– IN5550 –

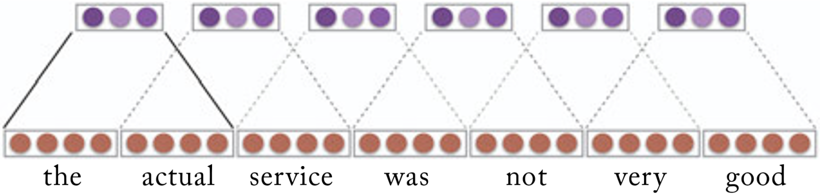
Neural Methods in Natural Language Processing

CNNs, Part 4: Additional comments and advanced options

Erik Velldal

Language Technology Group (LTG)
University of Oslo

Context and the receptive field



- CNNs improve on CBOW in also capturing ordered context.
- But still rather limited; only relationships local to windows of size k.
- Due to long-range compositional effects in natural language semantics, we'll often want to model as much context as feasible.
- One option is to just increase the filter size k.
- More powerful: a stack of convolution layers applied one after the other:
- Hierarchical convolutions.

Hierarchical convolutions

- Let $\boldsymbol{p}_{\mathbf{1}:\boldsymbol{m}} = \mathrm{CONV}^{k}_{\boldsymbol{U},\boldsymbol{b}}(\boldsymbol{w}_{\mathbf{1}:\boldsymbol{n}})$ be the result of applying a convolution (with parameters $\boldsymbol{U}$ and $\boldsymbol{b}$) across $\boldsymbol{w}_{\mathbf{1}:\boldsymbol{n}}$ with window-size k.
- Can have a succession of r layers that feed into each other:

$$\boldsymbol{p}^{\mathbf{1}}_{\mathbf{1}:\boldsymbol{m}_{\mathbf{1}}} = \mathrm{CONV}^{k_1}_{\boldsymbol{U}^{\mathbf{1}},\boldsymbol{b}^{\mathbf{1}}}(\boldsymbol{w}_{\mathbf{1}:\boldsymbol{n}})$$

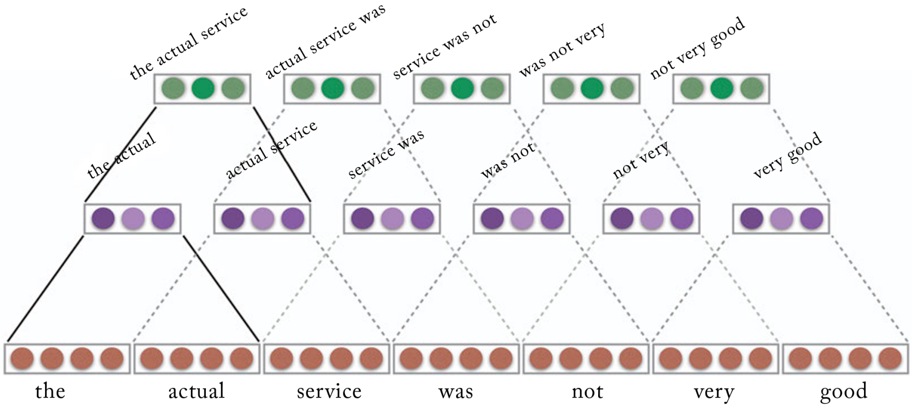
$$\boldsymbol{p}^{\mathbf{2}}_{\mathbf{1}:\boldsymbol{m}_{\mathbf{2}}} = \mathrm{CONV}^{k_2}_{\boldsymbol{U}^{\mathbf{2}},\boldsymbol{b}^{\mathbf{2}}}(\boldsymbol{p}^{\mathbf{1}}_{\mathbf{1}:\boldsymbol{m}_{\mathbf{1}}})$$

$$\ldots$$

$$\boldsymbol{p}^{\boldsymbol{r}}_{\mathbf{1}:\boldsymbol{m}_{\boldsymbol{r}}} = \mathrm{CONV}^{k^r}_{\boldsymbol{U}^{r},\boldsymbol{b}^{r}}(\boldsymbol{p}^{\boldsymbol{r}-\mathbf{1}}_{\mathbf{1}:\boldsymbol{m}_{\boldsymbol{r}-\mathbf{1}}})$$

- The vectors $\boldsymbol{p}^{r}_{\mathbf{1}:\boldsymbol{m}_{\boldsymbol{r}}}$ capture increasingly larger effective windows.

Two-layer hierarchical convolution with $k = 2$

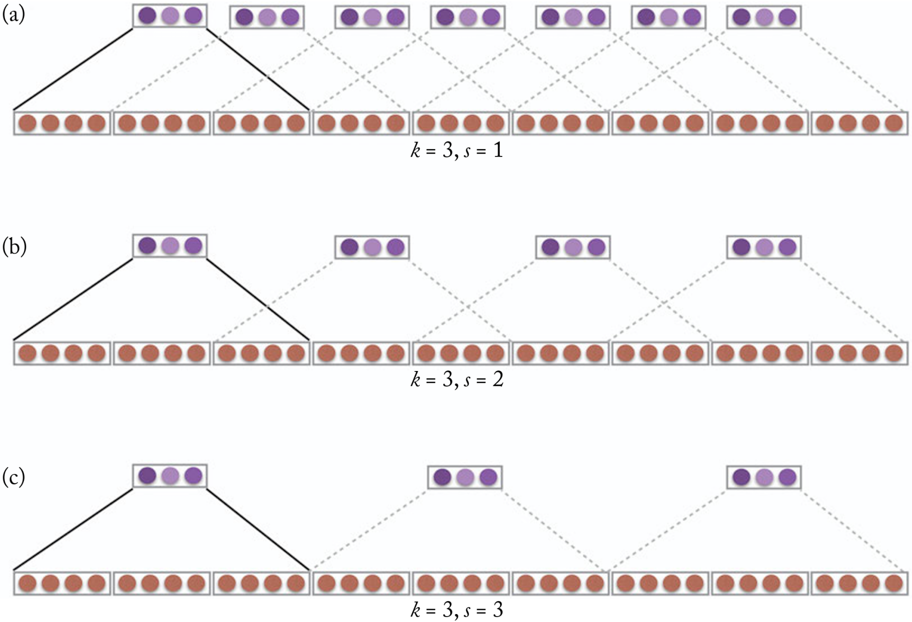


- Two different but related effects of adding layers:
- Larger receptive field wrt the input at each step: convolutions of successive layers see more of the input.
- Can learn more abstract feature combinations.

Stride

- The stride size specifies by how much we shift a filter at each step.
- So far we've considered convolutions with a stride size of 1: we slide the window by increments of 1 across the word sequence.
- But using larger strides is possible.
- Can slide the window with increments of e.g. 2 or 3 words at the time.
- A larger stride size leads to fewer applications of the filter and a shorter output sequence $\boldsymbol{p}_{\mathbf{1}:\boldsymbol{m}}$.

$k = 3$ and stride sizes 1, 2, 3



Other 'tricks'

- Dilated convolutions: skip some of the positions within the filters (or equivalently, introduce zero weights).
- Gives a wider filter region but with the same number of parameters.

Other ‘tricks’

- Dilated convolutions: skip some of the positions within the filters (or equivalently, introduce zero weights).
- Gives a wider filter region but with the same number of parameters.

- Multi-channel CNNs: each channel providing a different representation of the input. Apply convolutions to each.

Other ‘tricks’

- Dilated convolutions: skip some of the positions within the filters (or equivalently, introduce zero weights).
- Gives a wider filter region but with the same number of parameters.

- Multi-channel CNNs: each channel providing a different representation of the input. Apply convolutions to each.

- Hierarchical convolutions can be combined with parameter tying:
- Reusing the same $\boldsymbol{U}$ and $\boldsymbol{b}$ across layers.
- Allows for using an unbounded number of layers, to extend the receptive field to arbitrary-sized inputs.

Other ‘tricks’

- Dilated convolutions: skip some of the positions within the filters (or equivalently, introduce zero weights).
- Gives a wider filter region but with the same number of parameters.

- Multi-channel CNNs: each channel providing a different representation of the input. Apply convolutions to each.

- Hierarchical convolutions can be combined with parameter tying:
- Reusing the same $\boldsymbol{U}$ and $\boldsymbol{b}$ across layers.
- Allows for using an unbounded number of layers, to extend the receptive field to arbitrary-sized inputs.

- Skip-connections can be useful for deep CNNs:
- The output from one layer is passed to not only the next but also subsequent layers (ResNets, Highway nets, DenseNets, ...)

CNN use-cases

- While hugely successful in image processing, CNNs have had less impact in NLP (typically also much more shallow networks).
- Main use; document and sentence classification (e.g. for topic or polarity classification).
- Although they have also been applied to more 'structured' tasks like aspect-based SA and relation extraction.
- As of today, CNNs often applied at the character-level, to generate more robust word representations (typically concatenated with word embeddings before being passed to an RNN).

Recap: CNN Pros and Cons (1:2)

- Convolutional networks can learn to represent large n-grams efficiently…
- …without blowing up the parameter space and without having to represent the whole vocabulary.
- Parameter sharing (shared weights in all applications of a given filter)
- Multiple filters; act as specialized feature extractors.

Recap: CNN Pros and Cons (1:2)

- Convolutional networks can learn to represent large n-grams efficiently...
- ...without blowing up the parameter space and without having to represent the whole vocabulary.
- Parameter sharing (shared weights in all applications of a given filter)
- Multiple filters; act as specialized feature extractors.
- But not designed for modeling sequential language data:
- do not offer a very natural way of modeling long-range and structured dependencies.

Recap: CNN Pros and Cons (2:2)

- Lends itself well to GPU computations; optimized for matrix convolutions.
- Easily parallelized: each ‘region’ that a convolutional filter operates on is independent of the others; the entire input can be processed concurrently.
- Each filter also independent.

Recap: CNN Pros and Cons (2:2)

- Lends itself well to GPU computations; optimized for matrix convolutions.
- Easily parallelized: each 'region' that a convolutional filter operates on is independent of the others; the entire input can be processed concurrently.
- Each filter also independent.
- The cost of this is that we have to stack convolutions into deep layers in order to 'view' the entire input
- Those layers are in fact calculated sequentially...
- But the calculations at each layer happen concurrently and each individual computation is small.
- In practice, CNNs are much faster than RNNs.

Recap: CNN Pros and Cons (2:2)

- Lends itself well to GPU computations; optimized for matrix convolutions.
- Easily parallelized: each ‘region’ that a convolutional filter operates on is independent of the others; the entire input can be processed concurrently.
- Each filter also independent.
- The cost of this is that we have to stack convolutions into deep layers in order to ‘view’ the entire input
- Those layers are in fact calculated sequentially...
- But the calculations at each layer happen concurrently and each individual computation is small.
- In practice, CNNs are much faster than RNNs.