

Lecture 02: CNN for texts, embeddings for different languages

Radoslav Neychev

MADE, Moscow

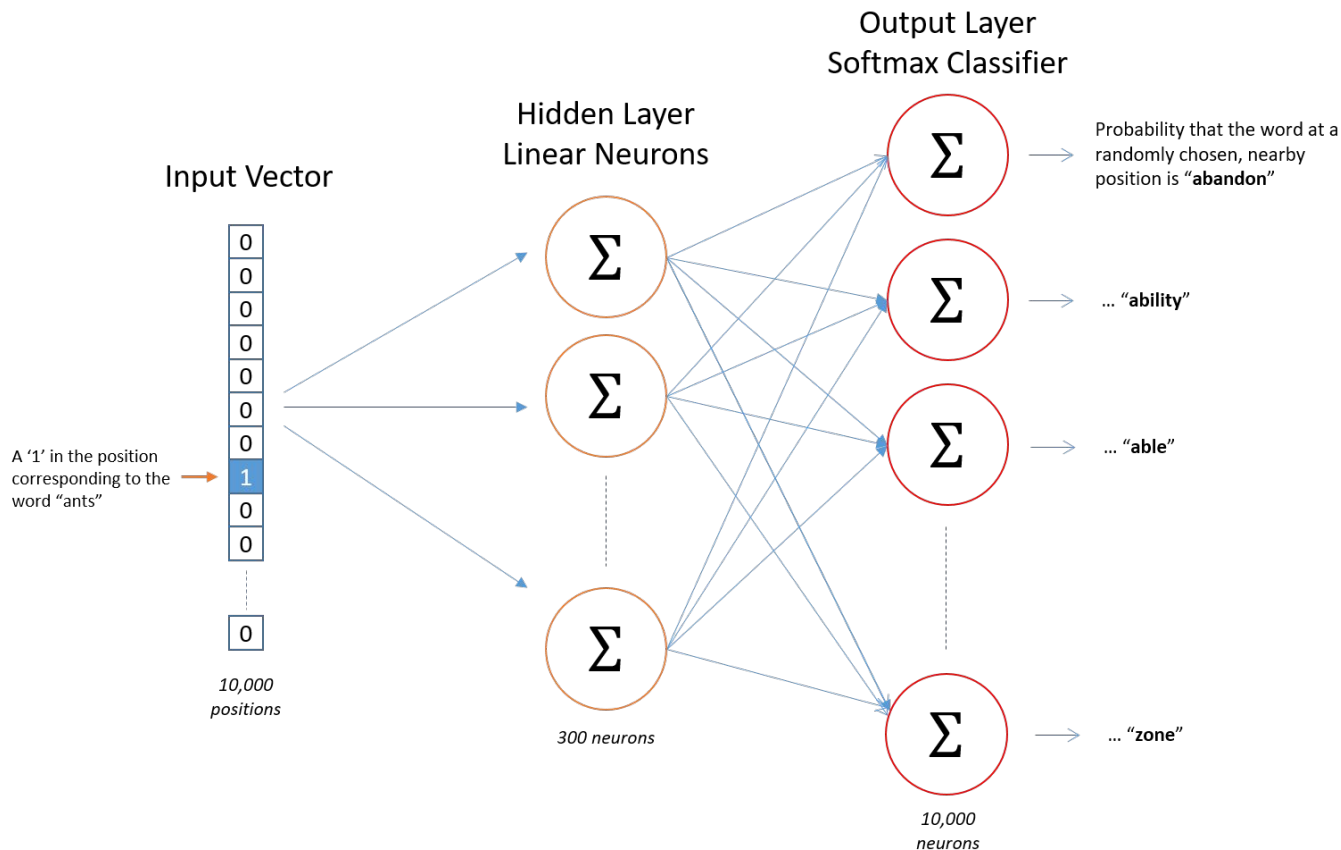
17.03.2021

- Embeddings in the wild
 - Recap
 - Unsupervised translation
- RNNs recap:
- CNNs for text processing

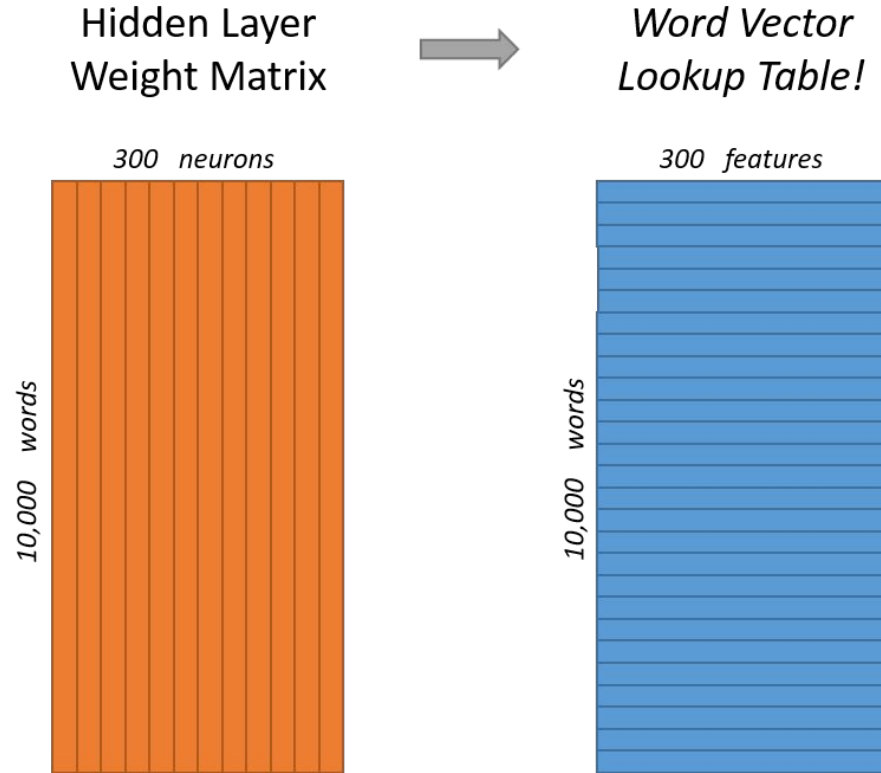
Embeddings: word2vec

Source Text	Training Samples					
<table><tr><td>The</td><td>quick</td><td>brown</td></tr></table> fox jumps over the lazy dog. ➡	The	quick	brown	(the, quick) (the, brown)		
The	quick	brown				
The <table><tr><td>quick</td><td>brown</td><td>fox</td></tr></table> jumps over the lazy dog. ➡	quick	brown	fox	(quick, the) (quick, brown) (quick, fox)		
quick	brown	fox				
The quick <table><tr><td>brown</td><td>fox</td><td>jumps</td></tr></table> over the lazy dog. ➡	brown	fox	jumps	(brown, the) (brown, quick) (brown, fox) (brown, jumps)		
brown	fox	jumps				
The <table><tr><td>quick</td><td>brown</td><td>fox</td><td>jumps</td><td>over</td></tr></table> the lazy dog. ➡	quick	brown	fox	jumps	over	(fox, quick) (fox, brown) (fox, jumps) (fox, over)
quick	brown	fox	jumps	over		

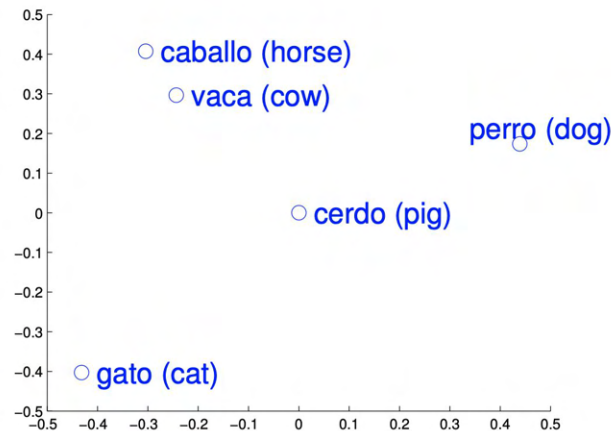
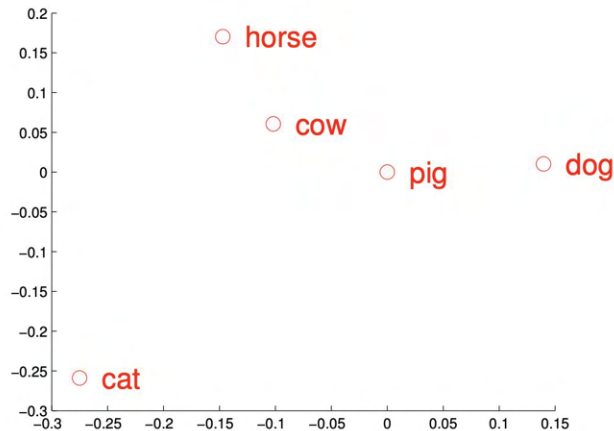
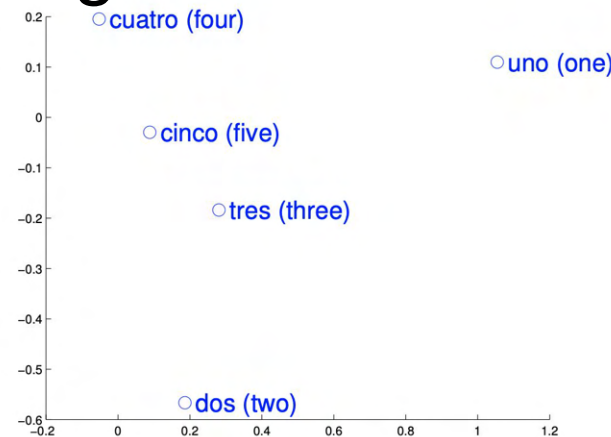
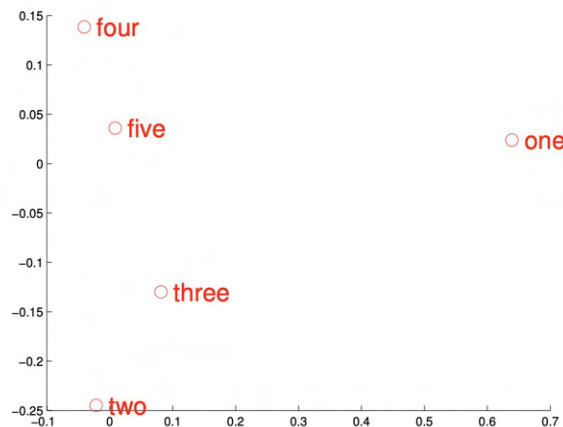
Embeddings: word2vec



Embeddings: word2vec



Word embeddings in different languages



Word embeddings in different languages

- Word embeddings are quite similar for different languages
- Assume there $n = 5000$ word-translation pairs $\{x_i, y_i\}_{i \in \{1, n\}}$
- Learn linear mapping between the source and target spaces

$$W^* = \underset{W \in M_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_F$$

- The translation of source word is $t = \operatorname{argmax}_t \cos(Wx_s, y_t)$.

Word embeddings in different languages

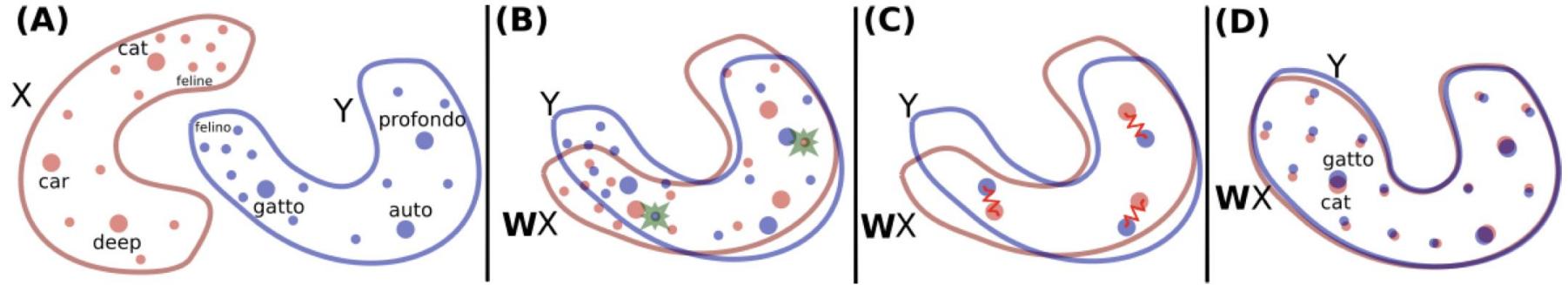
- Word embeddings are quite similar for different languages
- Assume there $n = 5000$ word-translation pairs $\{x_i, y_i\}_{i \in \{1, n\}}$
- Learn linear mapping between the source and target spaces

enforcing an orthogonality constraint on W :

$$W^* = \underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_F = UV^T, \text{ with } U\Sigma V^T = \operatorname{SVD}(YX^T).$$

- The translation of source word is $t = \operatorname{argmax}_t \cos(Wx_s, y_t)$.

Word embeddings in different languages



Comment: mapping between two languages can be done completely in unsupervised manner with GANs.

We will meet later.

More info available in the mentioned paper:

Source: [Word translation without parallel data, ICLR 2018](#)

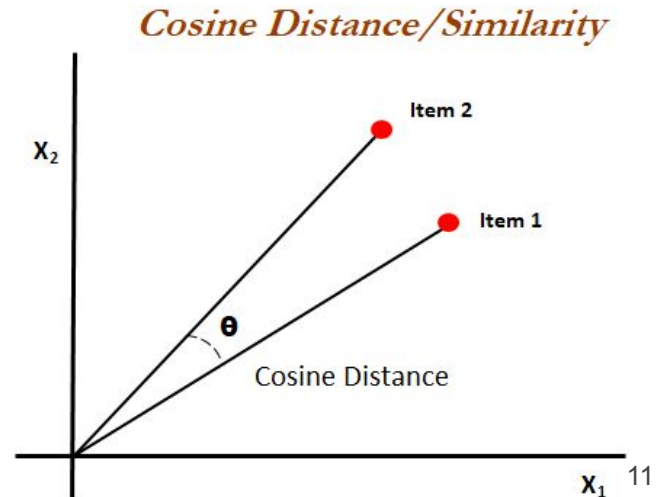
Why cosine distance/similarity?

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Cosine distance focuses on angle between the vectors.

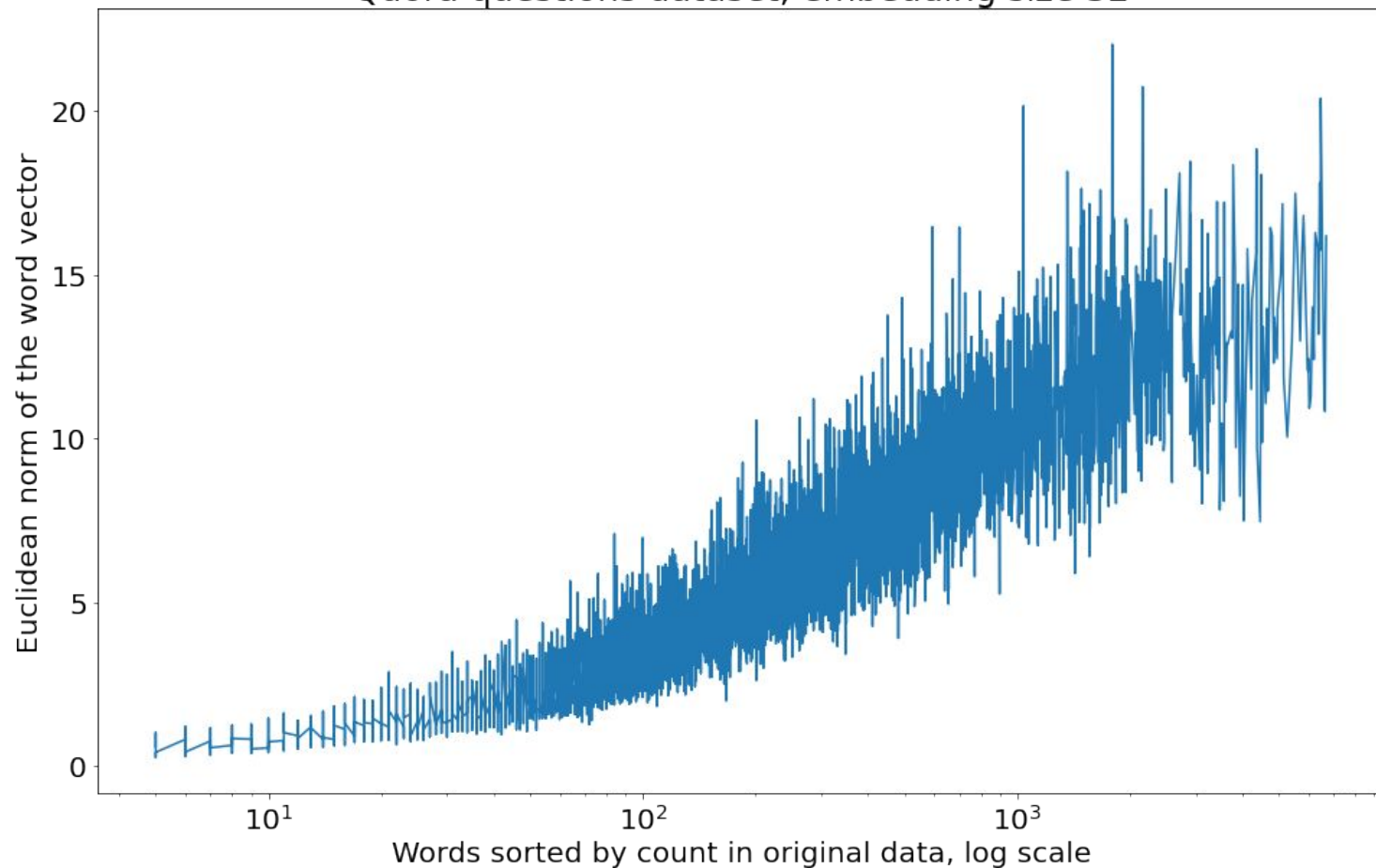
With count-based approaches (e.g. BOW)
it is really useful.

With word embeddings it is useful as well.



How word frequency affects the embedding vector norm

Quora questions dataset, embedding size 32



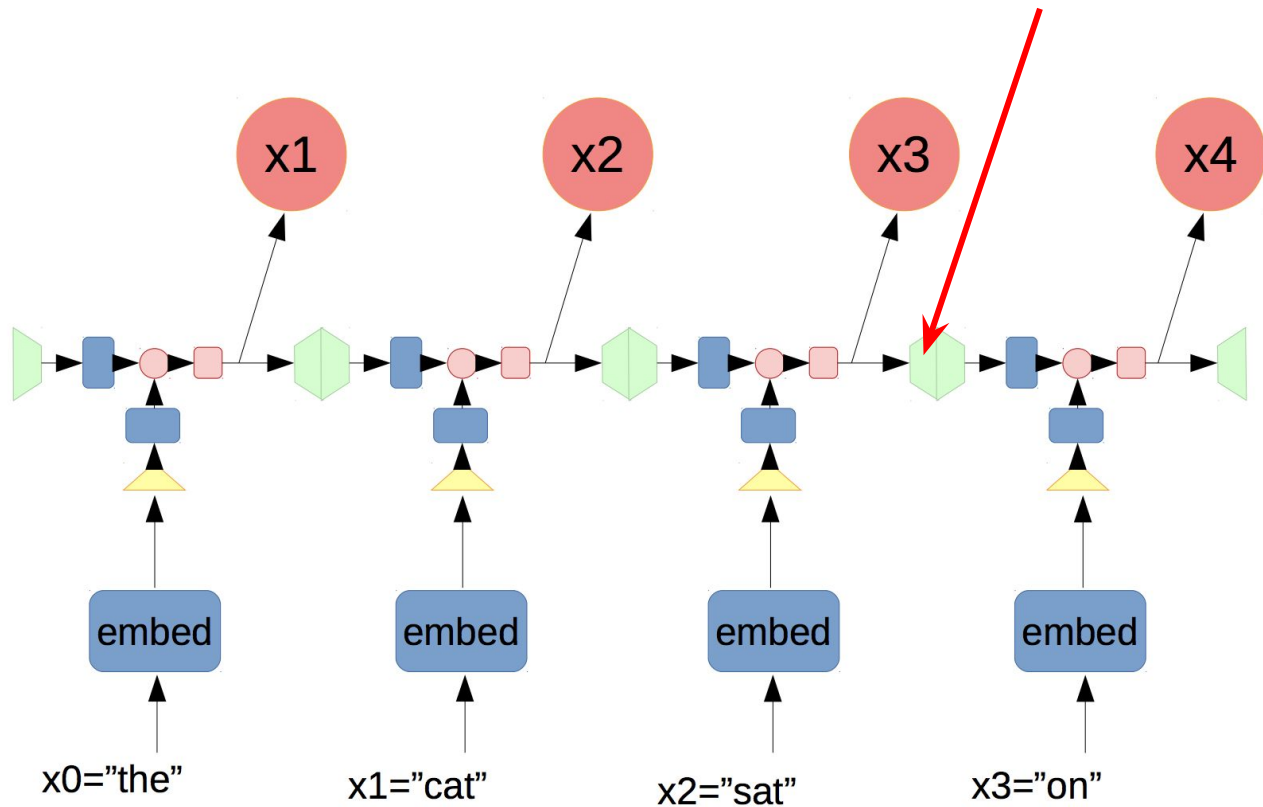
Vector norms for words with no specific context

word	count	vector norm
-----	-----	-----
overheat	11	0.81233
enormous	12	0.807057
dog	1212	11.2591
cat	1545	10.3738
laptop	1906	14.5192
phone	4124	15.7901
a	155726	11.4656
the	252068	8.47355

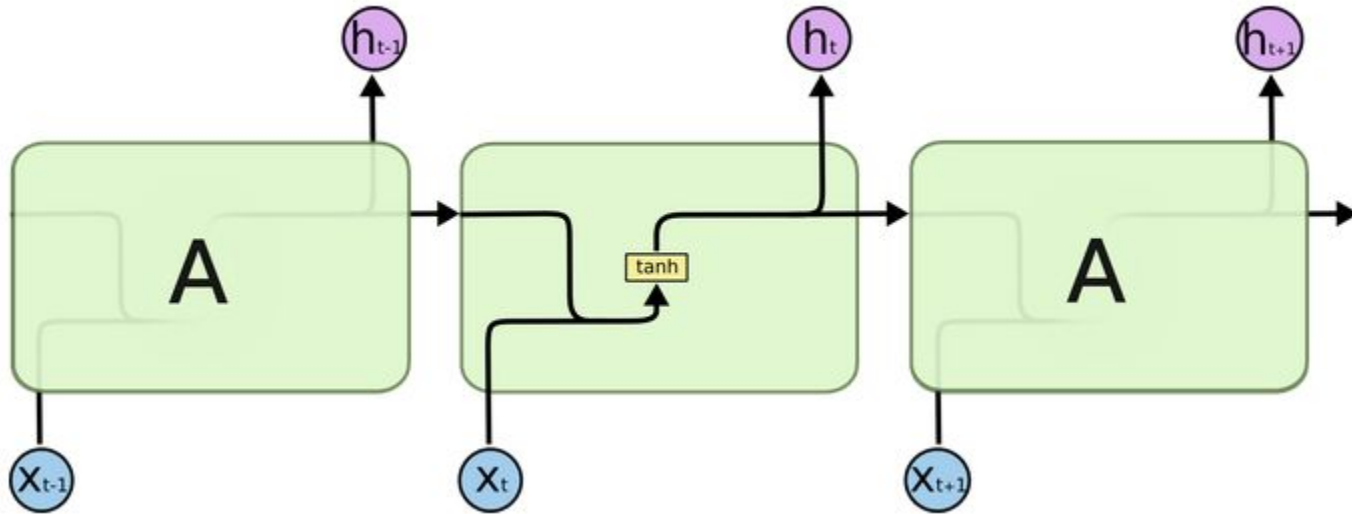
How to deal with texts?

Recap: RNN

Here is the embedding
for phrase [x0, x1, x2]

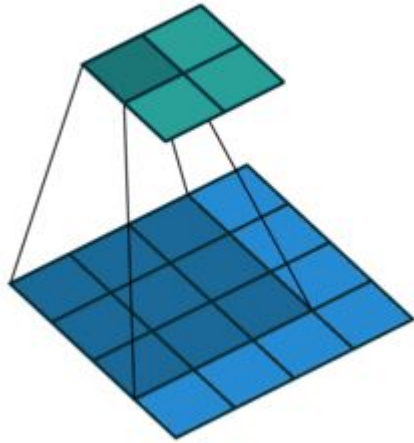


Recap: Vanilla RNN

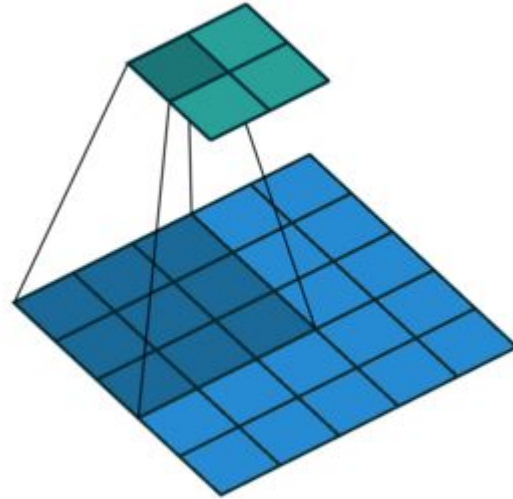


Applying CNNs to texts

CNN simple recap

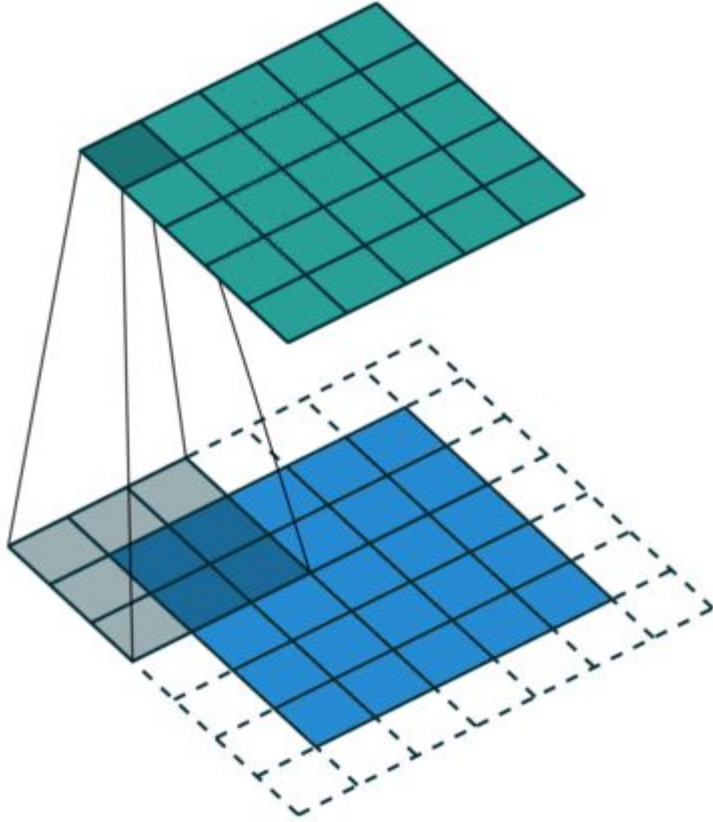


No padding,
no strides

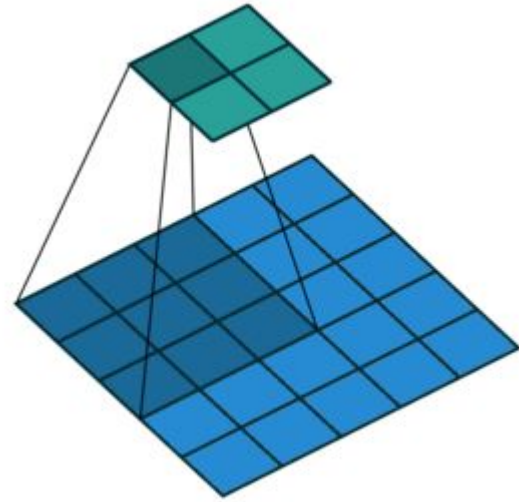


No padding,
with strides

CNN simple recap

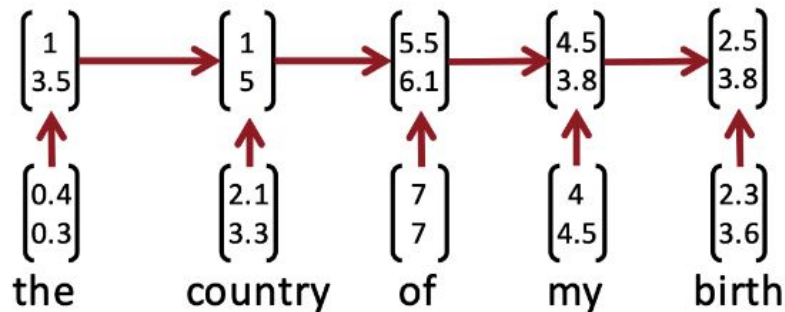


With padding,
no strides



No padding,
with strides

From RNN to CNN

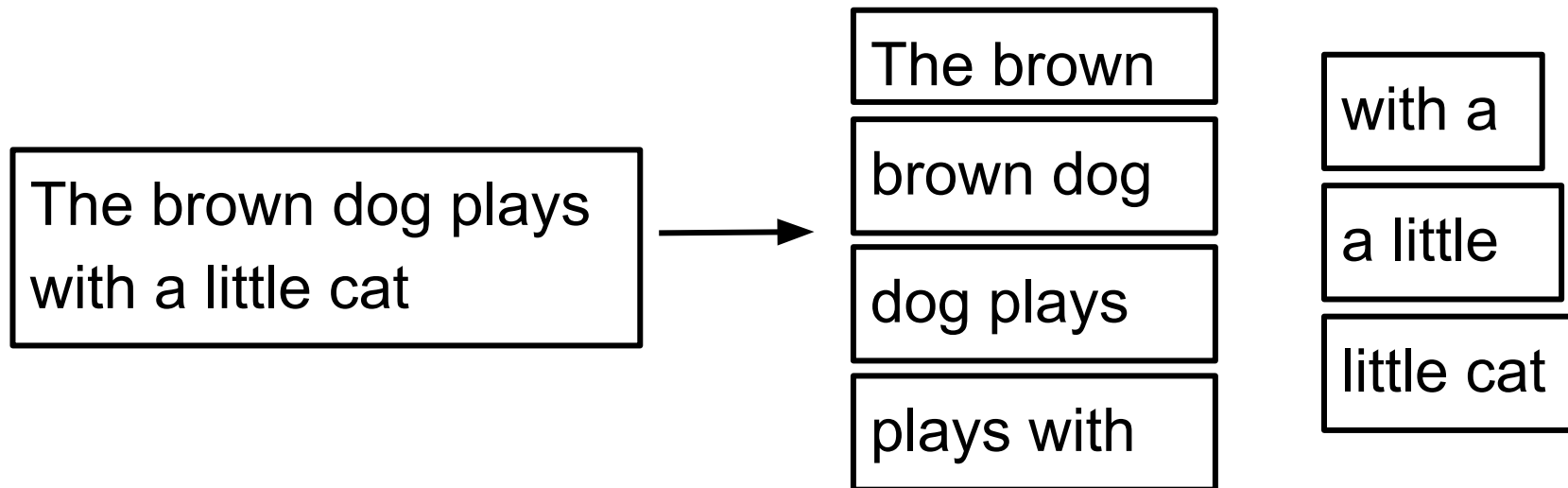


- Recurrent neural nets can not capture phrases without prefix context and often capture too much of last words in final vector

From RNN to CNN

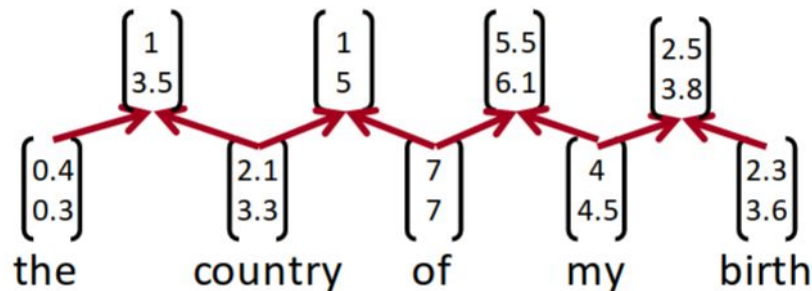
- RNN: Get compositional vectors for grammatical phrases only
- CNN: What if we compute vectors for every possible phrase?
 - Example: “*the country of my birth*” computes vectors for:
 - *the country, country of, of my, my birth, the country of, country of my, of my birth, the country of my, country of my birth*
- Regardless of whether it is grammatical
- Wouldn't need parser
- Not very linguistically or cognitively plausible

Recap: n-gramms



From RNN to CNN

- Imagine using only bigrams



- Same operation as in RNN, but for every pair

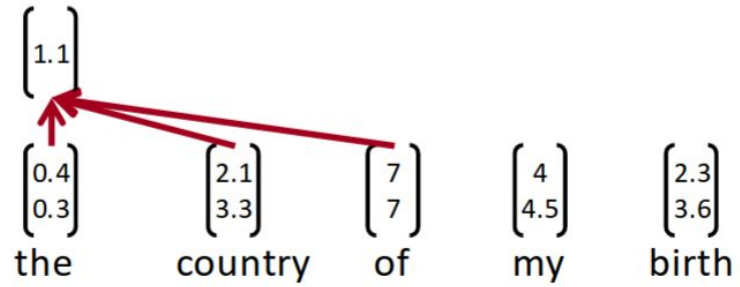
$$p = \tanh \left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b \right)$$

- Can be interpreted as convolution over the word vectors

One layer CNN

- Simple convolution + pooling
- Window size may be different (2 or more)
- The feature map based on bigrams:

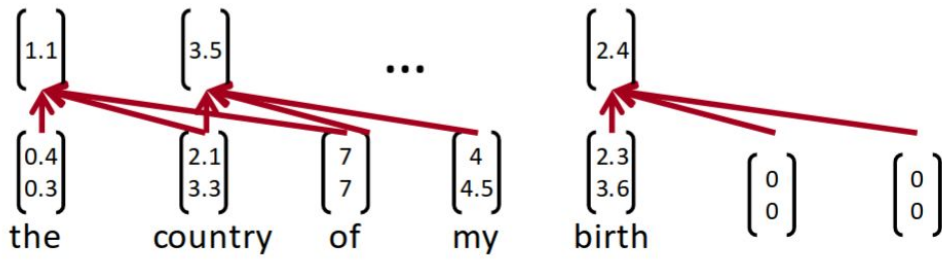
$$c_i = f(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b)$$



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

What's next?

We need more features!



One layer CNN

- Feature representation is based on some applied filter:

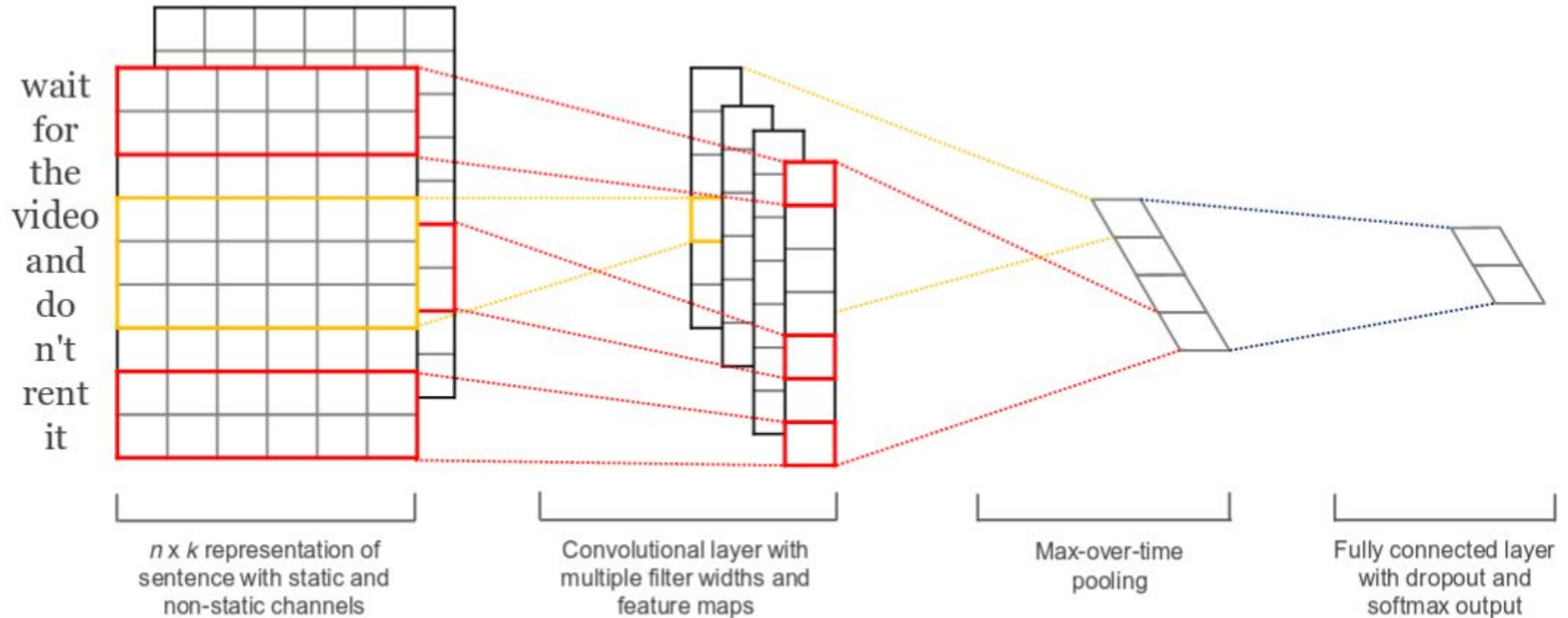
$$\mathbf{c} = [c_1, c_2, \dots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$$

- Let's use pooling:

$$\hat{c} = \max\{\mathbf{c}\}$$

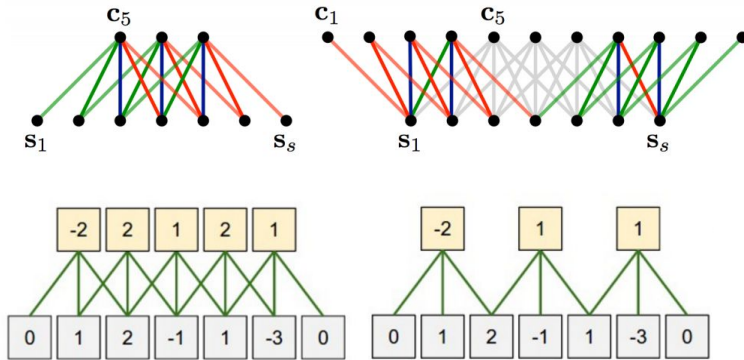
- Now the length of \mathbf{c} is irrelevant!
- So we can use filters based on unigrams, bigrams, tri-grams, 4-grams, etc.

Another example from Kim (2014) paper

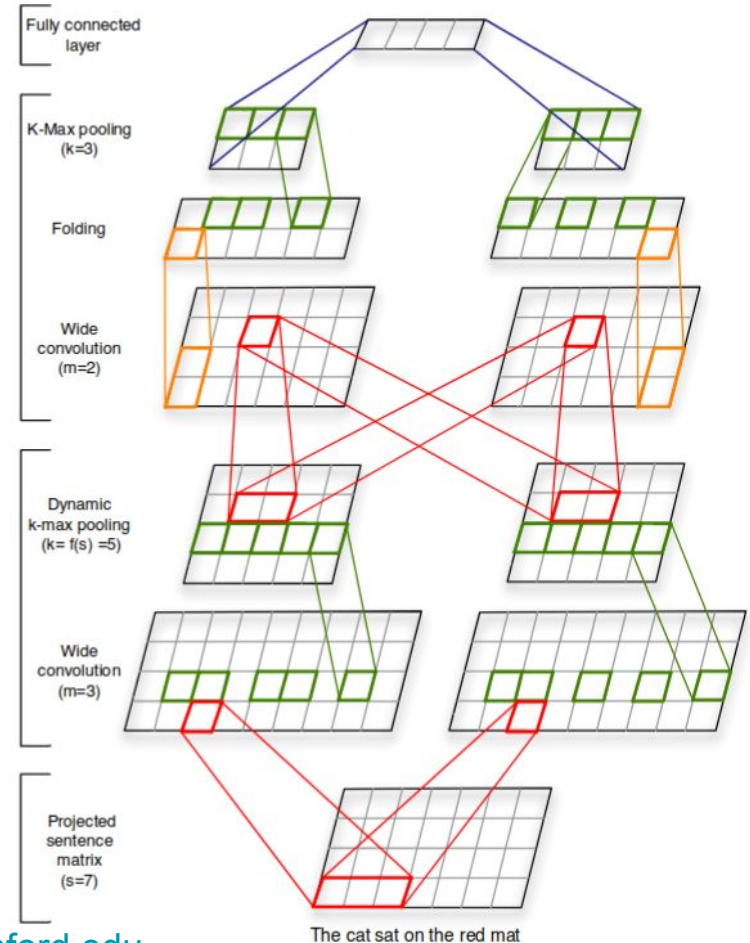


More about CNN

- Narrow vs wide convolution (stride and zero-padding)

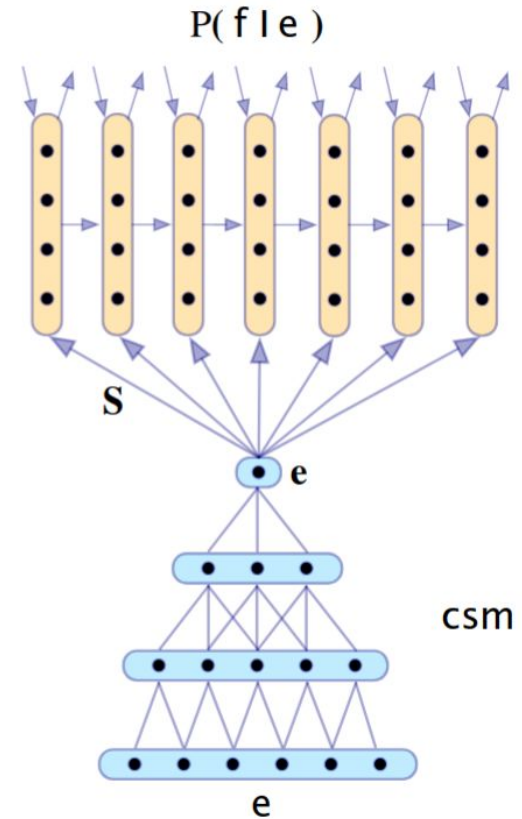


- Complex pooling schemes over sequences
- Great readings (e.g. Kalchbrenner et. al. 2014)



CNN applications

- Neural machine translation: CNN as encoder, RNN as decoder
- One of the first neural machine translation efforts
- Paper: [Recurrent Continuous Translation Models, Kalchbrenner and Blunsom, 2013](#)



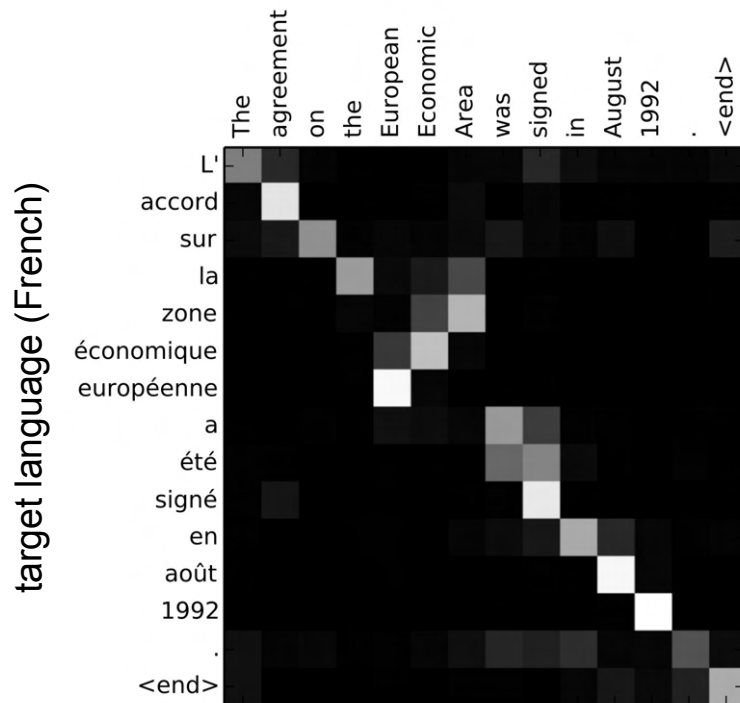
Approaches comparison

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	—	—

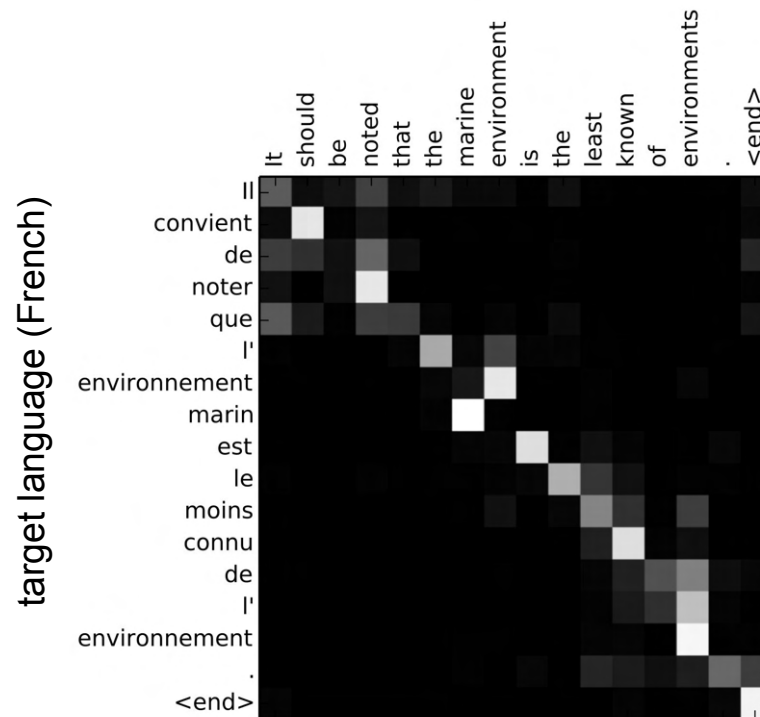
- Vanishing gradient is present not only in RNNs
 - Use some kind of memory or skip-connections
- LSTM and GRU are both great
 - GRU is quicker, LSTM catch more complex dependencies
- Clip your gradients
- Using CNNs for texts is similar to n-gramm trick
- CNNs are more effective in case of massive computations
- Combining RNN and CNN worlds? Why not

Attention outro

source language (English)



source language (English)



Word2vec embeddings capture only **local** context