ALTEGRAD 2020

Deep Learning architectures for NLP II

M. Vazirgiannis

https://bit.ly/2rwmvQU

LIX, Ecole Polytechnique

November 2020

OUTLINE

- **CNN, RNNs, LSTMs**
- Attention/Context based architectures
  - ELMO, BERT. BART

Convolutional Neural Networks

- In 1995, Yann LeCun and Yoshua Bengio introduced the concept of convolutional neural networks.
- Neuro-biologically motivated by the findings of locally sensitive and orientation-selective nerve cells in the visual cortex.
- They designed a network structure that implicitly extracts relevant features.
- Convolutional Neural Networks are a special kind of multi-layer neural networks.

[2] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. 3, 4

Convolutional Neural Networks

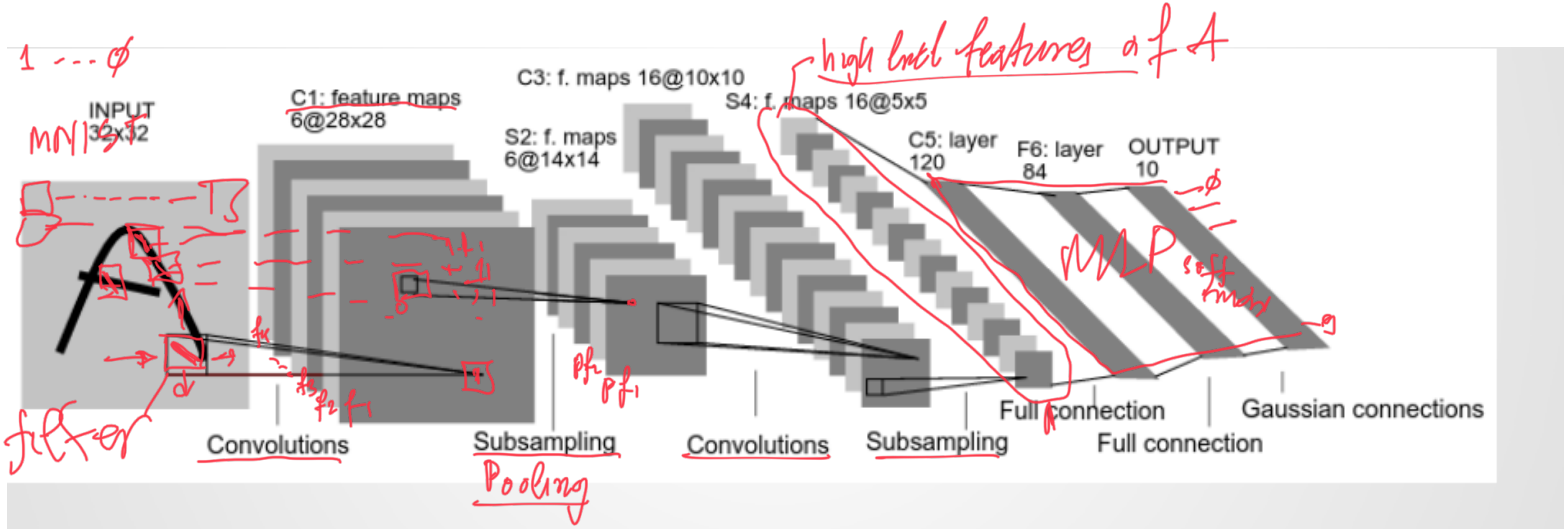
- inspired by studies of the cat's visual cortex [1], developed in computer vision to work on regular grids such as images [2].
- Feed-forward NNs , each neuron receives input from a neighborhood of the neurons (receptive fields) in the previous layer.
- Receptive fields, allow CNNs to recognize complex patterns in a hierarchical way, by combining lower-level, elementary features into higher-level features *compositionality*.
  - raw pixels=> edges =>shapes =>objects.

- absolute positions of features in the image are not important – only useful respective positions is useful composing higher-level patterns.
- Model detect a feature regardless of its position in the image - local invariance.
- Compositionality, local invariance two key concepts of CNNs.

[1] Hubel, David H., and Torsten N. Wiesel (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology 160.1:106-154. 4

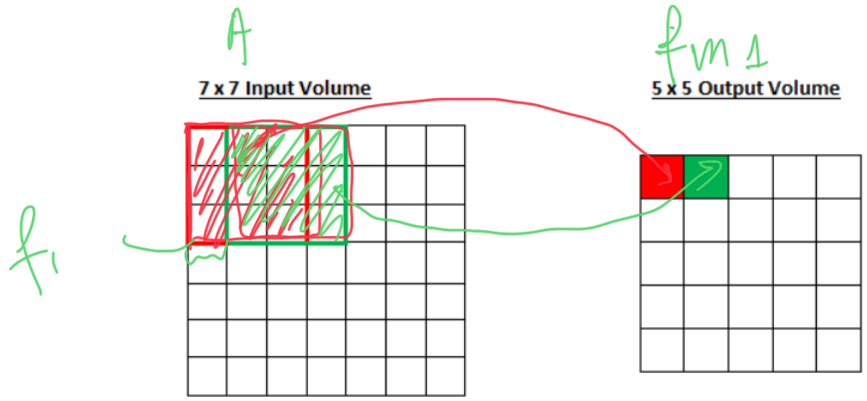
[2] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. 3, 4

CNN architecture

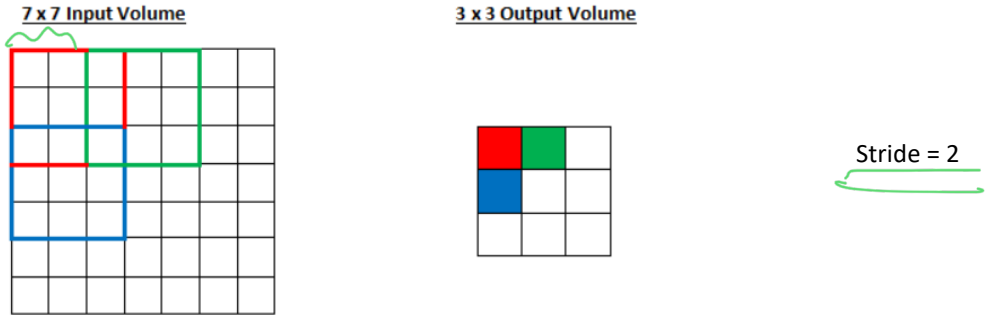


Lenet-5 (Lecun-98), Convolutional Neural Network for digits recognition

Convolution Example

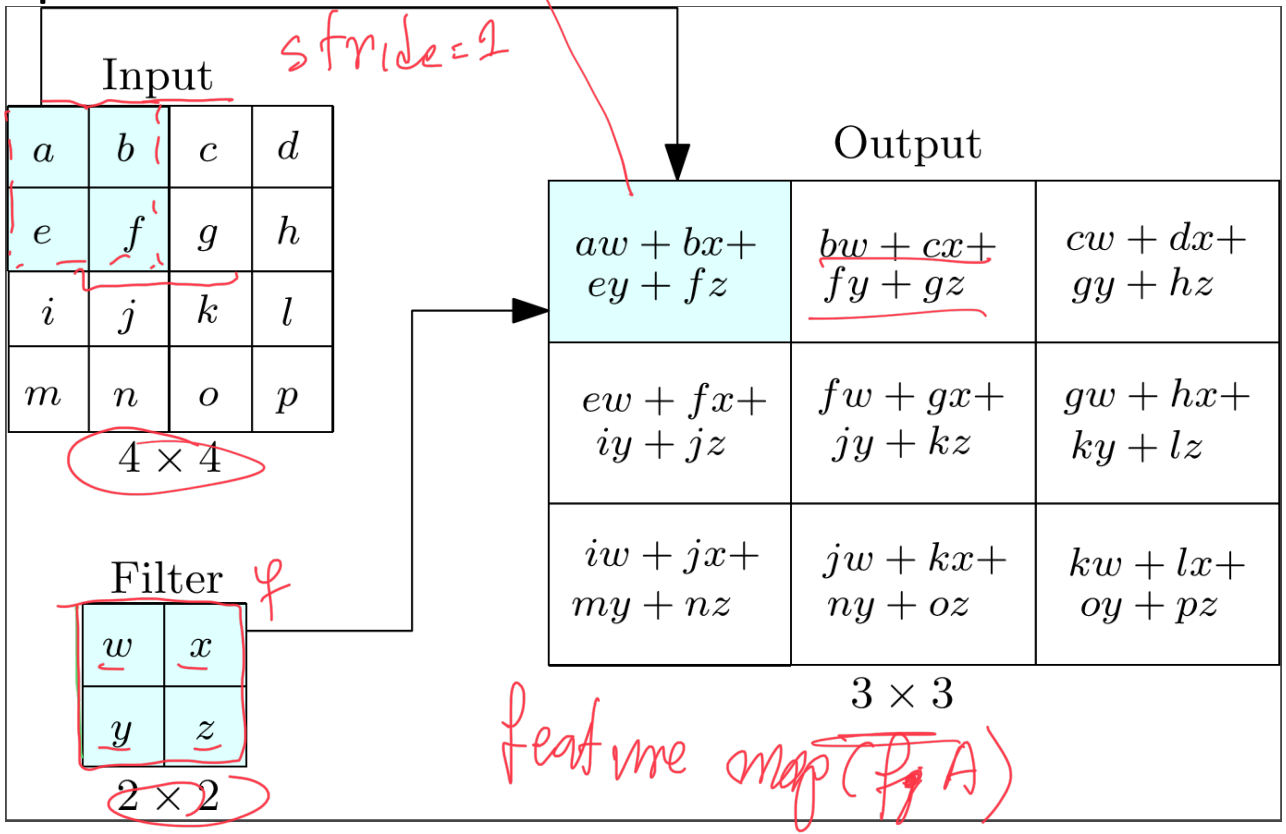


Stride = 1

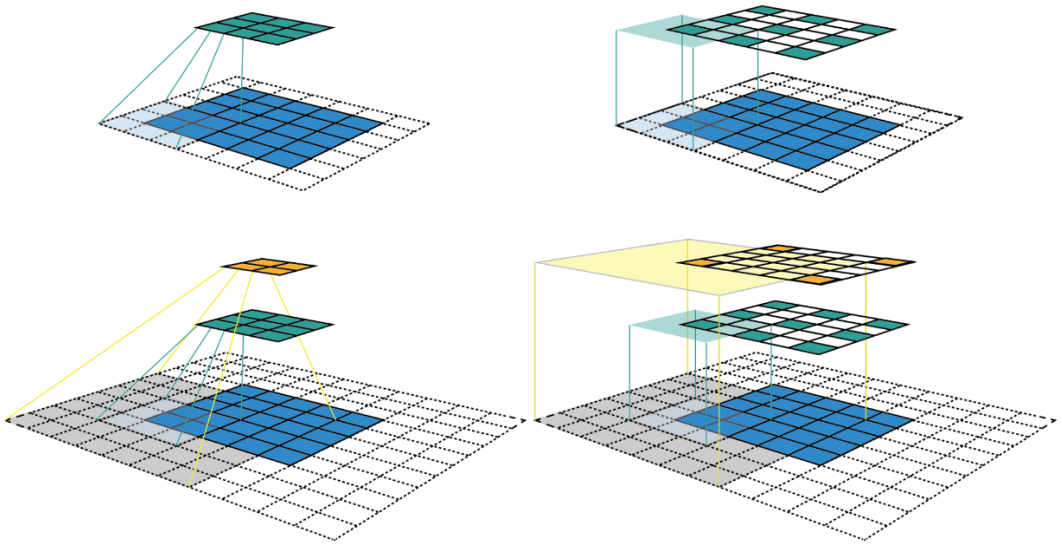


Stride = 2

Example


Input
a b c d
e f g h
i j k l
m n o p
4 × 4
Filter
w x
y z
2 × 2
Output
aw + bx + ey + fz
bw + cx + fy + gz
cw + dx + gy + hz
ew + fx + iy + jz
fw + gx + jy + kz
gw + hx + ky + lz
iw + jx + my + nz
jw + kx + ny + oz
kw + lx + oy + pz
3 × 3

CNN feature maps



Why Convolution?

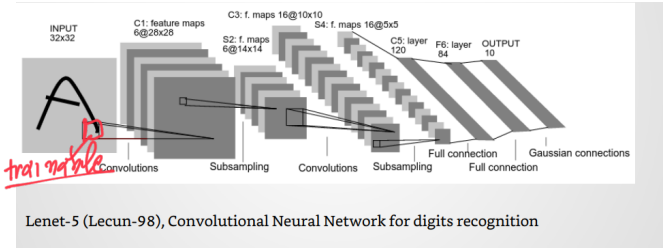
- Convolution exploits the property of spatial local correlations in the feature space by enforcing local connectivity pattern between neurons of adjacent layers

- Drastic reduce in the number of free parameters compared to fully connected network reducing overfitting and more importantly, computational complexity of the network

Feature maps

- Feature Map - Obtained by convolution of the feature matrix with a linear filter, adding a bias term and applying a non-linear function
- Non-linear functions:
  - Sigmoid $\frac{1}{1+e^{-x}}$
  - Tanh $\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
  - Rectified Linear Unit (ReLU) -> Most popular choice avoids saturation issues, makes learning faster

$$f(x) = \max(0, x)$$

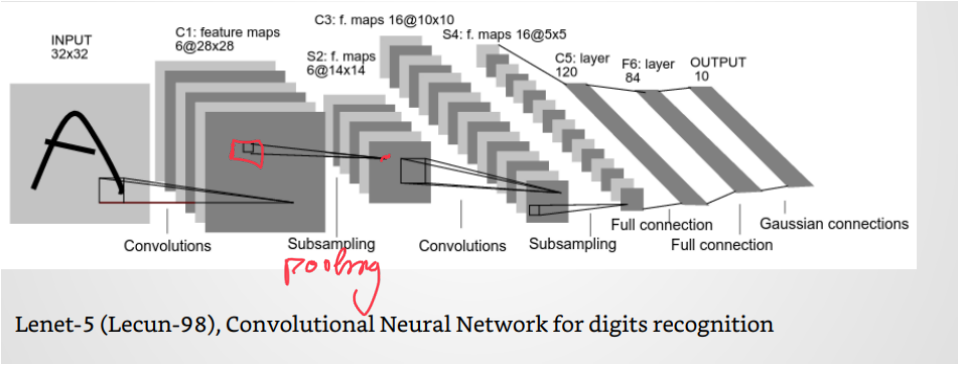


Lenet-5 (Lecun-98), Convolutional Neural Network for digits recognition

- Require a number of such feature maps at each layer to capture sufficient features

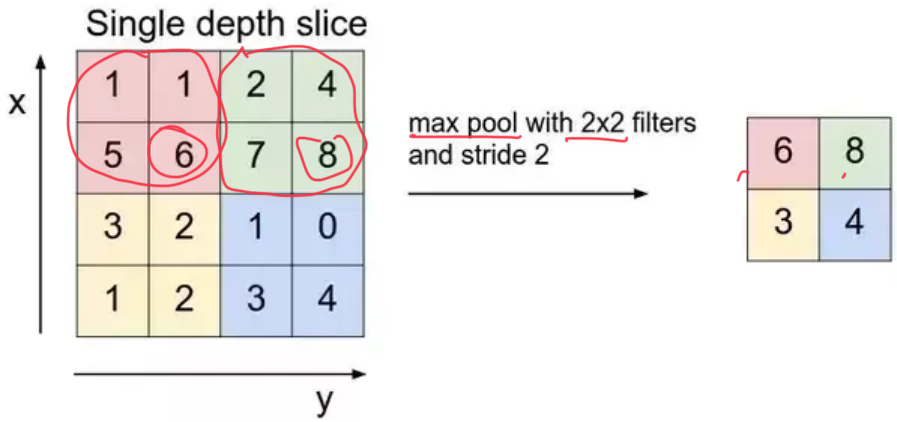
Pooling

- Sub-sampling layer
- Variants:
  - Max pooling
  - Weighted average
  - L2 norm of neighborhood
- Provides translation invariance
- Reduces computation



Lenet-5 (Lecun-98), Convolutional Neural Network for digits recognition

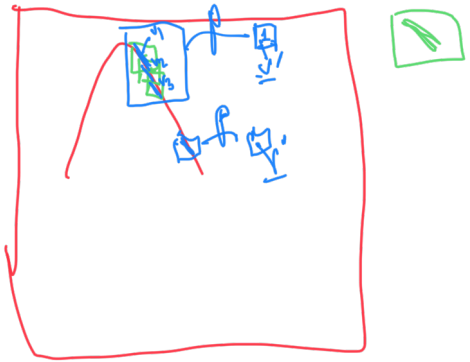
Max pooling - Example



See demo: http://cs231n.github.io/understanding-cnn/

How to reduce overfitting in CNNs

- Dropout: Randomly set the output value of network neurons to 0
  - Works as a regularization alternative
- Weight decay: keeps the magnitude of weights close to zero
- Data Augmentation: Slightly modified instances of the data



CNN for Text Classification

- Use the word embeddings of the document terms as input for Convolutional Neural Network
- Input must be fixed size
- Applies multiple filters to concatenated word vectors
- Produces new features for every filter
- picks the max as a feature for the CNN

CNN architecture for document classification

- Use the high quality embeddings as input for Convolutional Neural Network
- Applies multiple filters to concatenated word vector

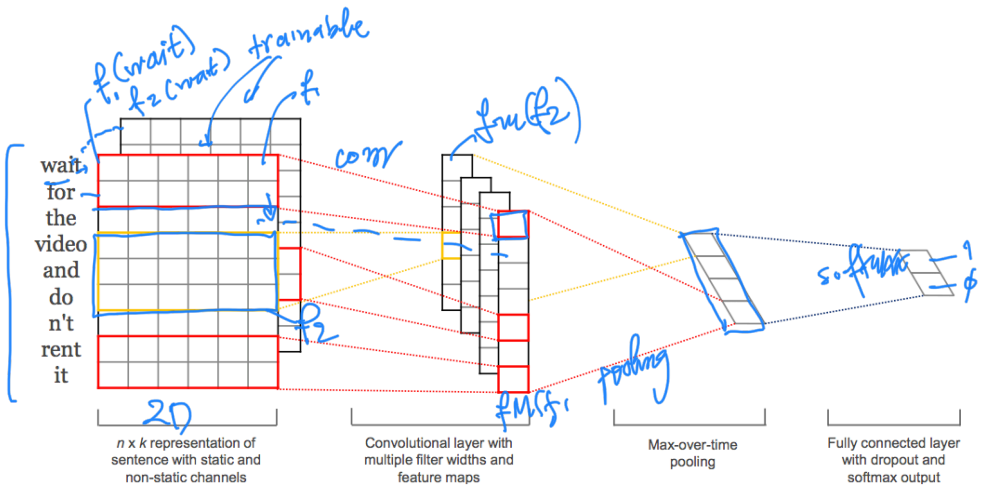
$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$$

- Produces new features for every filter

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

- And picks the max as a feature for the CNN

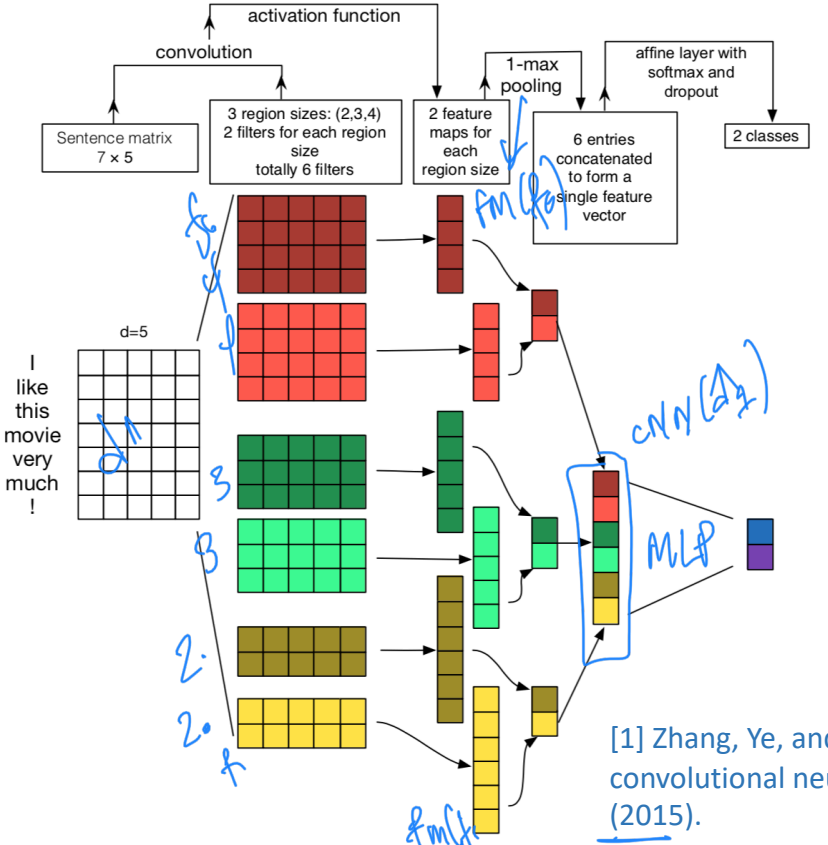
$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \qquad \hat{c} = \max\{\mathbf{c}\}$$



Yoon Kim - Convolutional Neural Networks for Sentence Classification

CNN architecture for document classification

[1]



- Data (text) only 1st column of input
- Rest of each row: embedding (in images 2D+RGB dimension)
- Filters of different sizes (4x5, 3x5 etc.)
  - Each size captures different features (need ~10^2 filters/size)
- Feature maps:
  - As many as the times filter fits on data matrix
- Max pooling maintains the “best features”
- Global feature map => classification via softmax

[1] Zhang, Ye, and Byron Wallace. "A sensitivity analysis of (and practitioners’ guide to) convolutional neural networks for sentence classification." arXiv preprint arXiv:1510.03820 (2015).

CNN for text classification

Many variations of the model [1]

- use existing vectors as input (CNN-static)
- learn vectors for the specific classification task through backpropagation (CNN-rand)
- Modify existing vectors for the specific task through backpropagation(CNN-non-static)

[1] Y. Kim, Convolutional Neural Networks for Sentence Classification, EMNLP 2014

CNN for text classification

- Combine multiple word embeddings
- Each set of vectors is treated as a ‘channel’
- Filters applied to all channels
- Gradients are back-propagated only through one of the channels
- Fine-tunes one set of vectors while keeping the other static

CNN for text classification

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**	–	–

Accuracy scores (Kim et al vs others)

CNN architecture for (short) document classification – T-SNE visualization (see Lab notes)

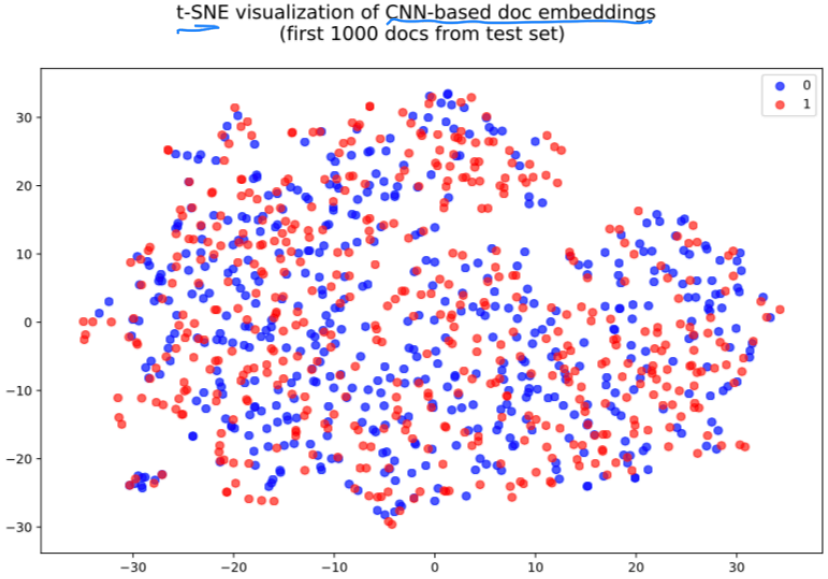


Figure 2: Doc embeddings before training.

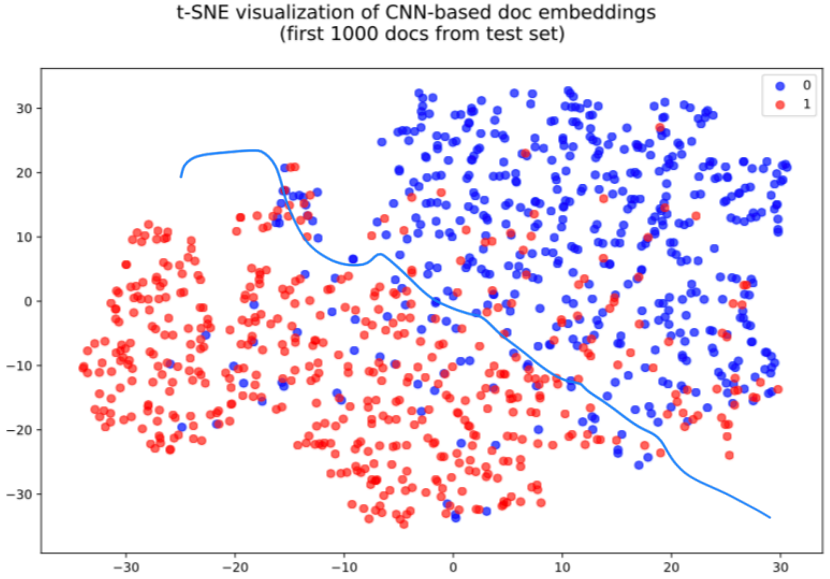


Figure 3: Doc embeddings after 2 epochs.

CNN architecture for document classification- Saliency maps (see Lab notes)

- words are most related to changing the doc classification
- A in R^{sxd}, s:# sentence words, d:size of embeddings

$$\text{saliency}(a) = \left| \frac{\partial(\text{CNN})}{\partial a} \Big|_a \right|$$

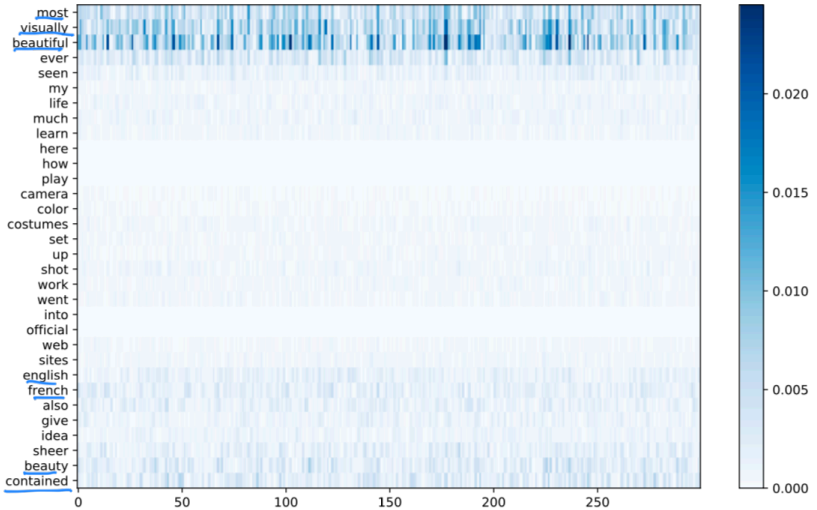


Figure 4: Saliency map for document 1 of the IMDB test set (true label: positive)

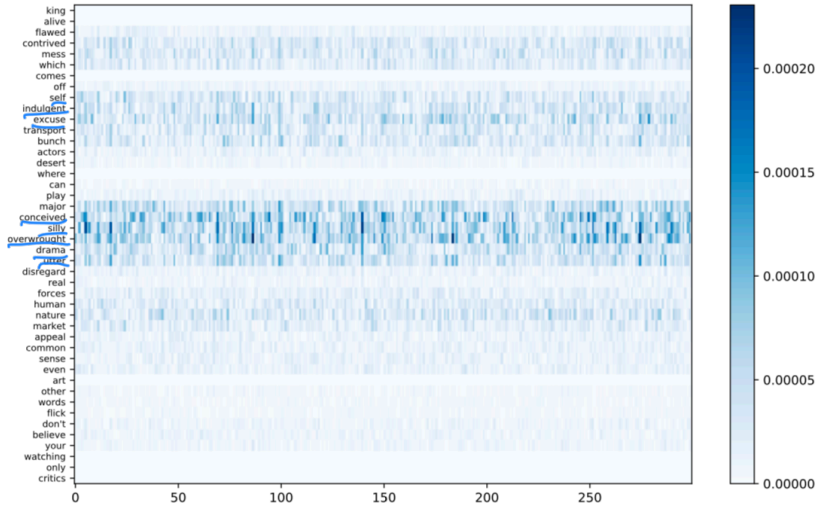
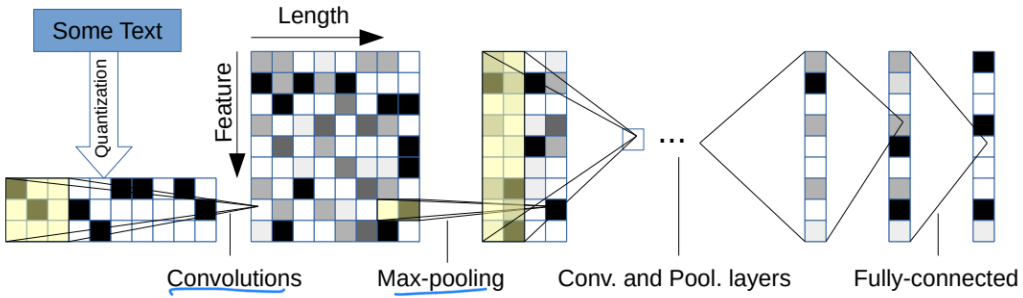


Figure 5: Saliency map for document 15 of the IMDB test set (true label: negative)

Character-level CNN for Text Classification

- Input: sequence of encoded characters
- quantize each character using “one-hot” encoding
- input feature length is 1014 characters
- 1014 characters able capture most of the texts of interest
- Also perform Data Augmentation using Thesaurus as preprocessing step

Model Architecture



- 9 layers deep
- 6 convolutional layers
- 3 fully-connected layers
- 2 dropout modules in between the fully-connected layers for regularization

Model Comparison

Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63
Lg. Lk. Conv.	8.55	4.95	1.72	4.89	40.52	29.06	45.95	5.84
Sm. Lk. Conv.	10.87	4.93	1.85	5.54	41.41	30.02	43.66	5.85
Lg. Lk. Conv. Th.	8.93	-	1.58	5.03	40.52	28.84	42.39	5.52
Sm. Lk. Conv. Th.	9.12	-	1.77	5.37	41.17	28.92	43.19	5.51
Lg. Full Conv.	9.85	8.80	1.66	5.25	38.40	29.90	40.89	5.78
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

variants of char. model

Testing errors for all models

Blue->best, Red->worst

links

- http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
- https://arxiv.org/pdf/1509.01626.pdf
- http://www.aclweb.org/anthology/D14-1181
- http://cs231n.github.io/convolutional-networks/
- http://ufldl.stanford.edu/tutorial/supervised/Pooling/

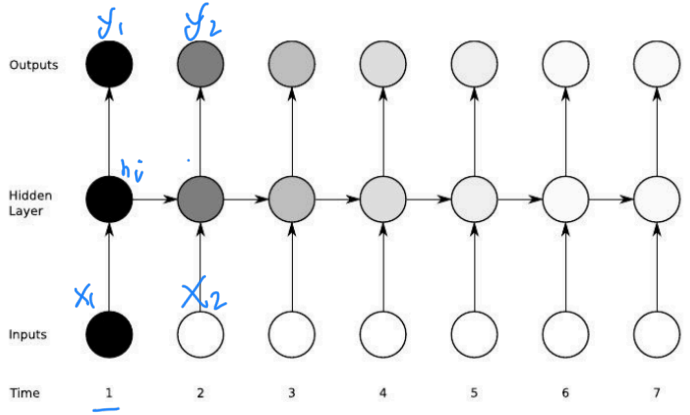
Recurrent Neural Networks

- CNNs are good at image classification
  - Input: an image
  - Output: probability of the class
    - $p(Red\ Panda|\ image) = 0,9$
    - $p(Cat\ |\ image) = 0,1$

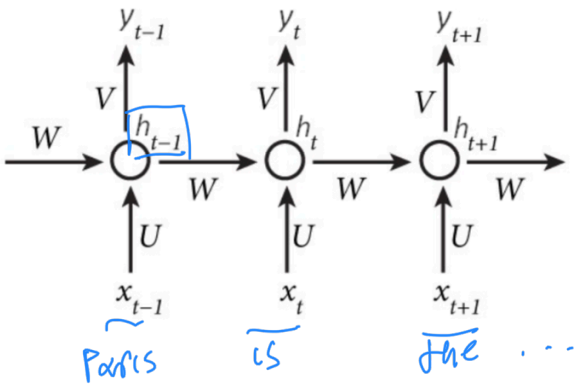
- Sequence learning: study of machine learning algorithms designed for sequential data (time series, language…)
  - Translate: "*machine learning is a challenging topic*" to French: "*L'apprentissage automatique est un sujet difficile*"
  - Predict the next word: "*John visited Paris, the capital of ….*"
  - Input and output strongly correlated within the sequence.

Recurrent Neural Networks

- Update the hidden state in a deterministic nonlinear way.
- RNNs are powerful,
- Distributed hidden state that allows them to store a lot of information about the past efficiently.
- Non-linear dynamics that allows them to update their hidden state in complicated ways.
- No need to infer hidden state, pure deterministic.
- Weight sharing



Recurrent Neural Networks

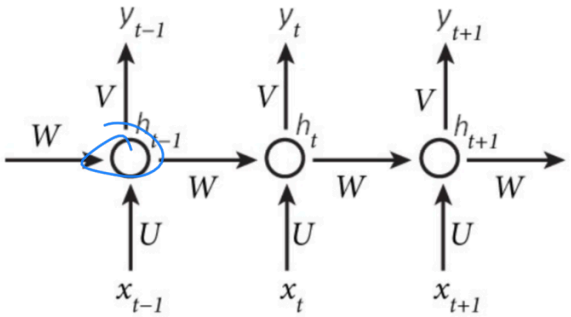


$y_t = V h_t$

$h_t = f(x_t \cdot U + h_{t-1} \cdot W)$

- input: ordered list of input vectors $x_1, \ldots, x_T$ initial hidden state h_0 initialized to all zeros,
- output
  - ordered list of hidden states $h_1, \ldots, h_T$,
  - ordered list of output vectors $y_1, \ldots, y_T$.
  - The output vectors may serve as input for other RNN units, when considering deep architectures .
  - The hidden states correspond to the “short-term” memory of the network.
- The last hidden state represents the encoding (embedding) of the time series

Recurrent Neural Networks



$$h_t = f(Ux_t + Wh_{t-1} + b)$$

- f a nonlinear function
- $x_t \in R^{d_{in}}, U \in R^{Hxd_{in}}, W \in R^{HxH}$,

Parameter matrices

- d_{in} size of the vocabulary
- H: dimension of the hidden layer (H~100)
- $y_t \in R^{d_{out}}$ transforms the current hidden state ht to depend on the final task:
  - i.e. for classification $y_t = softmax(Vh_t)$
- $V \in R^{d_{out}\chi H}$ parameter matrix shared across all steps (i.e. for word level language model $dout = |V|$

Deep RNNs

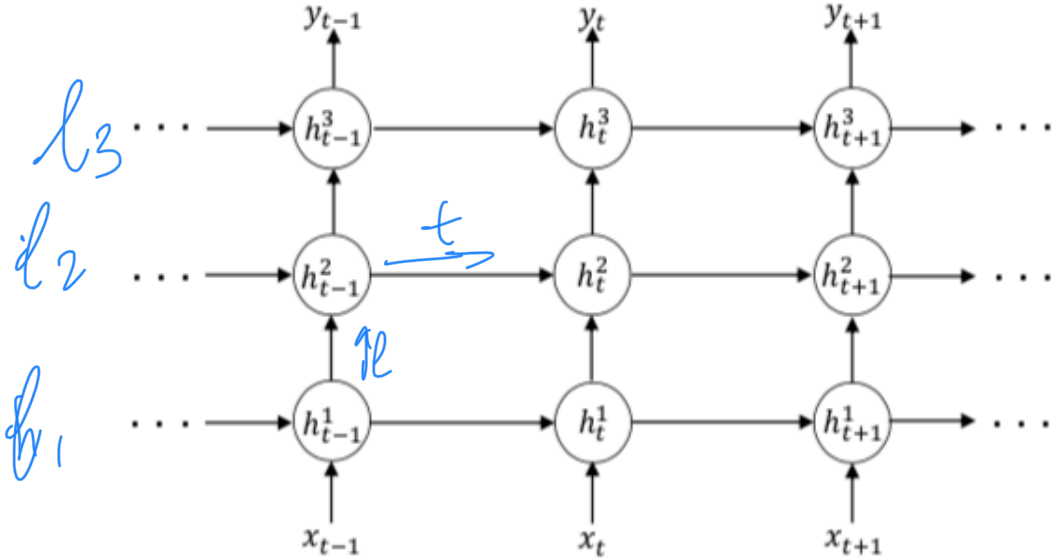
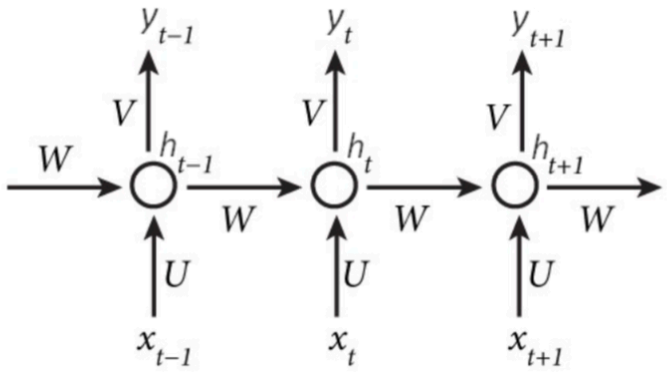


Figure 7: 3 steps of an unrolled deep RNN. Each circle represents a RNN unit. The hidden state of each unit in the inner layers (1 & 2) serves as input to the corresponding unit in the layer above.

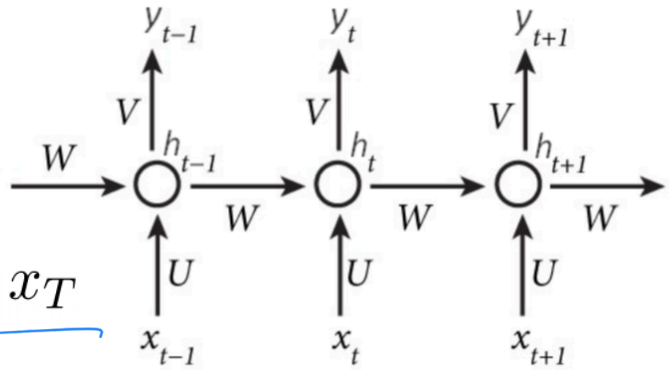
Recurrent Neural Networks



3 steps of an unrolled RNN

- CNNs are naturally efficient with grids[8],
- RNNs were specifically developed to be used with sequences
  - time series, or, in NLP, words (sequences of letters) or sentences (sequences of words).
  - language modeling $P[w_n | w_1, ..., w_{n-1}]$.
  - RNNs trained with such objectives can be used to generate new and quite convincing sentences from scratch
  - RNN can be considered as a chain of simple neural layers that share the same parameters.

Learning in RNNs



- Assuming input $x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T$
- As a single time:

$$h_t = \sigma(Wh_{t-1} + Ux_t)$$

$$y_t = softmax(Vh_t)$$

Learning in RNNs

$$h_t = \sigma(W h_{t-1} + U x_t)$$
$$y_t = softmax(V h_t)$$

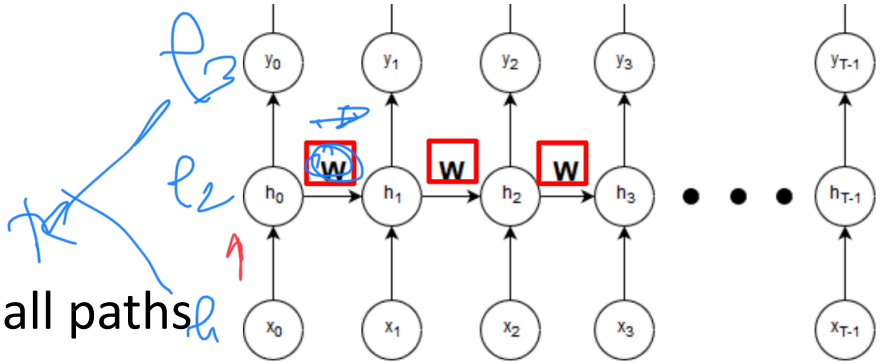
- Main idea: we use the same set of W, U, V weights at all time steps!
- $h_0 \in R^{HxH}$ initialization vector for the hidden layer at time step 0
- $\hat{y} \in R^{|V|}$ is a probability distribution over the vocabulary
- Loss function (@ time t): cross entropy predicting words instead of classes

$$J^{(t)}(\theta) = -\sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$$

Learning in RNNs – backpropagation in time

- Learning the weights of W:

$$\mathbf{W} \leftarrow \mathbf{W} - \alpha \frac{\partial \mathbf{y}}{\partial \mathbf{W}}$$



- Computation involves summation over all paths regarding
- i. time

$$\frac{\partial \mathbf{y}}{\partial \mathbf{W}} = \sum_{j=0}^{T-1} \frac{\partial \mathbf{y}_j}{\partial \mathbf{W}}$$

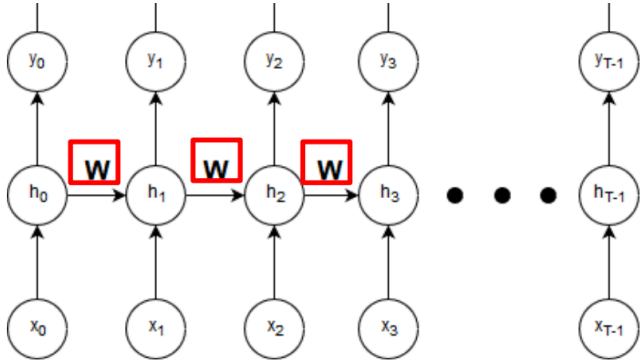
- ii. Levels of hidden layers

$$\frac{\partial y_j}{\partial \mathbf{W}} = \sum_{k=1}^{j} \frac{\partial y_j}{\partial h_k} \frac{\partial h_k}{\partial \mathbf{W}}$$

Learning in RNNs – backpropagation in time

- Therefore:

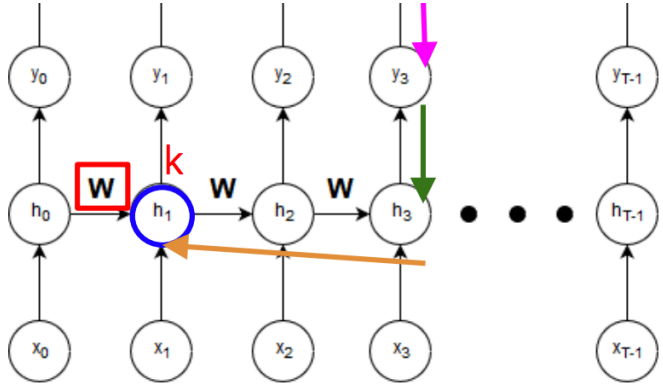
$$\frac{\partial y_j}{\partial \mathbf{W}} = \sum_{k=1}^{j} \frac{\partial y_j}{\partial h_j} \frac{\partial h_j}{\partial h_k} \frac{\partial h_k}{\partial \mathbf{W}}$$



- Indirect dependency. One final use of the chain rule:

$$\frac{\partial h_j}{\partial h_k} = \prod_{m=k+1}^{j} \frac{\partial h_m}{\partial h_{m-1}}$$

Learning in RNNs – backpropagation in time



$$\frac{\partial y_j}{\partial \mathbf{W}} = \sum_{j=0}^{T-1} \sum_{k=1}^{j} \frac{\partial y_j}{\partial h_j} \left(\prod_{m=k+1}^{j} \frac{\partial h_m}{\partial h_{m-1}} \right) \frac{\partial h_k}{\partial \mathbf{W}}$$

Learning in RNNs – vanishing/exploding gradients

$$\frac{\partial y_j}{\partial \mathbf{W}} = \sum_{j=0}^{T-1} \sum_{k=1}^{j} \frac{\partial y_j}{\partial h_j} \left(\prod_{m=k+1}^{j} \frac{\partial h_m}{\partial h_{m-1}} \right) \frac{\partial h_k}{\partial \mathbf{W}}$$

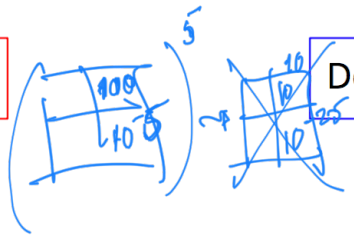
$$h_m = f(\mathbf{W}_h h_{m-1} + \mathbf{W}_x x_m)$$

$$\frac{\partial h_m}{\partial h_{m-1}} = \mathbf{W}_h^T \operatorname{diag}\left(f'(\mathbf{W}_h h_{m-1} + \mathbf{W}_x x_m)\right)$$

$$\frac{\partial h_j}{\partial h_k} = \prod_{m=k+1}^{j} \mathbf{W}_h^T \operatorname{diag}\left(f'(\mathbf{W}_h h_{m-1} + \mathbf{W}_x x_m)\right)$$

Weight Matrix

Derivative of activation function



- Repeated matrix multiplications leads to vanishing and exploding gradients.

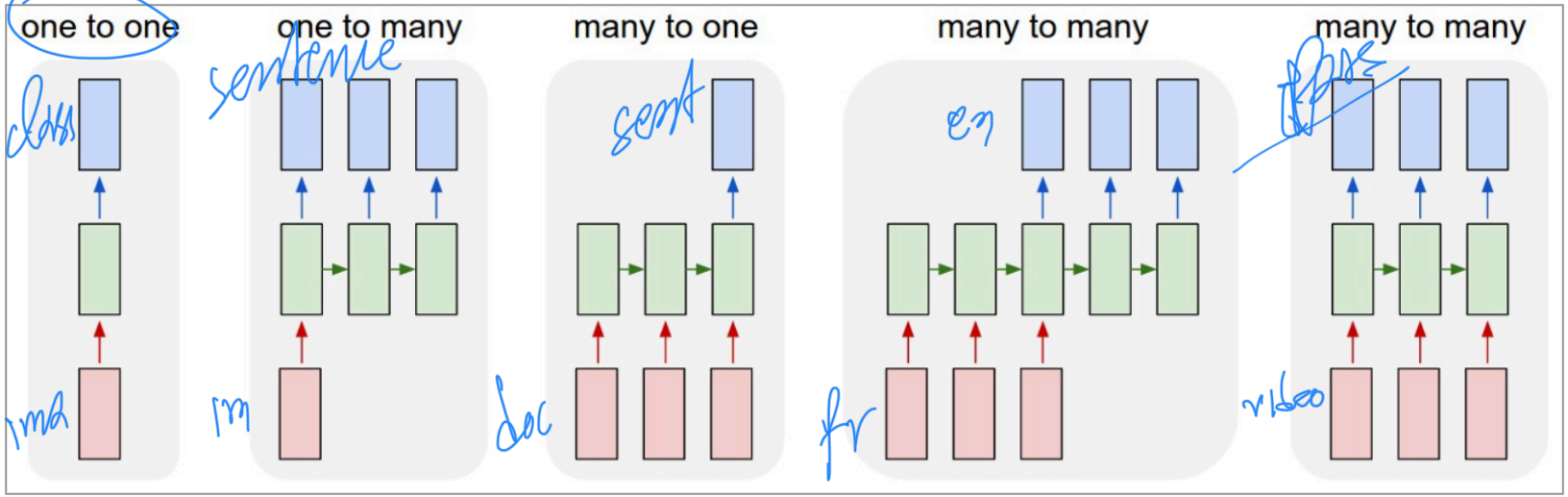
Learning in RNNs – vanishing/exploding gradients

- Initialization of W with 1s
- Using relu as activation function $f(z) = rect(z) = max(z, 0)$
- Using - clip gradients to a maximum value [Mikolov]

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

RNN Applications



fixed-sized input to fixed-sized output (e.g. image classification).

Sequence output (e.g. image captioning takes an image and outputs a sentence of words).

Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).

Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French).

Synced sequence input and output (e.g. video classification where we wish to label each frame of the video)..

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

LSTM Unit

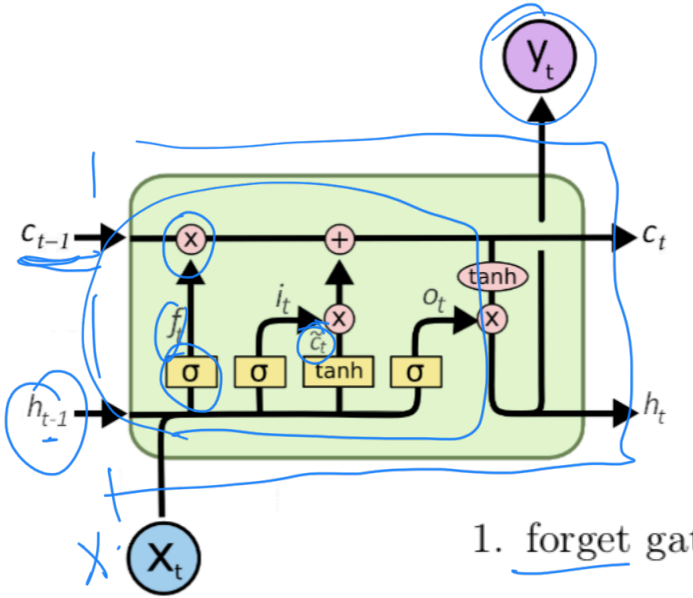
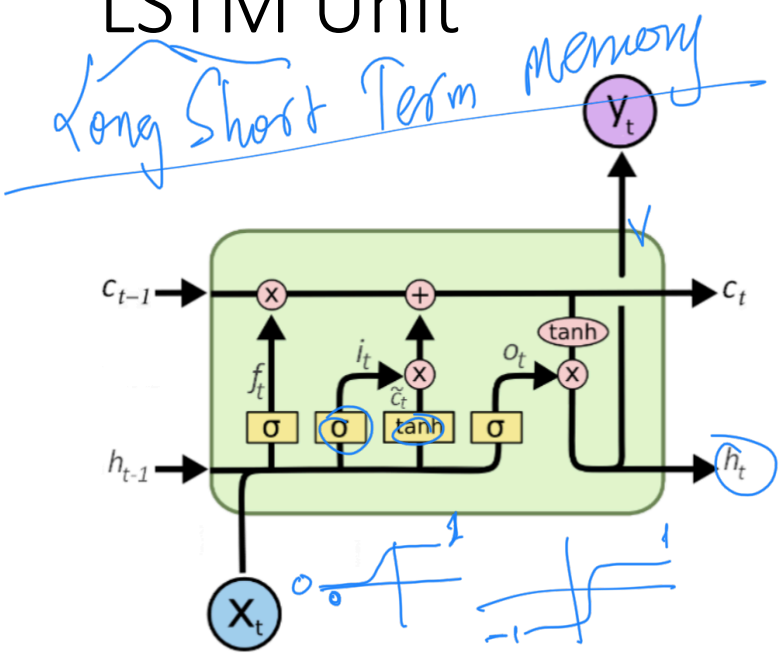


Figure 8: The LSTM unit. Adapted fro

- To tackle RNN problems of i. vanishing gradients and ii. Keep track of information for longer term.
- There is a "memory bus" Ct on which we chose how much to write and read

1. forget gate layer: $f_t = \sigma\left(U_f x_t + W_f h_{t-1} + b_f\right)$
2. input gate layer: $i_t = \sigma\left(U_i x_t + W_i h_{t-1} + b_i\right)$
3. candidate values computation layer: $\tilde{c}_t = \tanh\left(U_c x_t + W_c h_{t-1} + b_c\right)$
4. output gate layer: $o_t = \sigma\left(U_o x_t + W_o h_{t-1} + b_o\right)$

LSTM Unit



1. forget gate layer: $f_t = \sigma\left(U_f x_t + W_f h_{t-1} + b_f\right)$
2. input gate layer: $i_t = \sigma\left(U_i x_t + W_i h_{t-1} + b_i\right)$
3. candidate values computation layer: $\tilde{c}_t = \tanh\left(U_c x_t + W_c h_{t-1} + b_c\right)$
4. output gate layer: $o_t = \sigma\left(U_o x_t + W_o h_{t-1} + b_o\right)$

Assume a new training example x_t and the current hidden state h_{t-1},

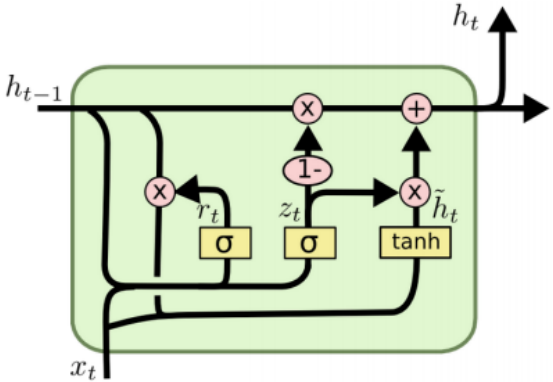
- forget gate layer f_t determines how much of the previous cell state c_{t-1} should be forgotten (what fraction of the memory should be freed up),
- input gate layer decides how much of the candidate values $\tilde{c_t}$ should be written to the memory: how much of the new information should be learned. Combining the output of the two filters updates the cell state:

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

$$h_t = \tanh\left(c_t\right) \circ o_t$$

$$y_t = \mathrm{softmax}\left(V h_t\right)$$

Gated Recurrent Unit (GRU)



- Simplified LSTM Unit
  - Reset gate $r_t = \sigma(U_r x_t + W_r h_{t-1} + b_r)$
  - Update gate $z_t = \sigma(U_z x_t + W_z h_{t-1} + b_z)$

- Candidate hidden state: $\hat{h}_t = \tanh(U_h x_t + W_h(r_t \circ h_{t-1}) + b_h)$

- update gate serves as the forget gate in LSTM
- Determines fraction of information from previous hidden state to be forgotten
- input gate is coupled on the forget gate

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \hat{h}_t$$

References (1/2)

1. Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.
2. Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. Vol. 2. IEEE, 2006.
3. Luong, Minh-Tang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).
4. Mueller, J., & Thyagarajan, A. (2016, February). Siamese Recurrent Architectures for Learning Sentence Similarity. In *AAAI* (pp. 2786-2792).
5. Course slides "Recurrent Neural Networks" Richard Socher, Stanford, 2016
6. Recurrent Neural Network Architectures Abhishek Narwekar, Anusri Pampari, University of Illinois, 2016
7. Neculoiu, Paul, Maarten Versteegh, and Mihai Rotaru. "Learning text similarity with siamese recurrent networks." *Proceedings of the 1st Workshop on Representation Learning for NLP*. 2016.
8. Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." arXiv preprint arXiv:1509.00685 (2015).
9. Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.
10. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning (pp. 2048-2057).
11. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. NAACL 2016 (pp. 1480-1489).

OUTLINE

- RNNs, LSTMs
- **Attention/Context based architectures**
  - **ELMO**, BERT. BART

Context & Senses

- **“a word is defined by “the company it keeps” (Firth, 1957)**
- Words are *ambiguous*:
  - The amount of deposits in the banks decreased by 3.5% in March
  - The trees on bank of the river were offering their shade to the visitors
- We need different vectors to represent all word/token senses

Deep Contextualized Word Representations

Best paper NAACL 2018

transfer learning has been used in vision since 2012 (ImageNet)
it is used in NLP since 2018! [1] [2]

ELMo: Embeddings from Language Models

Initial problem: traditional word vectors map each word to a single, context-independent vector

But some words have more than one sense! (polysemy)

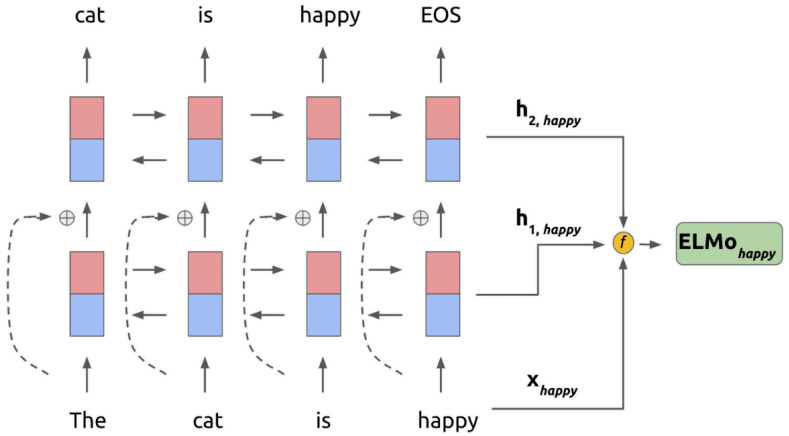
e.g., bank, get, wood, play, mean…

Solution:

- token assigned a representation based on entire input sentence
- use the internal representations of a RNN language model pre-trained on a large dataset

Deep Contextualized Word Representations

- bidirectional LSTM is trained with a coupled language model (LM) on a large text corpus - ELMo (Embeddings from Language Models) representations.

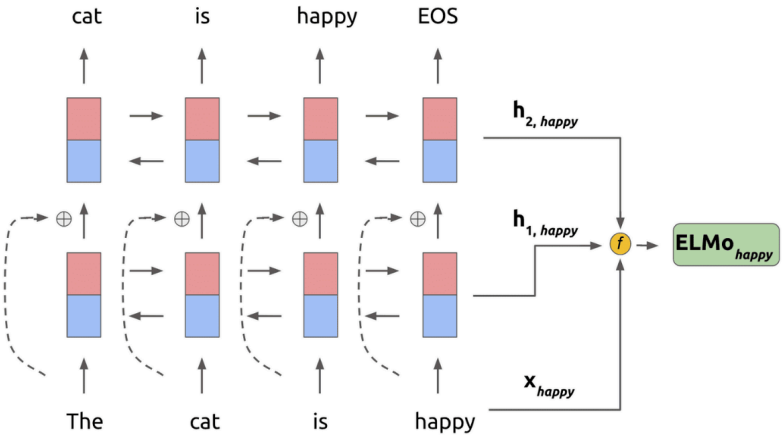


  - Forward $p(t_1, t_2 ..., t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2 ..., t_N)$
  - Bacward $p(t_1, t_2 ..., t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2} ..., t_{k+N})$

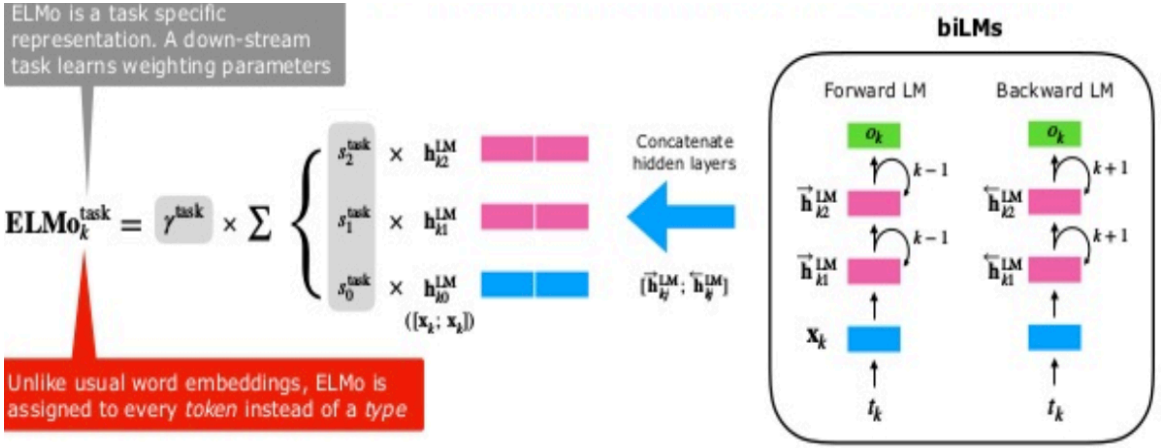
Graphic from: https://www.mihaileric.com/posts/deep-contextualized-word-representations-elmo/

Deep Contextualized Word Representations

- ELMo representations are deep: function of all of the internal layers of the biLM.
  - learn a linear combination of the vectors stacked above each input word for each end task,
  - Combining the internal states in this manner allows for very rich word representations.
  - higher-level LSTM states ʼcontext-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised word sense disambiguation tasks)
  - lower level states model aspects of syntax (e.g., they can be used to do part-of-speech tagging).
  - Simultaneously exposing all of these signals is highly beneficial, allowing the learned models select the types of semi-supervision that are most useful for each end task.



Deep Contextualized Word Representations



Semi-supervised approach:

- 1) a deep bidirectional RNN language model is pretrained on a large dataset
- 2) the vector of each word in a given input sentence is computed as a weighted sum of the RNN hidden states
- 1) is *unsupervised*, 2) weights are learned in a *supervised* way on some task-specific dataset

Graphic from: https://www.mihaileric.com/posts/deep-contextualized-word-representations-elmo/

$$\mathbf{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

$h_{k,j}^{LM}$ is the k[th] hidden representation of the j[th] layer of the bi-RNN LM

s^{task} is the softmax weight vector, γ^{task} is a scaling parameter (for optimization)

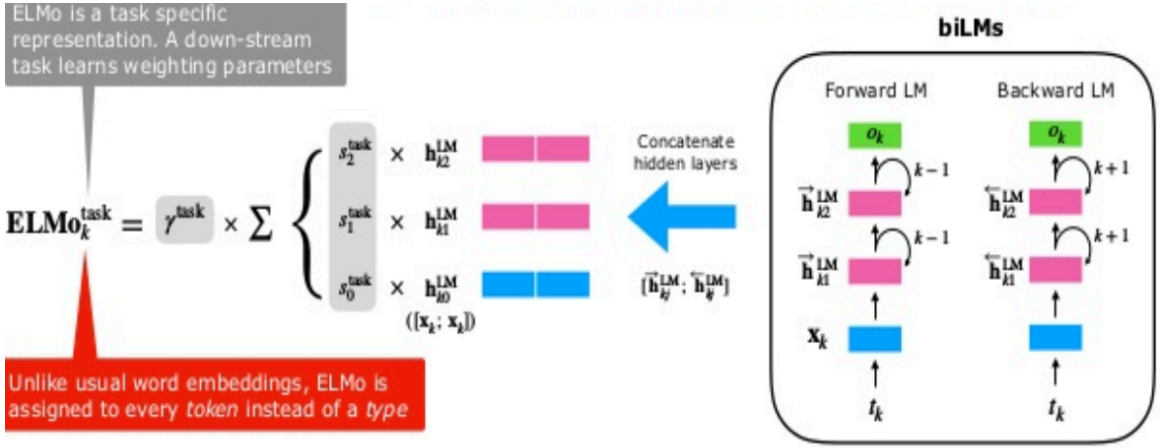
The authors use L=2 in all experiments

Deep Contextualized Word Representations

Use of ELMo in practice, on a task-specific dataset:

- 1) use the pretrained language model in prediction mode and store $h_{k,j}^{LM}$ for each word (each k) and each layer (each j)
- 2) concatenate $ELMo_k^{task}$ with the corresponding input vector* of whatever supervised model is used to solve the task (e.g., RNN, CNN, feed-forward…)
- 3) update s^{task} and γ^{task} with the other parameters of the supervised model during training

*given by word2vec, glove, etc.



Results:

ELMo improves over the baselines on 8 tasks, ranging from question answering to co-reference resolution, sentiment analysis, POS-tagging, and disambiguation

Deep Contextualized Word Representations

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

SQuAD: Question answering, SNLI: Entailment, SRL: Semantic Role Labelling ("Who did what to whom"), NER: Name entity recognition, SST: Sentiment Analysis, Coref: task of clustering mentions in text that refer to the same underlying real world entities

Deep Contextualized Word Representations

Model	F_1
WordNet 1st Sense Baseline	65.9
Raganato et al. (2017a)	69.9
Iacobacci et al. (2016)	**70.1**
CoVe, First Layer	59.4
CoVe, Second Layer	*64.7*
biLM, First layer	67.4
biLM, Second layer	*69.0*

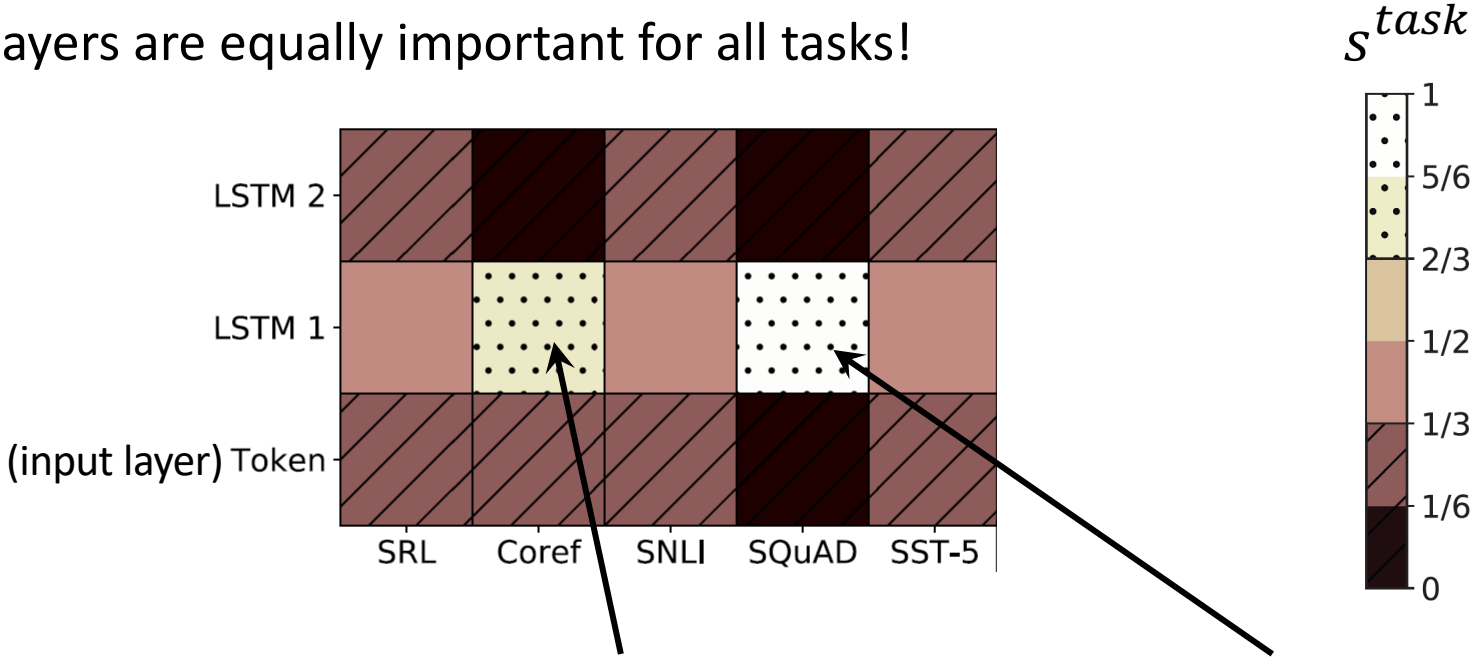
WSD

Model	Acc.
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	**97.8**
CoVe, First Layer	*93.3*
CoVe, Second Layer	92.8
biLM, First Layer	*97.3*
biLM, Second Layer	96.8

POS tagging

Deep Contextualized Word Representations

Not all layers are equally important for all tasks!



The 1st layer clearly dominates for co-reference resolution and question answering

For the other tasks, the weights are more evenly distributed among layers

OUTLINE

- RNNs, LSTMs
- **Attention/Context based architectures**
  - ELMO, **BERT,** BART
- Energy Based architectures

ATTENTION BASED ARCHITECTURES

Attention is ubiquitous in DL today

Image captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

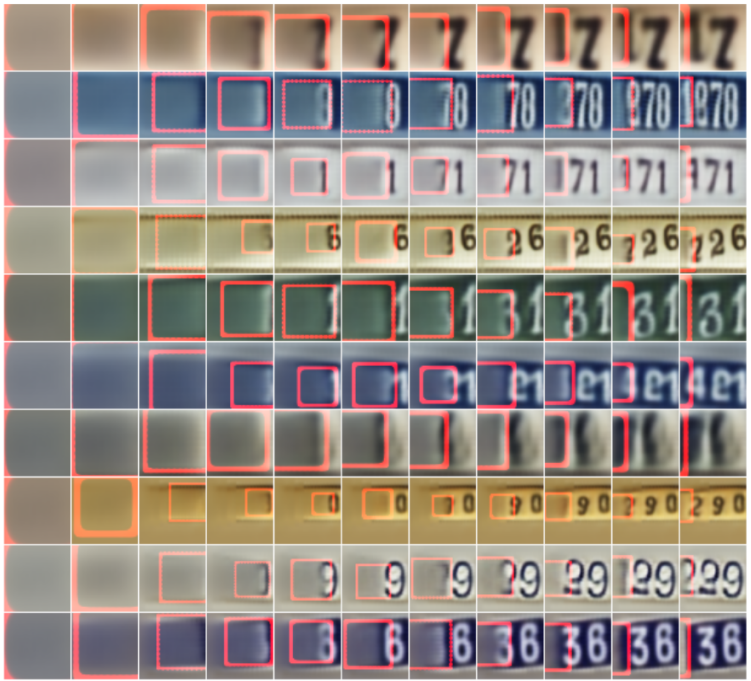
A group of people sitting on a boat in the water.

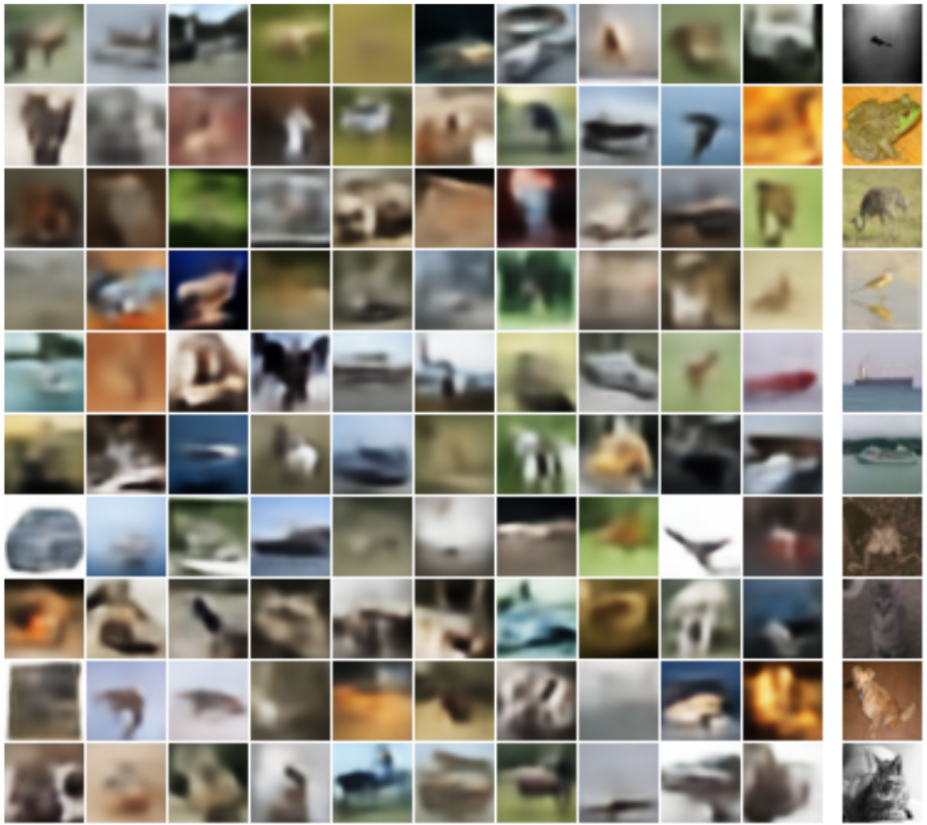
A giraffe standing in a forest with trees in the background.

Show, attend and tell: Neural image caption generation with visual attention (Xu et al. 2015)

Attention is ubiquitous in DL today

Image generation

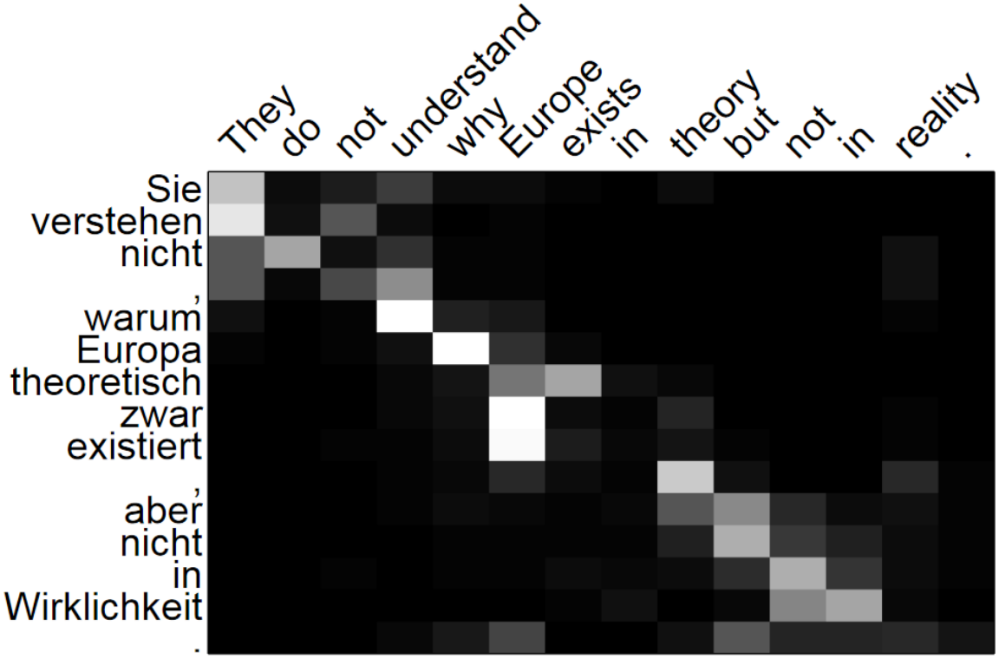




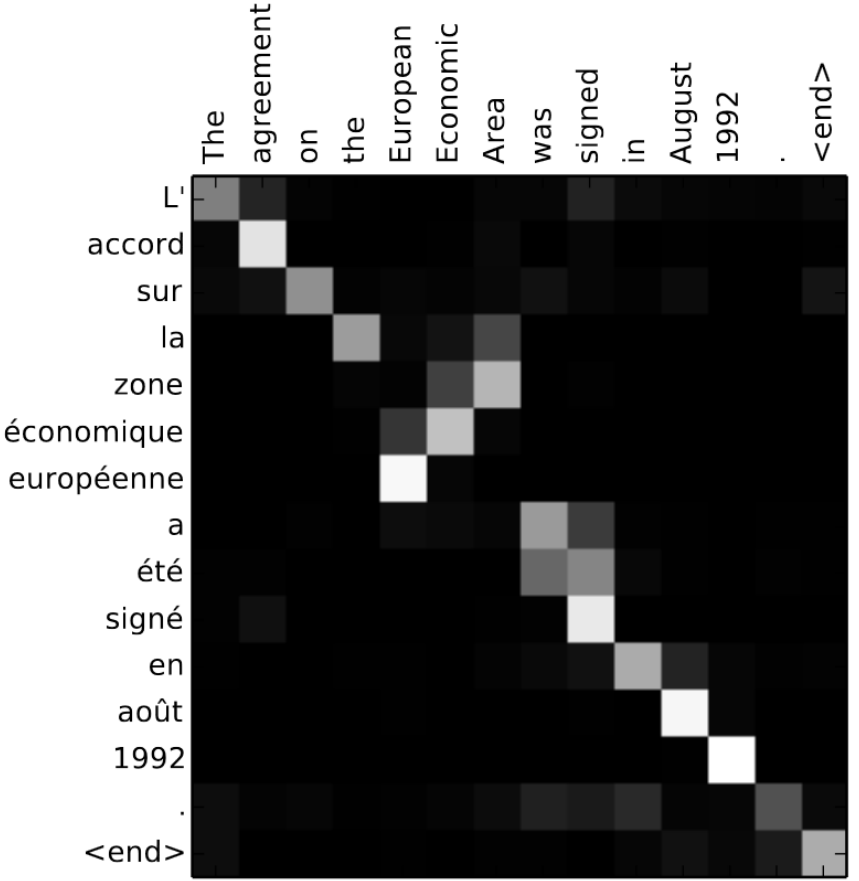
DRAW: A recurrent neural network for image generation (Gregor et al. 2015)

Attention is ubiquitous in DL today

Neural Machine Translation



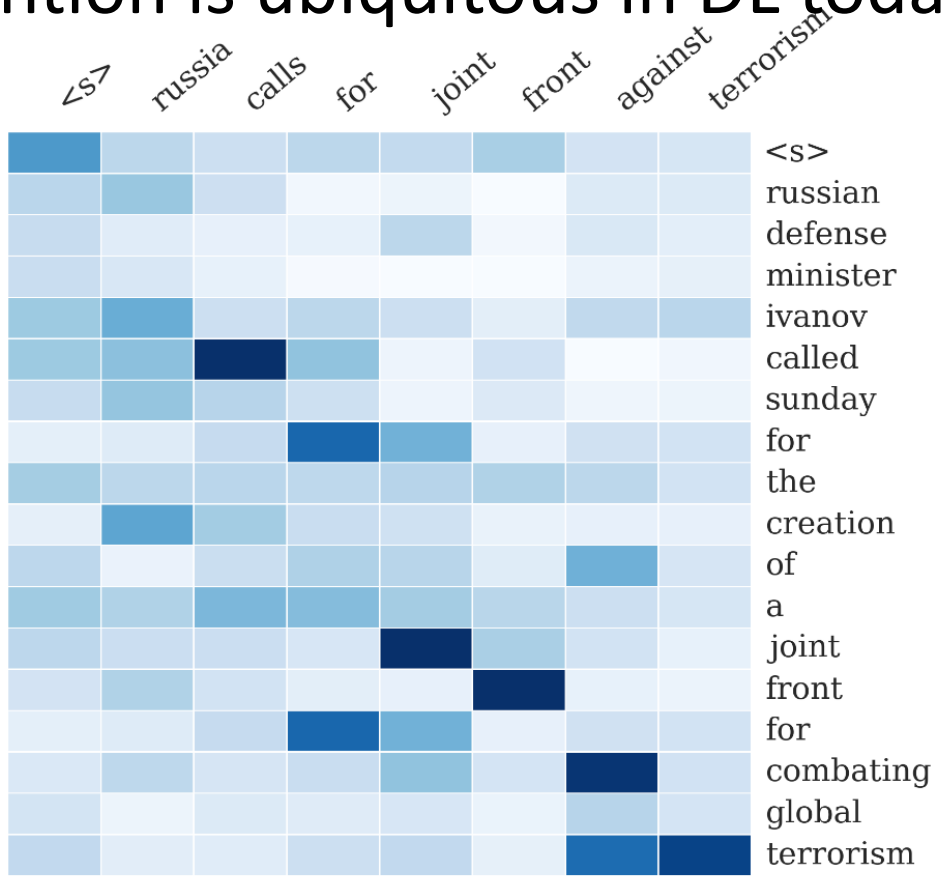
Effective approaches to attention-based neural machine translation (Luong et al. 2015)



Neural machine translation by jointly learning to align and translate (Bahdanau et al. 2014)

Attention is ubiquitous in DL today

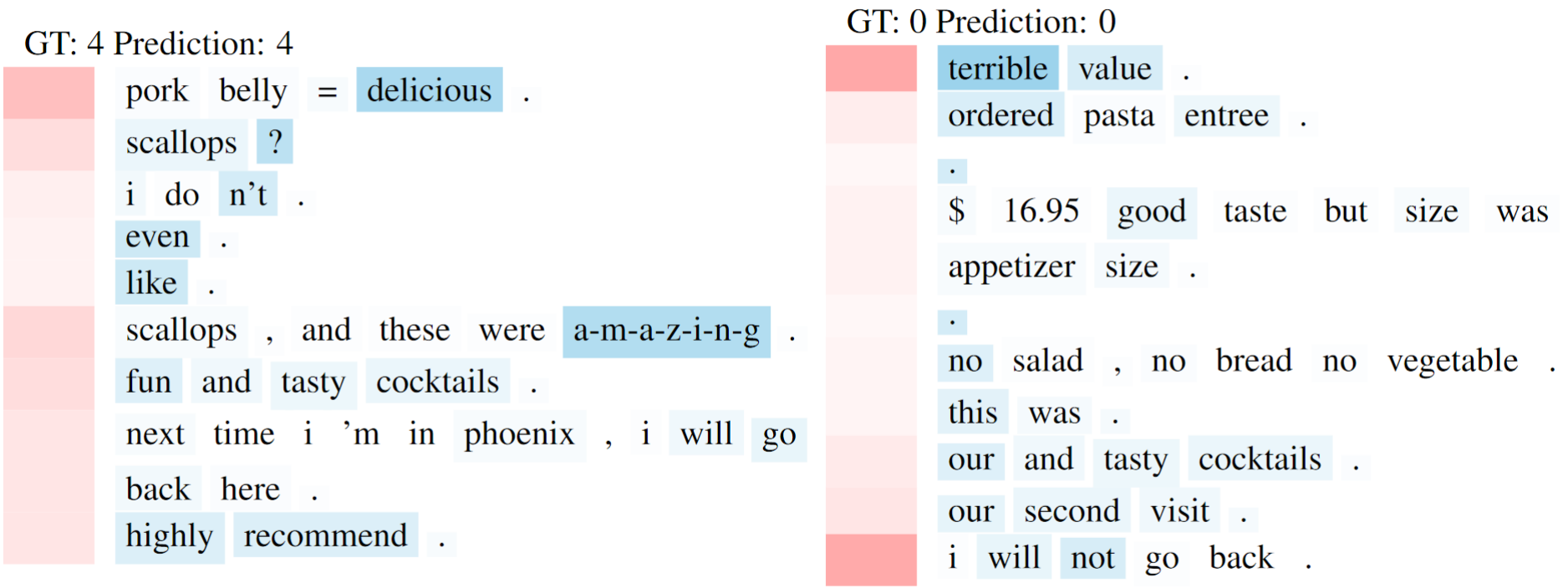
Abstractive summarization



A neural attention model for abstractive sentence summarization (Rush et al. 2015)

Attention is ubiquitous in DL today

Sentiment analysis



Hierarchical attention networks for document classification (Yang et al. 2016)

Encoder-Decoder Architectures

General idea:

encoder **decoder**

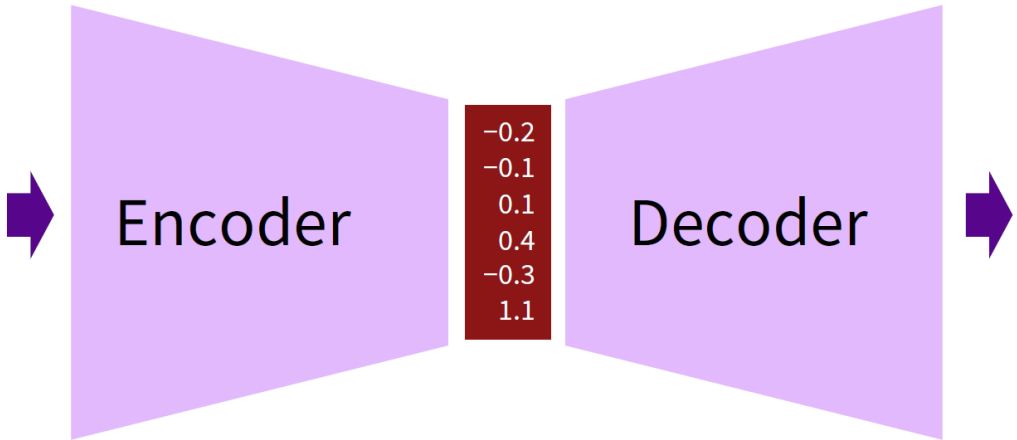
Input -> embedding -> Output

- image
- document
- source language
- …

- caption
- summary
- target language
- …

Known as *sequence-to-sequence* (seq2seq) when input and output are sequences (e.g., NLP applications)

The full architecture is differentiable => end-to-end training



What is attention?

Objective

- Traditional models (e.g., encoder) having to embed the input into a single fixed-length vector (lossy).
- Alternatively information can be kept and stored into multiple vectors.
- information can be retrieved later on (e.g., by the decoder). (Bahdanau et al. 2014)

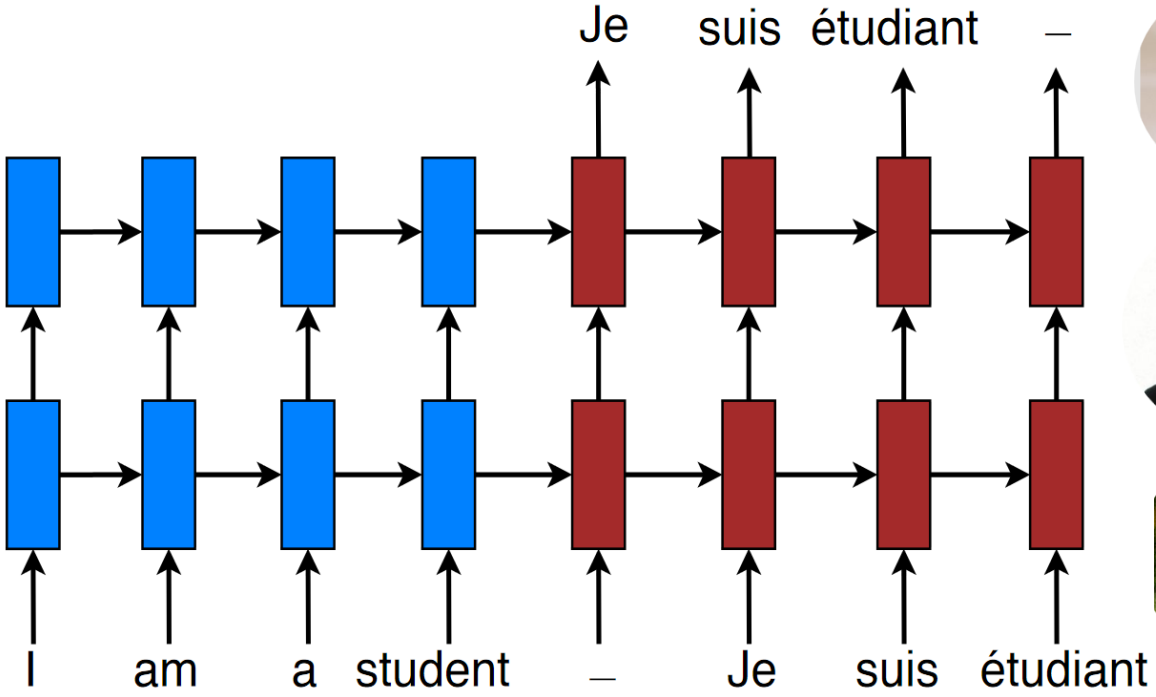
Quick history:

- developed in the context of **encoder-decoder** architectures for neural machine translation (Bahdanau et al. 2014)
- rapidly applied to naturally related tasks like image captioning (Xu et al. 2015) and summarization (*Luong et al. 2015*)
- also proposed for encoders only, e.g. for text classification (Yang et al. 2015) and representation learning (Conneau et al. 2017).
  Known as *self or inner attention* in such cases.

Encoder-Decoder for Neural Machine Translation

Encoder **Decoder**

***Target** sentence (generated)*



***Source** sentence (input)*

Minh-Thang Luong

Research Scientist at Google
Verified email at google.com - Homepage
Deep Learning Natural Language Processing

Hieu Pham

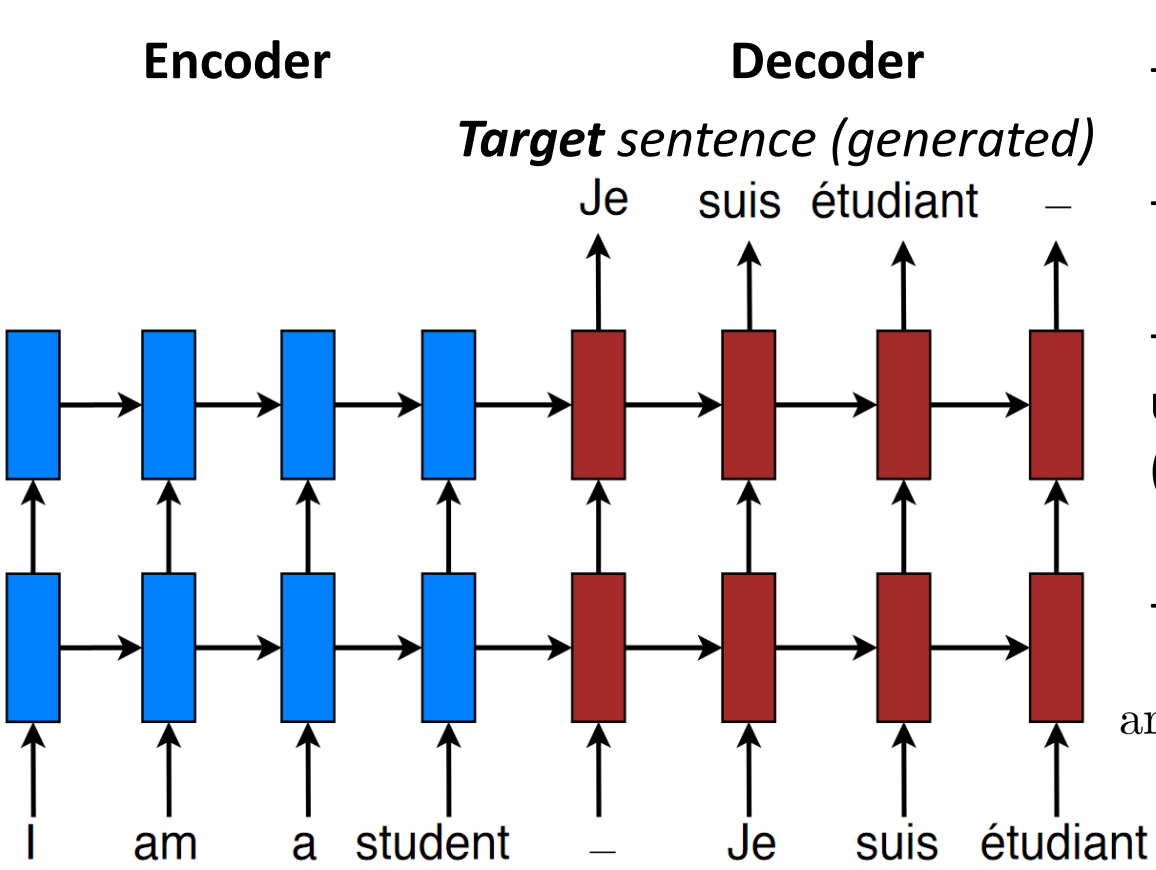
Carnegie Mellon University, Google Brain
Verified email at google.com
Machine Learning

Christopher D Manning

Professor of Computer Science and Linguistics
Verified email at stanford.edu - Homepage
Natural Language Processing

Effective approaches to Attention-Based Neural Machine Translation (2015)

Encoder-Decoder for Neural Machine Translation



- source: $\left(x_1, \ldots, x_{T_x}\right)$

- target: $\left(y_1, \ldots, y_{T_y}\right)$

- Both encoder & decoder are unidirectional deep RNNs (a.k.a. *stacking* RNNs)

- Training objective:

$$\text{argmax}_\theta \left\{ \sum_{(x,y)\in \text{corpus}} \log\, p(y|x;\theta) \right\}$$

Luong et al. (2015)

Encoder-Decoder for Neural Machine Translation

Encoder: usually: CNN, stacking RNN* with LSTM or GRU units…
* unidirectional (Luong et al. 2015) or
bidirectional (Bahdanau et al. 2014).

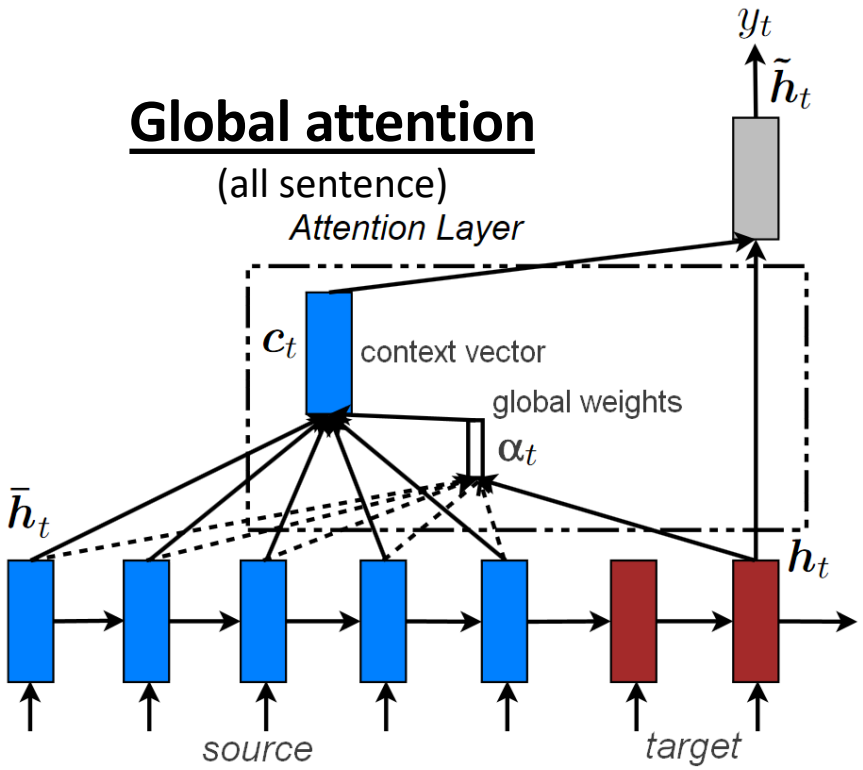
Decoder: unidirectional *RNN* (well suited to text generation), best if deep.

Generates the target sentence $\left(y_1, \cdots, y_{T_y}\right)$ one word at a time:

$$P\left[y_t | \{y_1, \ldots, y_{t-1}\}, c_t\right] = \text{softmax}\left(W_s \tilde{h}_t\right)$$

Luong et al. (2015)

Encoder-**Decoder** for Neural Machine Translation



$$P\left[y_t|\{y_1,...,y_{t-1}\},c_t\right] = \text{softmax}\left(W_s\tilde{h}_t\right)$$

attentional hidden state

$$\tilde{h}_t = \tanh\left(W_c\left[c_t;h_t\right]\right)$$

context vector

decoder hidden state

$$c_t = \sum_{i=1}^{T_x} \alpha_{t,i}\bar{h}_i$$

i^{th} encoder hidden state

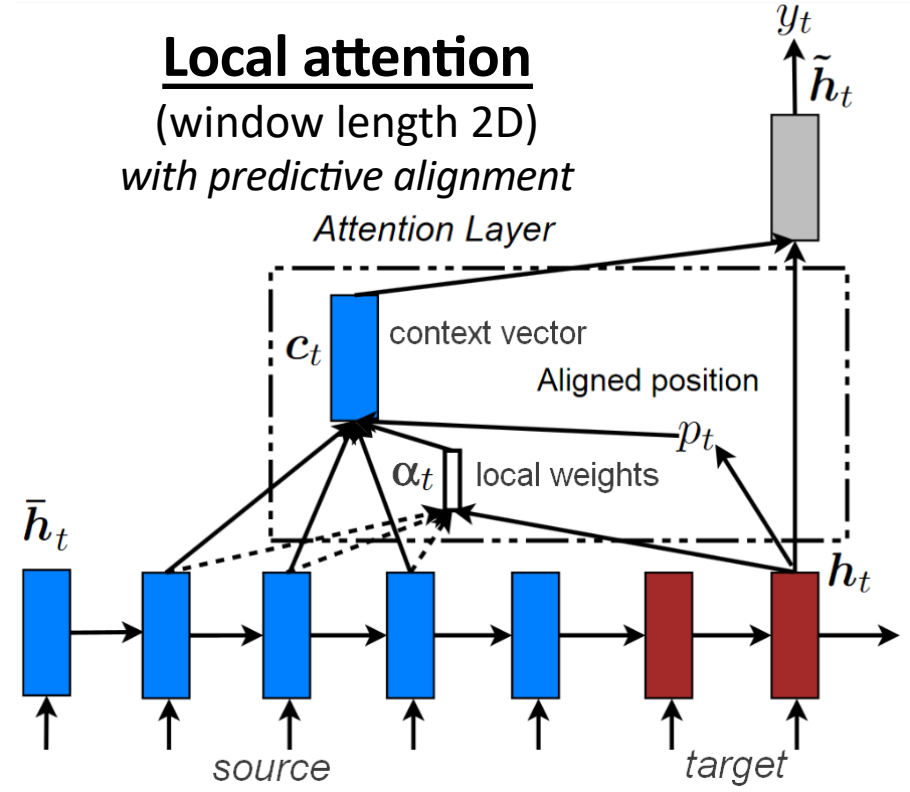
alignment vector

$$\alpha_{t,i} = \frac{\exp\left(\text{score}\left(h_t,\bar{h}_i\right)\right)}{\sum_{i'=1}^{T_x}\exp\left(\text{score}\left(h_t,\bar{h}_{i'}\right)\right)}$$

$$\text{score}\left(h_t,\bar{h}_i\right) = h_t^\top \bar{h}_i$$

Luong et al. (2015)

Encoder-**Decoder** for Neural Machine Translation



$$P\left[y_t|\{y_1,...,y_{t-1}\},c_t\right] = \text{softmax}\left(W_s\tilde{h}_t\right)$$

attentional hidden state

$$\tilde{h}_t = \tanh\left(W_c\left[c_t; h_t\right]\right)$$

decoder hidden state

context vector

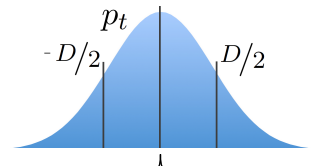
$$p_t = T_x \cdot \sigma\left(v_p^\top \tanh\left(W_p h_t\right)\right)$$

$$c_t = \sum_{i=p_t-D}^{p_t+D} \alpha_{t,i}\bar{h}_i$$

i[th] encoder hidden state

alignment vector

$$\alpha_{t,i} = \frac{\exp\left(\text{score}(h_t,\bar{h}_i)\right)}{\sum_{i'=p_t-D}^{p_t+D}\exp\left(\text{score}(h_t,\bar{h}_{i'})\right)}\exp\left(-\frac{(i-p_t)^2}{2(D/2)^2}\right)$$



$$\text{score}(h_t,\bar{h}_i) = h_t^\top W_\alpha \bar{h}_i$$

Luong et al. (2015)

Encoder-Decoder for Neural Machine Translation

Facts:

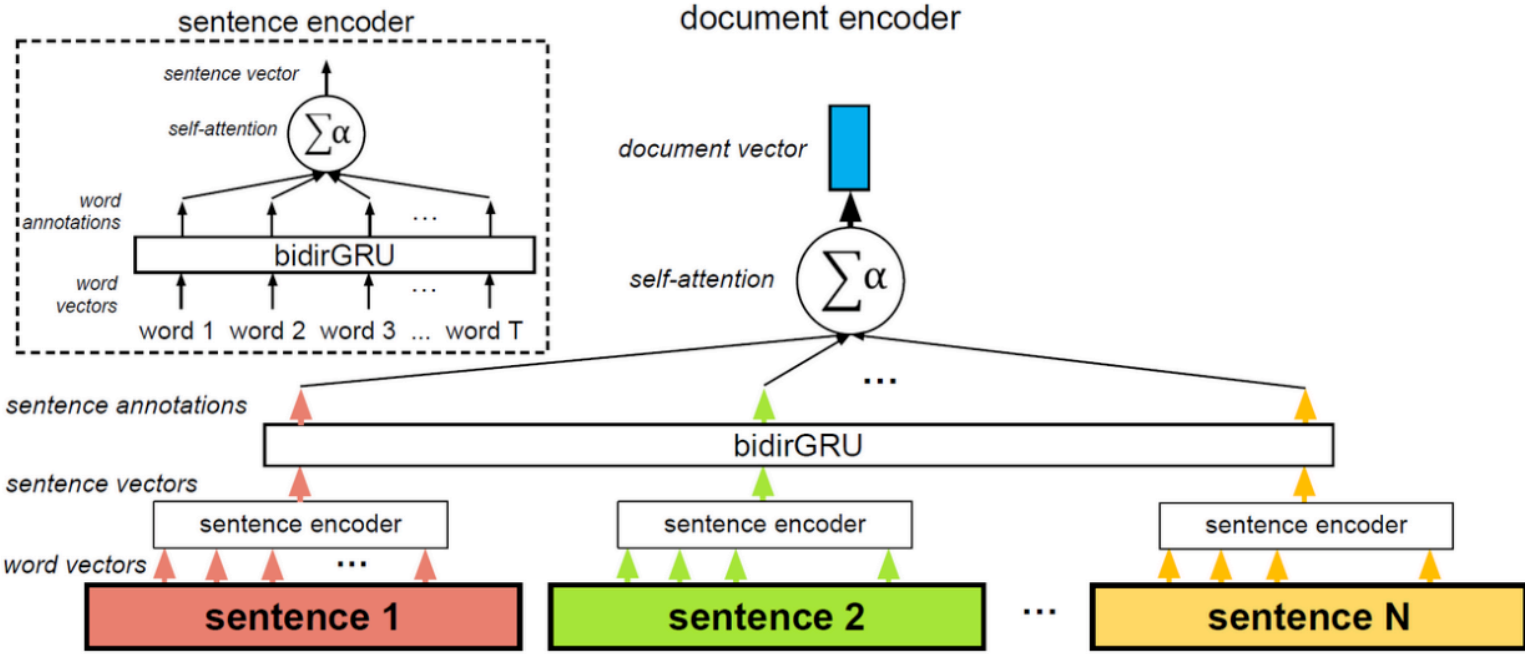
- Tested on the English <-> German task (WMT'14 dataset)
- 4.5M sentence pairs
- Encoder and decoder RNNs feature 4 layers of stacking and 1000-dimensional hidden states
- Window of size *D=10* for local attention
- Trained for 12 epochs (total of 7-10 days on a single GPU, at 1K target words/s)
- New state-of-the-art performance

Lessons learned:

- Local attention with predictive alignment gives better results than global attention
- Dot product ($\text{score}(h_t, \bar{h}_i) = h_t^\top \bar{h}_i$) works well for global attention
- The general formulation ($\text{score}(h_t, \bar{h}_i) = h_t^\top W_\alpha \bar{h}_i$) is better for local attention

Luong et al. (2015)

Self-attention for RNN encoders



Self-attention for RNN encoders

- Input is a sentence $(x_1, \ldots, x_T)$
- We're only interested in getting an embedding s of the sentence for some downstream task (e.g., classification)

encoder hidden state

context vector

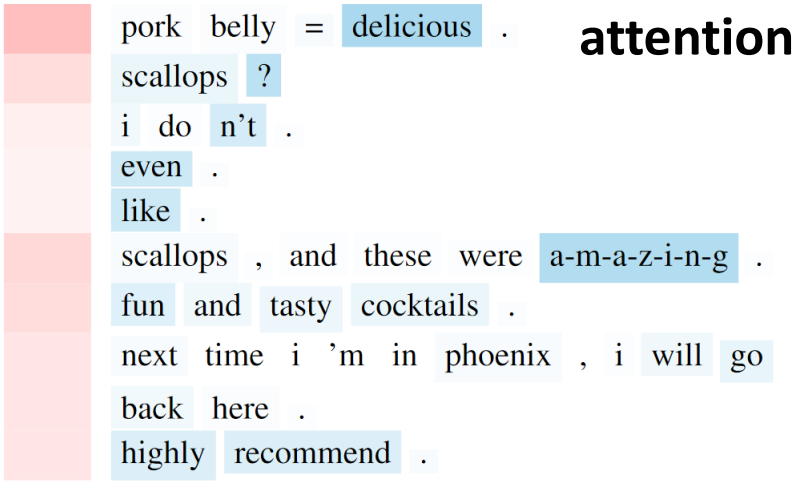
$$u_t = \tanh(W h_t)$$

$$\alpha_t = \frac{\exp(\text{score}(u_t, u))}{\sum_{t'=1}^{T} \exp(\text{score}(u_{t'}, u))}$$

$$s = \sum_{t=1}^{T} \alpha_t h_t$$

Where $\text{score}(u_t, u) = u_t^\top u$

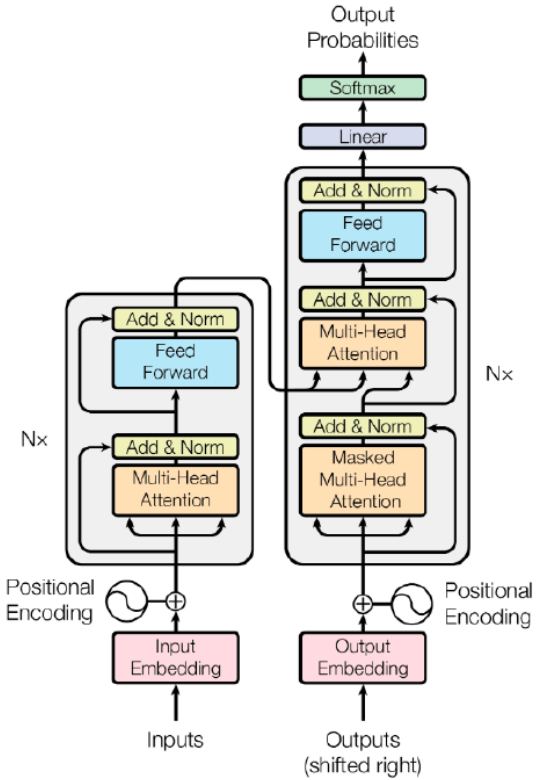
The same process can be repeated over the sentence vectors s -> **hierarchical attention**



Attention is All You Need - Tranformer

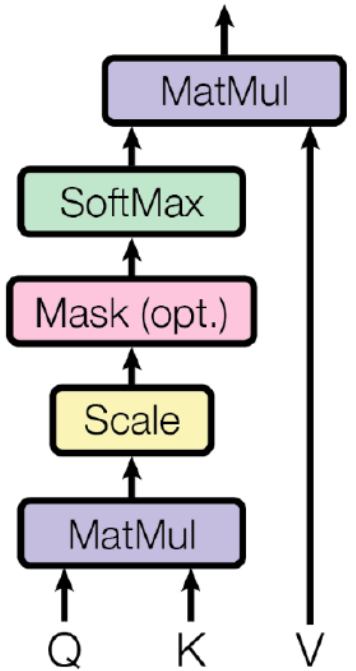
Transformer

- A model that follows the encoder-decoder structure, where the input tokens are mapped to a sequence of continuous representations, and these representations are consumed by the decoder which generates a sequence of outputs.
- Does not use recurrent neural networks or convolutional neural networks. Instead, it only uses dense and attention layers.



Transformer – Scaled dot product attention

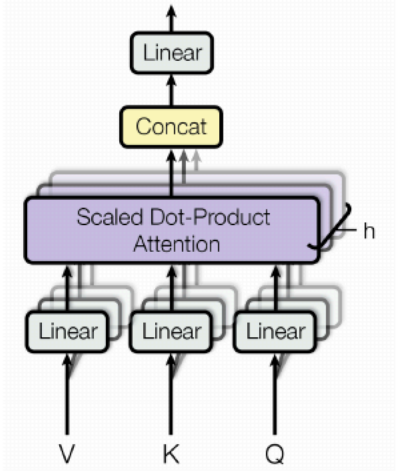
- A query Q_i representing the token at position i attend to a sequence of tokens (represented by K): $Q_i K^T$.
- The result scaled by $\sqrt{d_k}$ (dimension of Q_i) is passed to a softmax function to compute a score for each of the tokens: $softmax(\frac{Q_i K^T}{\sqrt{d_k}})$
- A weighted sum of the Value vectors is calculated as the new representation of the token at position i: $softmax(\frac{Q_i K^T}{\sqrt{d_k}})V$
- Queries are stacked in one matrix Q, the output of the attention layer becomes: $softmax(\frac{QK^T}{\sqrt{d_k}})V$



Transformer – Multihead Attention

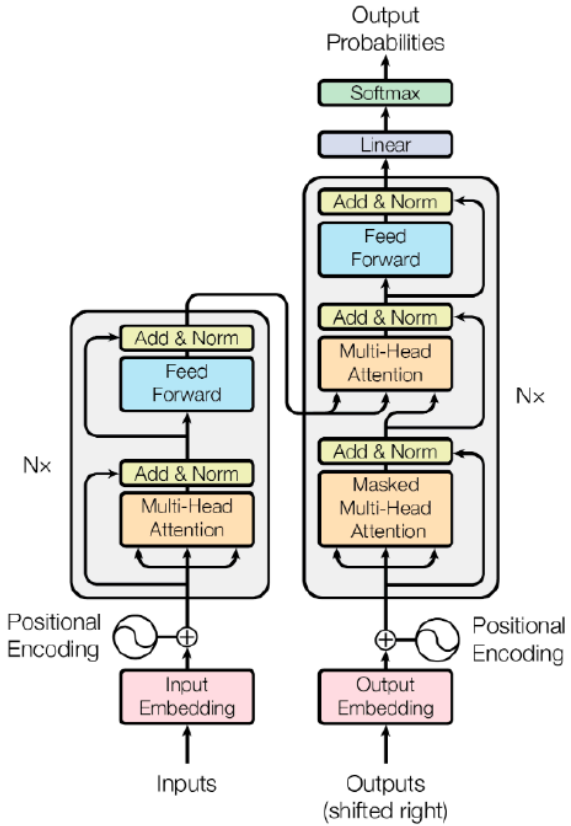
- Q, K and V are linearly projected h times.
- Scaled dot-product attention is applied h times on the projections to produce h heads:
  $head_j = Attention(QW_j^Q, KW_j^K, VW_j^V)$
- heads are concatenated and linearly projected to produce the new representation:

$$MultiHead(Q, K, V) = Concat(head_i, ..., head_h)W$$



Transformers- Attention

- **Self-attention:** Applied in the encoder and in the decoder:
  - In the encoder: Q, K and V come from the previous encoder sub-layer.
  - In the decoder: Q, K and V come from the previous decoder layer and a mask is applied to prevent the decoder from attending future positions.
- **Encoder-decoder attention:** Applied only in the decoder. Q come from the previous decoder layer. K and V come from the output representations of the encoder.



Transformer - experiments

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost

Pretrained Language Models

- Pre-trained on unsupervised large corpora.
- Fine-tuned on supervised tasks, with an important gain in performance, especially in the case where the number of training examples is limited.

GPT

- Pre-trained auto-regressive Transformer decoder.
- Pre-training objective: maximize: $\sum_i \log P(u_i | u_{i-k}, ..., u_{i-1}; \theta)$

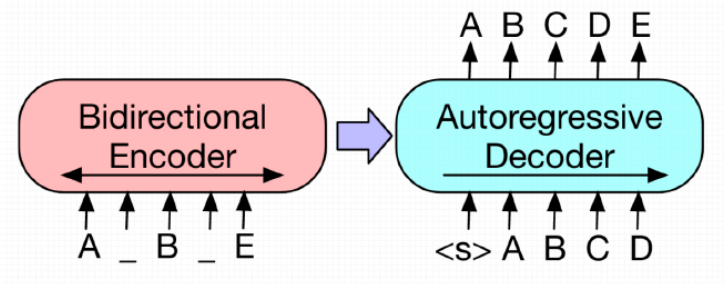
BERT

- Pre-trained stack of Transformer encoders.
- Two unsupervised pre-training tasks:
  - Masked language model.
  - Next sentence prediction.

BertSum – Bert for Summarization

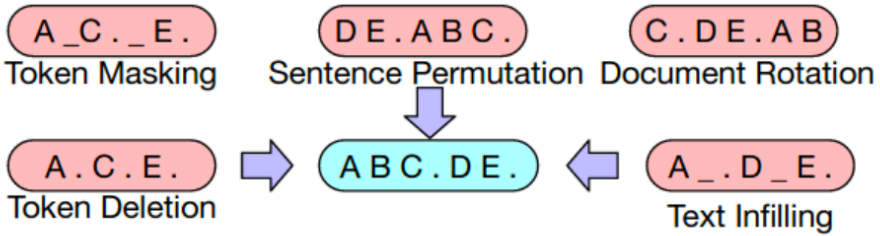
- Typically, the abstractive summarization task can be seen as a sequence-to-sequence problem:
  - A sequence of input tokens $[x_1, x_2, ..., x_n]$ (the source document) is encoded into a sequence of representations $[z_1, z_2, ..., z_n]$.
  - An auto-regressive decoder, consume these representations and produce a sequence $[y1, y_2, ..., y_m]$ (the summary) token-by-token.
- BertSum uses Bert architecture and weights as encoder to produce document representations.
- The decoder is trained from scratch, contrary to the encoder.
- Two different optimizers with different learning rates are used to update the parameters of the encoder and the decoder.

Bart Overview



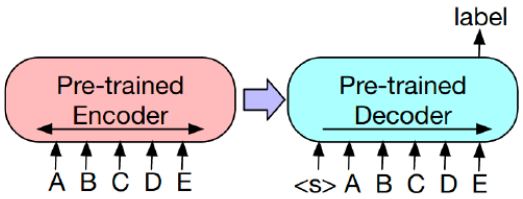
- A denoising autoencoder, that uses a bidirectional encoder and a left-to-right autoregressive decoder.
- Uses the standard Transformer architecture, with 12 encoder layers and 12 decoder layers.
- Pre-trained on 160GB of english text.
- Given its architecture, Bart can be used in both discriminative and generative tasks.

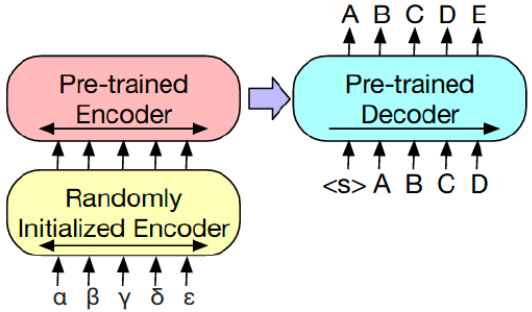
Bart – Pre-training Task



- The Pre-training task is to denoise a corrupted text. Several noising functions are tested:
  - Token Masking: random tokens are replaced with a [MASK] token.
  - Token Deletion: random tokens are deleted.
  - Text Infilling: Random text spans are sampled with span length drawn from Poisson distribution ($\lambda = 3$) and replaced with [MASK] token.
  - Sentence Permutation: Sentences are shuffled.
  - Sentence Rotation: sentences are rotated.
- Ablation experiments show that a combination of text infilling and sentence permutation perform consistently good on supervised tasks.

Bart – Fine tuning





- Bart can be fine-tuned on different types of tasks:
  - **Sequence Generation:** Given the decoder is autoregressive, Bart can be directly used to generate sequences in the same language as the input sequence.
  - **Sequence classification:** The final hidden state of the final decoder token is used. (Figure to the left)
  - **Tokens classification:** The top hidden state of the each decoder token is used.
  - **Machine translation:** The encoder embedding layer is replaced with an encoder that encodes the source language. (Figure to the right)

Bart – Evaluation

	SQuAD 1.1 EM/F1	SQuAD 2.0 EM/F1	MNLI m/mm	SST Acc	QQP Acc	QNLI Acc	STS-B Acc	RTE Acc	MRPC Acc	CoLA Mcc
BERT	84.1/90.9	79.0/81.8	86.6/-	93.2	91.3	92.3	90.0	70.4	88.0	60.6
UniLM	-/-	80.5/83.4	87.0/85.9	94.5	-	92.7	-	70.9	-	61.1
XLNet	**89.0**/94.5	86.1/88.8	89.8/-	95.6	91.8	93.9	91.8	83.8	89.2	63.6
RoBERTa	88.9/**94.6**	**86.5/89.4**	**90.2/90.2**	96.4	92.2	94.7	**92.4**	86.6	**90.9**	**68.0**
BART	88.8/**94.6**	86.1/89.2	89.9/90.1	**96.6**	**92.5**	**94.9**	91.2	**87.0**	90.4	62.8

	CNN/DailyMail			XSum		
	R1	R2	RL	R1	R2	RL
Lead-3	40.42	17.62	36.67	16.30	1.60	11.95
PTGEN (See et al., 2017)	36.44	15.66	33.42	29.70	9.21	23.24
PTGEN+COV (See et al., 2017)	39.53	17.28	36.38	28.10	8.02	21.72
UniLM	43.33	20.21	40.51	-	-	-
BERTSUMABS (Liu & Lapata, 2019)	41.72	19.39	38.76	38.76	16.33	31.15
BERTSUMEXTABS (Liu & Lapata, 2019)	42.13	19.60	39.18	38.81	16.50	31.27
BART	**44.16**	**21.28**	**40.90**	**45.14**	**22.27**	**37.25**

mBart – Multilingual Bart

- Multilingual sequence-to-sequence denoising auto-encoder.
- Follows BART architecture and objectives, but trained on several mono-language corpora across several languages.
- Can be directly fine-tuned on machine translation supervised tasks.
- Shows important performance gain in low-resource settings.
- mBART25 pre-trained model is published: trained on 1369.6 GB of raw text data from 25 different languages corpora.

THANK YOU

Acknowledgements

- Moussa Kamaledine – transformer part
- Tixier: Attention

References (1)

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. arXiv preprint arXiv:1705.02364.

Chopra, Sumit, Raia Hadsell, and Yann LeCun. "Learning a similarity metric discriminatively, with application to face verification." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 2005.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.

Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. Vol. 2. IEEE, 2006.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).

Mueller, J., & Thyagarajan, A. (2016, February). Siamese Recurrent Architectures for Learning Sentence Similarity. In *AAAI* (pp. 2786-2792).

References (2)

Neculoiu, Paul, Maarten Versteegh, and Mihai Rotaru. "Learning text similarity with siamese recurrent networks." *Proceedings of the 1st Workshop on Representation Learning for NLP*. 2016.

Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." arXiv preprint arXiv:1509.00685 (2015).

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning (pp. 2048-2057).

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. NAACL 2016 (pp. 1480-1489).

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., & Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, *1*(0).

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, *2*(1), 1-127.

References (3)

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1994). Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems* (pp. 737-744).

Neculoiu, P., Versteegh, M., & Rotaru, M. (2016). Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP* (pp. 148-157).

Hoffer, E., & Ailon, N. (2015, October). Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition* (pp. 84-92). Springer, Cham.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

Dor, L. E., Mass, Y., Halfon, A., Venezian, E., Shnayderman, I., Aharonov, R., & Slonim, N. (2018). Learning Thematic Similarity Metric from Article Sections Using Triplet Networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Vol. 2, pp. 49-54).

Bellet, A., Habrard, A., & Sebban, M. (2013). A survey on metric learning for feature vectors and structured data.

Chen, W., Chen, X., Zhang, J., & Huang, K. (2017, July). Beyond triplet loss: a deep quadruplet network for person re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 2, No. 8).

References (4)

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

Peters, Matthew E., et al. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).