static”
- Both channels are added to c_i before max-pooling

Classification after one CNN layer

- First one convolution, followed by one max-pooling
- To obtain final feature vector: $\mathbf{z} = [\hat{c}_1, \ldots, \hat{c}_m]$
  (assuming m filters w)
- Simple final softmax layer $y = softmax\left(W^{(S)}z + b\right)$

Static vs. Non-static

	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*

Non-static으로 학습시키니 word vector가 의미를 더 잘 이해하게 되었군!

Page5 Table3 (Yoon Kim, Convolutional Neural Networks for Sentence Classification, EMNLP 2014)

CNN과 Word Vector를 이용한 문장 분류

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**	–	–

Page4 Table2 (Yoon Kim, Convolutional Neural Networks for Sentence Classification, EMNLP 2014)

PPT/References from

https://www.slideshare.net/keunbongkwak/convolutional-neural-networks-for-sentence-classification
- https://www.youtube.com/watch?v=mPxi1YgU9Zw

http://web.stanford.edu/class/cs224n/lectures/lecture12.pdf
- https://www.youtube.com/watch?v=_0bOjspRG6s
- https://www.youtube.com/watch?v=Lg6MZw_OOLI&list=PL3FW7Lu3i5Jsnh1rnUwq_TcylNr7EkRe6&index=14

http://docs.likejazz.com/cnn-text-classification-tf/

https://arxiv.org/pdf/1408.5882.pdf
https://arxiv.org/pdf/1510.03820.pdf