Deep Learning for Natural Language Processing

Lecture 6 – Convolutional Neural Networks

Dr. Ivan Habernal

May 18, 2021

Trustworthy Human Language Technologies
Department of Computer Science
Technical University of Darmstadt

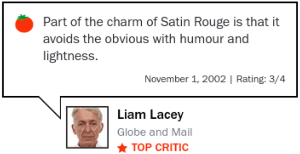
www.trusthlt.org

This lecture

Today: Classifying sentences or documents

Motivation example

Predicting the sentiment (positive, negative, or neutral)[1]



Die Another Day (2002) A thunderous ride at first, quiet cadences of pure finesse are few and far between; their shortage dilutes the potency of otherwise respectable action. Still, this flick is fun, and host to some truly excellent sequences. - Hollywood Report Card

Read More | Posted Nov 21, 2002

- Some of the words are **very informative** of the sentiment (charm, fun, excellent) other words are less informative (Still, host, flick, lightness, obvious, avoids)

- Informative clue is informative regardless of its position in the sentence

[1] `https://www.rottentomatoes.com`

Motivation

Problem 1: Variable-sized input

- standard MLP always expect the same input size

Problem 2: Relevance of words

- "to" and "a" are not very informative, but content words like "kidnapping" are important for most tasks independent of their position in the input

Problem 3: MLP may have too many parameters ("too complex models") in certain situations

This lecture

Convolution and pooling

Convolutional networks for NLP

1-D convolution operation

$$s[i] = (f * g)[i] = \sum_{m=-M}^{M} f[i-m] \cdot g[m]$$

where

- $s[i]$ is the output at position i
- f is the input representation
- $*$ is the convolution operator
- i is the current position in input
- M is the window size
- g is the filter (kernel)

Beware of many different terminologies in the literature!

1-D convolution operation example

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \end{bmatrix}$$

Input representation f Filter g (also 'kernel') Convolved output

1-D convolution operation example

$$\begin{bmatrix} \boxed{1 \;\; 1 \;\; 1} & 0 & 0 \end{bmatrix} * \begin{bmatrix} \boxed{1 \;\; 0 \;\; 1} \end{bmatrix} = \begin{bmatrix} \boxed{2} & ? & ? \end{bmatrix}$$

Input representation f Filter g Convolved output

$$\begin{bmatrix} 1 & \boxed{1 \;\; 1 \;\; 0} & 0 \end{bmatrix} * \begin{bmatrix} \boxed{1 \;\; 0 \;\; 1} \end{bmatrix} = \begin{bmatrix} 2 & \boxed{1} & ? \end{bmatrix}$$

Input representation f Filter g Convolved output

2-D convolution operation

Similar to 1-D

$$S[i,j] = (f * g)[i,j] = \sum_{m=-M}^{M} \sum_{n=-N}^{N} f[i-m, j-n] \cdot g[m,n]$$

where

- $S[i,j]$ is the output at position i, j
- f is the input representation
- $*$ is the convolution operator
- i, j is the current position in input
- M, N is the window sizes
- g is the filter (kernel)

2-D convolution operation example

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Input representation f

Filter g

(also 'kernel')

Convolved output

2-D convolution operation example

$$\begin{bmatrix} \boxed{\begin{matrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{matrix}} & \begin{matrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \end{matrix} \\ \begin{matrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{matrix} & \begin{matrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix} \end{bmatrix} * \begin{bmatrix} \boxed{\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}} \end{bmatrix} = \begin{bmatrix} \boxed{4} & ? & ? \\ ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Input representation f

Filter g

(also ‘kernel’)

Convolved output

$$\underbrace{1 \cdot 1 + 1 \cdot 0 + 1 \cdot 1}_{\text{1st row}} + \underbrace{0 \cdot 0 + 1 \cdot 1 + 1 \cdot 0}_{\text{2nd row}} + \underbrace{0 \cdot 1 + 0 \cdot 0 + 1 \cdot 1}_{\text{3rd row}} = 4$$

2-D convolution operation example

Move over all input regions

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 3 & ? \\ ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Input representation f

Filter g

(also 'kernel')

Convolved output

2-D convolution operation example

The goal is to **learn** good kernels (filter parameters)

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \\ 2 & 3 & 1 \end{bmatrix}$$

Input representation f

Filter g

(also ‘kernel’)

Convolved output

And in texts

Sentiment classification

*The **movie** was **really good**.*

*We saw this **really good movie**.*

*The **movie**, which we saw yesterday with all the colleagues in this tiny movie theater next to the bridge, was (despite my expectations) **really good**.*

For this task, position information does not really matter.

Dimensionality

Advantages of text flow

- Usually only one dimension

- as opposed to two dimensions (or even three) in images

- Convolutional networks in NLP are also called
time-delay neural networks (TDNN)

Convolutional layer in NLP

- Input sentence $\mathbf{x}_{i:i+n}$ are stacked word embeddings of dimensionality d

- Convolutional filter $\mathbf{w} \in \mathbb{R}^{h \times d}$

where h is the filter size (how many neighboring words are convoluted)

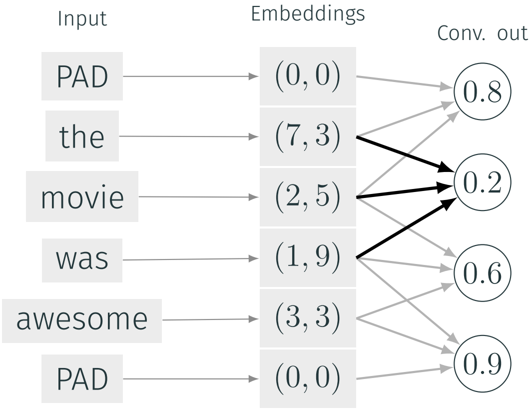
- Convolution operation $\mathbf{w} * \mathbf{x}_{i:i+n}$

- Non-linear operation

$$c_i = \mathrm{ReLU}(\mathbf{w} * \mathbf{x}_{i:i+n} + b)$$

Example

$d = 2 \qquad h = 3$



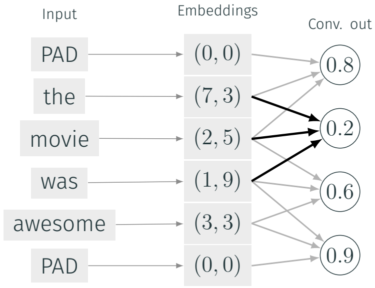
Properties of convolutional networks

Not every input is connected to every output in the following layer

- **sparse connectivity** (vs fully-connected/dense layers)

For each window, we use the same weights and bias values

- **parameter sharing**



Stride

Stride: the step size for moving over the sentence

- Stride 1 common in NLP; other values in comp. vision

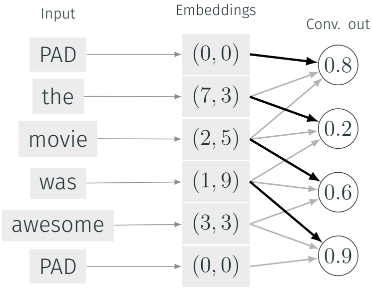


Figure 1: Stride = 1

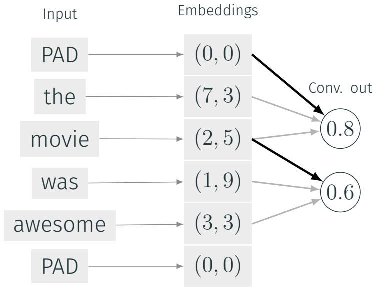


Figure 2: Stride = 2

Dense layer vs. Convolutional Layer

In principle, a convolutional layer could handle variable-sized inputs

- But in practice, it handles fixed-sized input, just like in an MLP

We usually pad with zeros so that all sequences in our data have the same length

- Sometimes we also truncate

Pooling

Pooling layer

Another new building block: pooling layer

- Idea: capture the most important activation

Let $c_1, c_2, \ldots, \in \mathbb{R}$ denote the output values of the convolutional filter

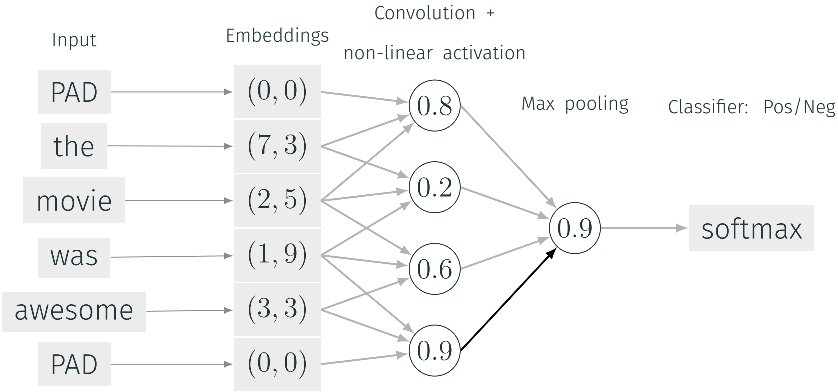
Compute output o for a **max-over-time pooling** layer as

$$o = \max_i c_i$$

Max-over-time pooling is most common in NLP. You can also find min-pooling and mean-pooling in other areas. Could also use some other averaging

Note that there are no associated weights

Classification with convolution and pooling

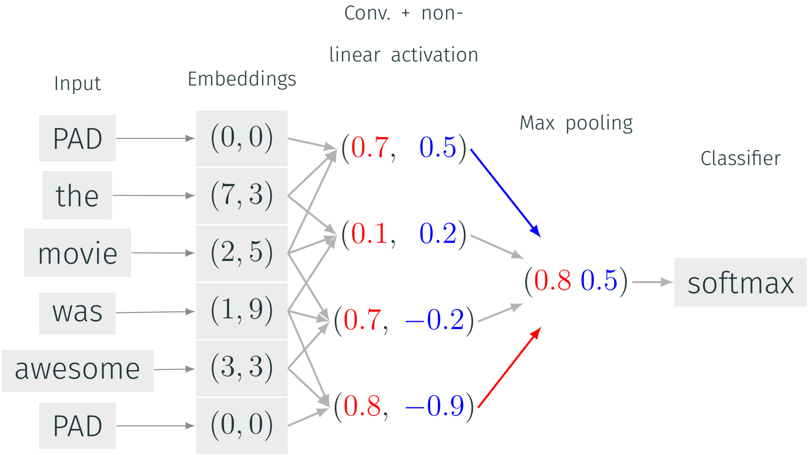


Multiple filters

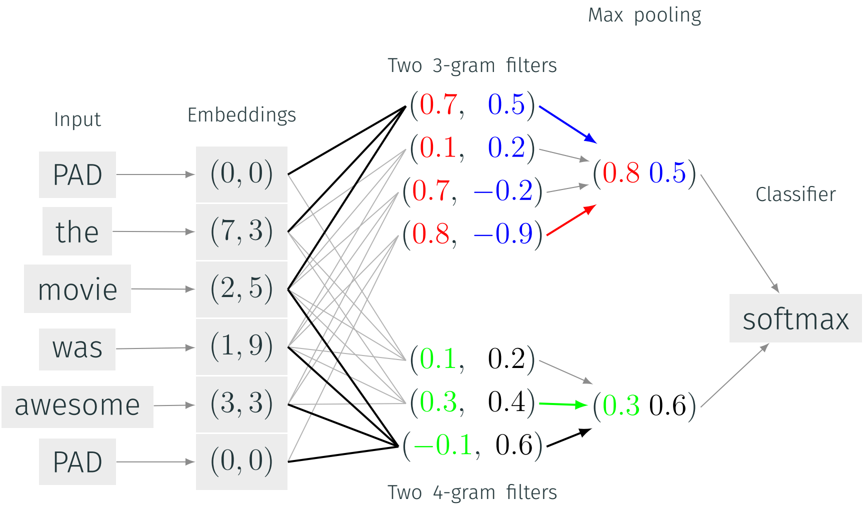
Usually we have **many** filters (hundreds or thousands), not just one

They may be of same or of different sizes

Multiple filters



Multiple filters



Properties of pooling

Idea: Extracting relevant features independent of their position in the input

- Problems:

Output remains the same if a feature occurs once or multiple times

The music was great, but the cast was horrible, the plot was horrible and the costumes were horrible.

What do CNNs learn?

Investigating CNNs[2]

In computer vision

- Effectiveness of deep CNNs can be very well explained

- Primary conv. layers detect the edges of an object

- As we go deeper, more complex features of an image are learnt

In NLP

- Not much understanding behind their success

[2] A. Madasu and V. Anvesh Rao (2019). "Sequential Learning of Convolutional Features for Effective Text Classification". In: *Proceedings of EMNLP-IJCNLP*. Hong Kong, China: Association for Computational Linguistics, pp. 5657–5666

Open questions

Popular CNN architectures use convolution with a fixed window size over the words in a sentence.

- Is sequential information preserved using convolution across words?

- What will be the effect if word sequences are randomly shuffled and convolution is applied on them?

Open questions

Previously proposed CNN architectures use max pooling operation across convolution outputs to capture most important features[3]

- Will the feature selected by max pooling always be the most important feature of the input or otherwise?

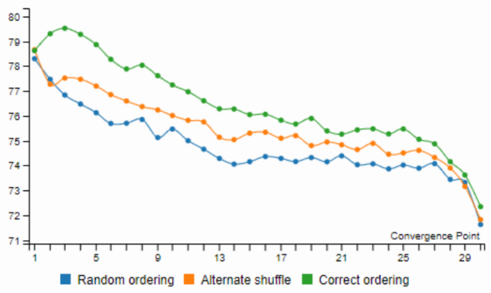
[3] Y. Kim (2014). "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of EMNLP*. Doha, Qatar: Association for Computational Linguistics, pp. 1746–1751

Experiments

Train a CNN with convolution applied on words

- Fixed window size varying from one to maximum sentence length
- Repeat this experiment with randomly shuffling
    - **Random ordering**: All the words in an input sentence
    - **Alternate shuffle**: Swap every two consecutive words, e.g. 'read book forget movie' $\rightarrow$ 'book read movie forget'

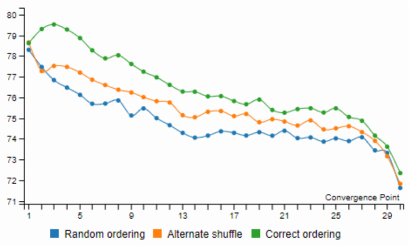
Analysis



CNNs fail to fully incorporate sequential information – performance on random ordering and correct ordering are marginally near each other

Performance with correct ordering on window size 7 ≈ random ordering on window size 1

Analysis



Increasing window size $\rightarrow$ worse in capturing sequential information

But: Performance on random ordering is still higher than other context blind algorithms like bag-of-words

CNNs still learn something valuable! What?

Experiment 2: What it learns

Train a CNN with window size 1 on sentiment classification

- Convolution acts over a single word $\rightarrow$ no ability to capture sequential information

- Words will always have same respective convolution output

- Convolution layer here acts like an embedding transformation, where input embedding space is transformed into another space

Analysis

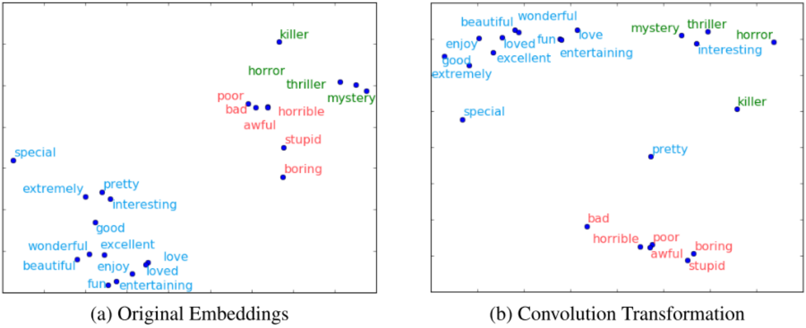


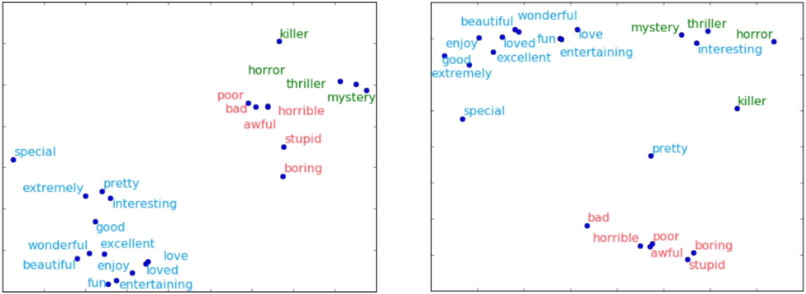
Figure 1: t-SNE projection of original embeddings and after convolution transformation

Blue = positive sentiment

Red = negative sentiment

Green = semantically close to negative sentiment words

Analysis



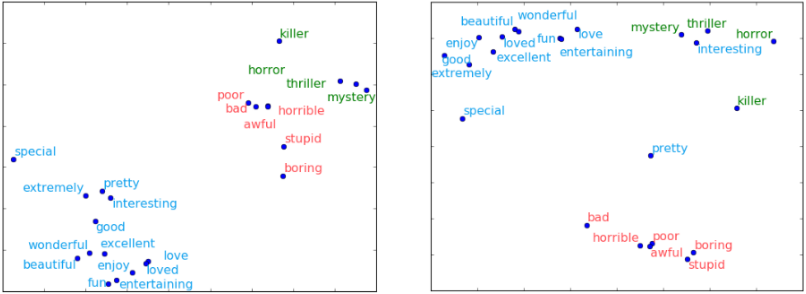
(a) Original Embeddings

(b) Convolution Transformation

Figure 1: t-SNE projection of original embeddings and after convolution transformation

Convolution layer tunes input embeddings → closer to the positive sentiment cluster than to their original semantic cluster

Analysis



(a) Original Embeddings (b) Convolution Transformation

Figure 1: t-SNE projection of original embeddings and after convolution transformation

"killer" in the original space very close to negative words "bad" and "awful" (Glove embeddings)

However in the sentiment dataset, "killer" is often used to describe a movie very positively

Take-home message

Single convolution filter output value captures a weighted summation over all the features of the original word embedding.

This enables the network to learn more task appropriate features as many as the number of filters.

A. Madasu and V. Anvesh Rao (2019). "Sequential Learning of Convolutional Features for Effective Text Classification". In: *Proceedings of EMNLP-IJCNLP*. Hong Kong, China: Association for Computational Linguistics, pp. 5657–5666

Summary

Convolutional networks can deal with variable sized input

- Sparse connectivity, parameter sharing - Narrow vs wide convolution

Pooling enables focus on most relevant features

- Max-over-time pooling

Convolutional networks for NLP

- Sentence classification

License and credits

Credits

Ivan Habernal, Steffen Eger

Content from ACL Anthology papers licensed under CC-BY `https://www.aclweb.org/anthology`

Review screenshots courtesy of `https://www.rottentomatoes.com`

References

Kim, Y. (2014). "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of EMNLP*. Doha, Qatar: Association for Computational Linguistics, pp. 1746–1751.

References ii

Madasu, A. and V. Anvesh Rao (2019). "Sequential Learning of Convolutional Features for Effective Text Classification". In: *Proceedings of EMNLP-IJCNLP*. Hong Kong, China: Association for Computational Linguistics, pp. 5657–5666.