Faculty for Informatics

Technical University of Munich

Natural Language Processing
IN2361

Prof. Dr. Georg Groh

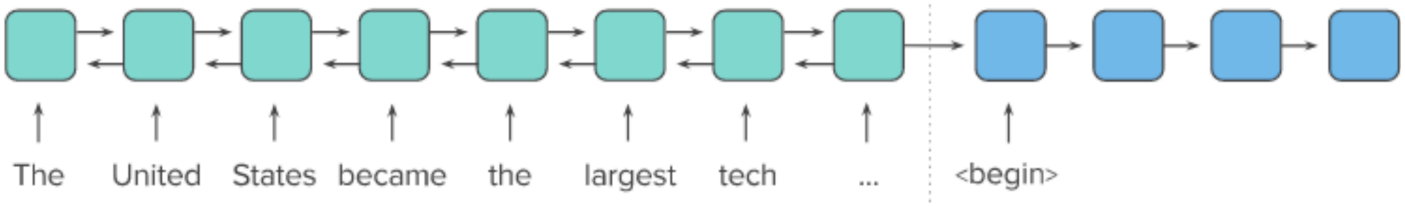
Social Computing
Research Group

Deep NLP
Part E: Transformer Networks and CNNs

- content is based on [2] (lecture 12)
- certain elements (e.g. figures, equations or tables) were taken over or taken over in a modified form from [2]
- citations of [2] are omitted for legibility
- errors on these slides are fully in the responsibility of Georg Groh
- BIG thanks to Richard Socher and his colleagues at Stanford for publishing materials [2] of a great Deep NLP lecture

Problems with RNNs = Motivation for Transformer Networks [3]

- Sequential computation prevents parallelization



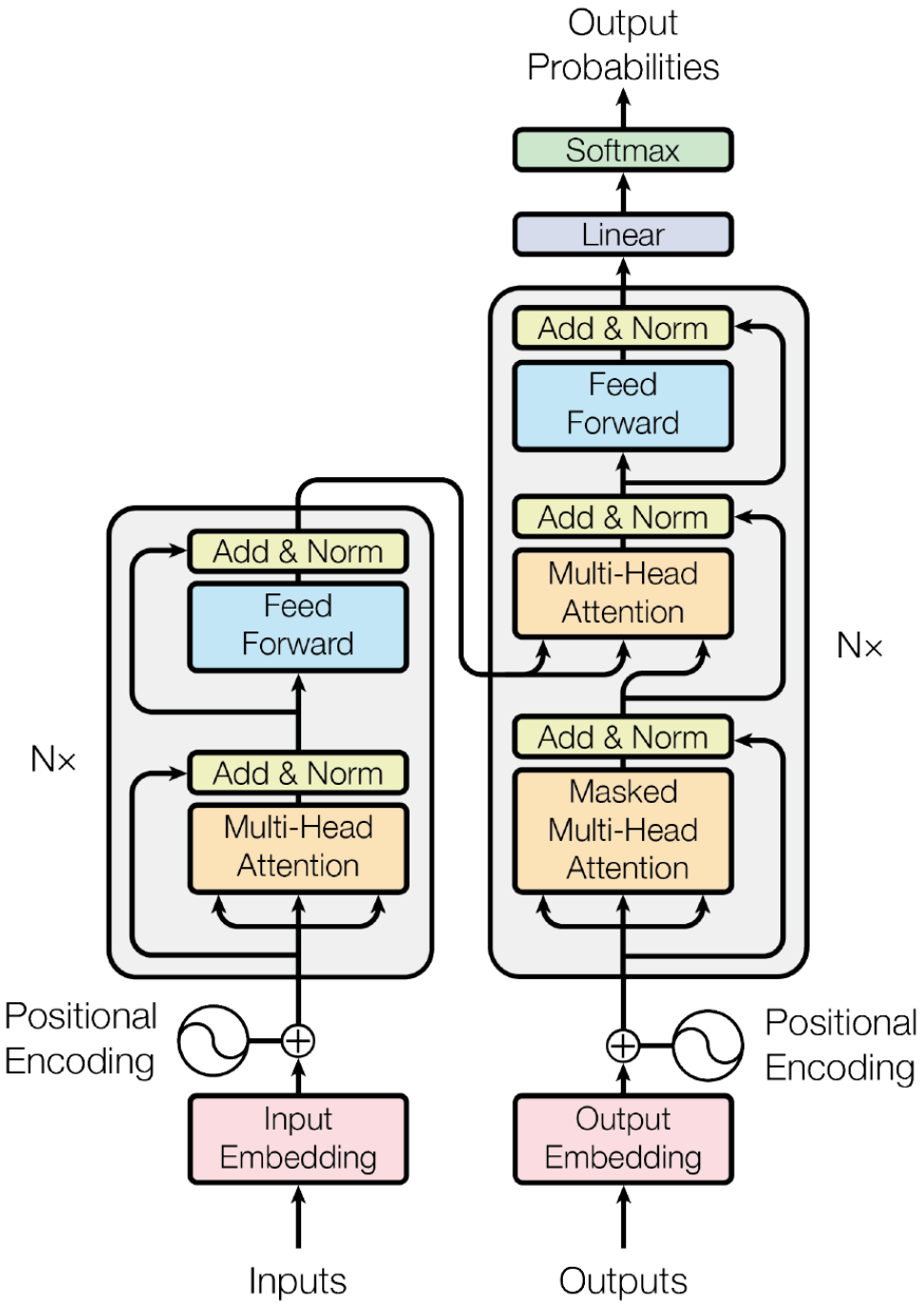
- Despite GRUs and LSTMs, RNNs still need attention mechanism to deal with long range dependencies → path length for co-dependent computation between states grows with sequence-length

- But if attention gives us access to any state… maybe we don't need the RNN at all? → idea in [3]: Transformer Networks

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need.

Transformer Networks [3]

- Sequence-to-sequence
- Encoder-Decoder
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier
- most vector dimensions $d_{model} = 512$;
- N = 6 layers



Attention Mechanism [3]

General Dot Product Attention in the notation of [3]:

- input: d_k dim. query q, d_k dim. keys k_i , d_v dim. values v_i
- output: weighted sum of values; weights: softmax on dot-product btw. keys and query
- →

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

- if multiple queries Q are processed → compact expression

$$A(Q, K, V) = softmax(QK^T)V$$

$[|Q| \times d_k] \times [d_k \times |K|] \times [|K| \times d_v]$

softmax row-wise

$= [|Q| \times d_v]$

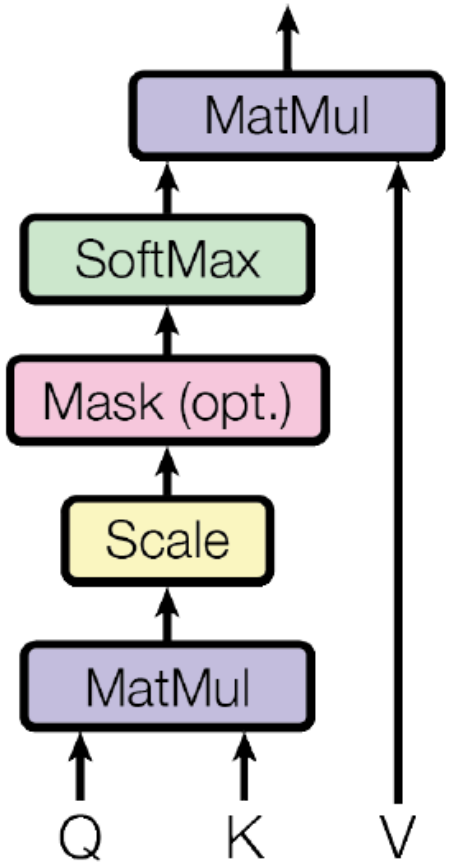
Attention Mechanism [3]

Scaled Dot Product Attention

- d_k gets large → variance of the $q \cdot k_i$ increases → some $q \cdot k_i$ may be very large → softmax is pushed into regions, where its gradient is very small

- → scale the softmax:

$$A(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$



[3]

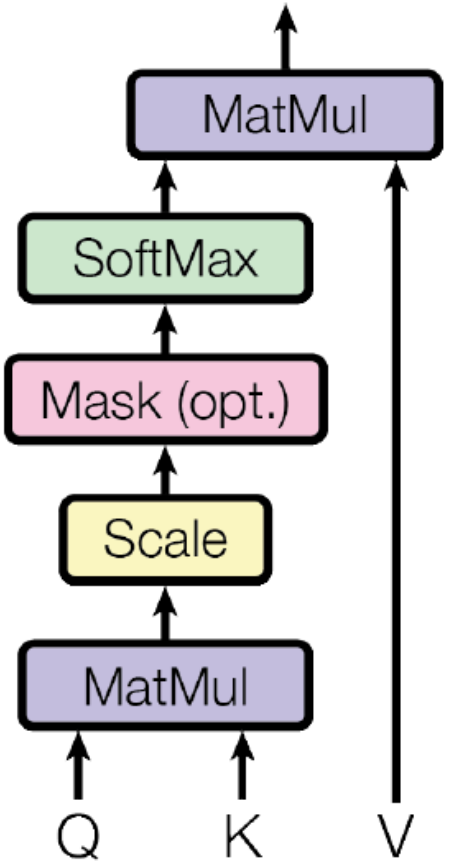
Attention Mechanism [3]

Self Attention

- e.g. encoder first layer: Q = K = V: d_{model} dim. input word embeddings → this is self attention: words attend to themselves

- e.g. decoder self attention layers: Q = K = V: each position in the decoder attends to all positions in the decoder up to and including that position (←→ masking)

- e.g. encoder-decoder attention layers:
  - queries: from previous decoder layer, keys and values: from output of encoder.
  - allows every position in decoder to attend over all positions in input sequence.
  - mimics typical encoder-decoder attention mechanisms in seq2seq models



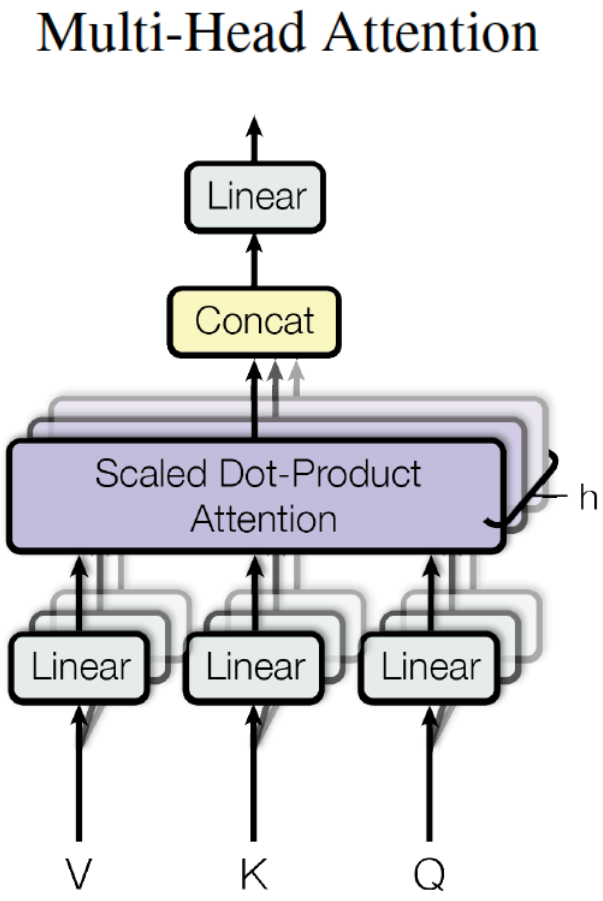
[3]

Attention Mechanism [3]

Multi-Head Attention

- e.g. encoder first layer: self attention: words attend to themselves
- problem: standard attention: only one way for words (vectors) to interact with one-another
- solution: Multi-Head Attention:
  - linearly map/project Q, K, V into h many lower dimensional spaces via W matrices
  - then apply attention, concatenate outputs, and pipe through linear layer:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



[3]

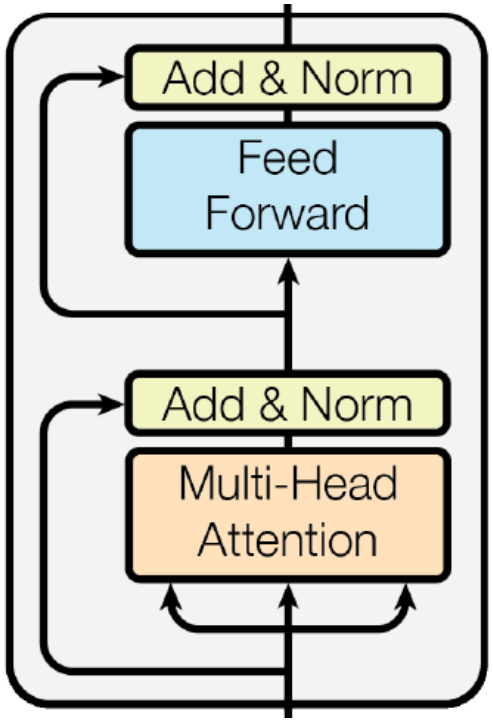
$W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}, W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}, W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$ $d_k = d_v = d_\text{model}/h = 64$ $h = 8$

Complete Block of Encoder [3]

- each block: two “sublayers”:
  - multihead attention
  - two-layer fully connected feed forward (C)NN with ReLu:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- each sublayer:
  - residual (short-circuit) connection and Layer Norm:
  - Layer Norm(x + Sublayer(x))
  - Layer Norm changes input to have mean 0 and variance 1 (see [4])



Encoder Input [3]

- d_{model}-dim. Byte-Pair word embeddings

- no recurrence steps → add d_{model}-dim. Positional Encodings to word embeddings to make use of order of sequence
  → same words at different locations have different overall representations
  - $$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
    $$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$
    (*i* iterates over embedding dimensions)

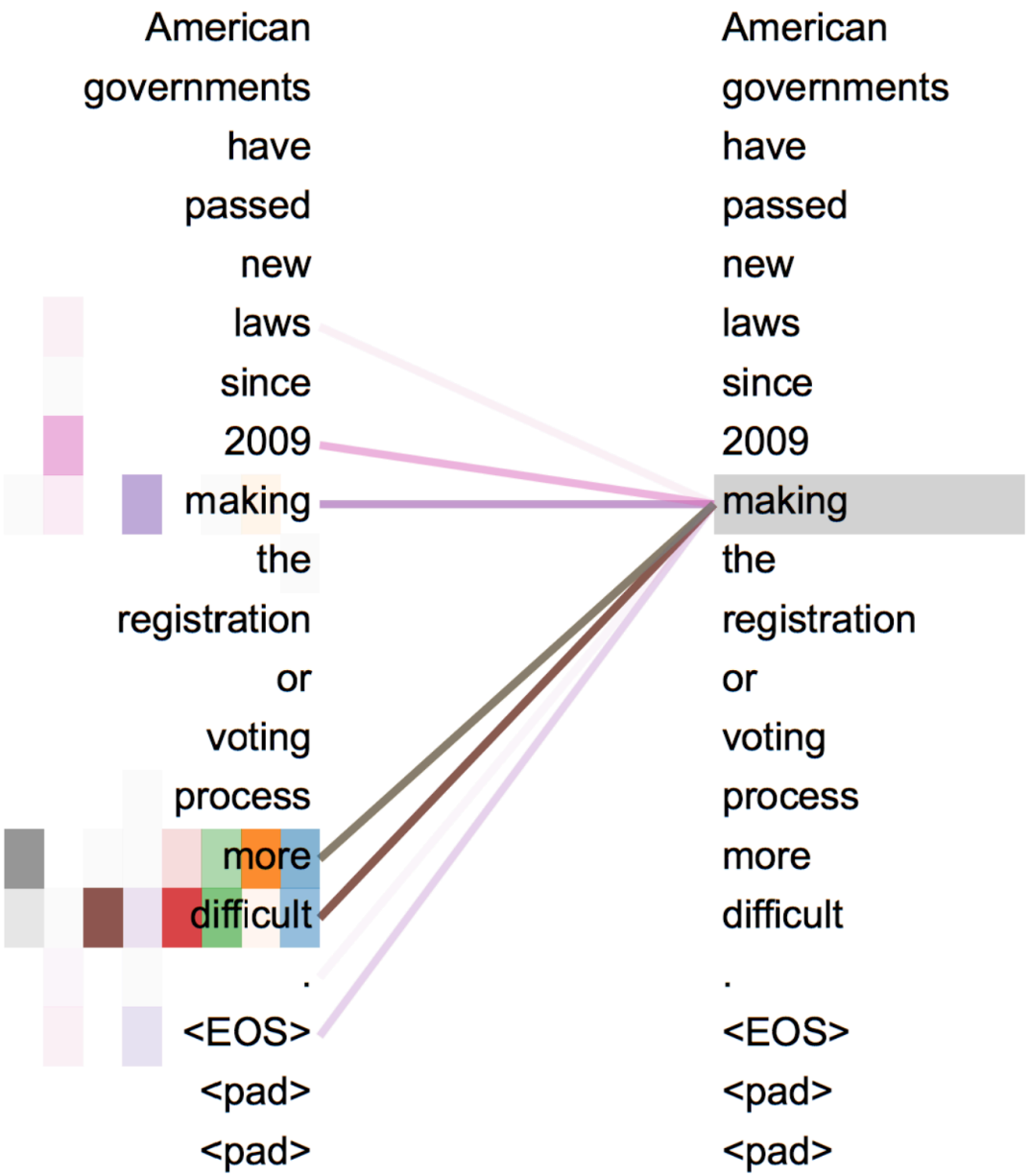
  - → addition theorems of sin, cos → PE_{pos+k}: linear function of PE_{pos}

Attention Visualization [3]

- Words start to pay attention to other words in sensible ways

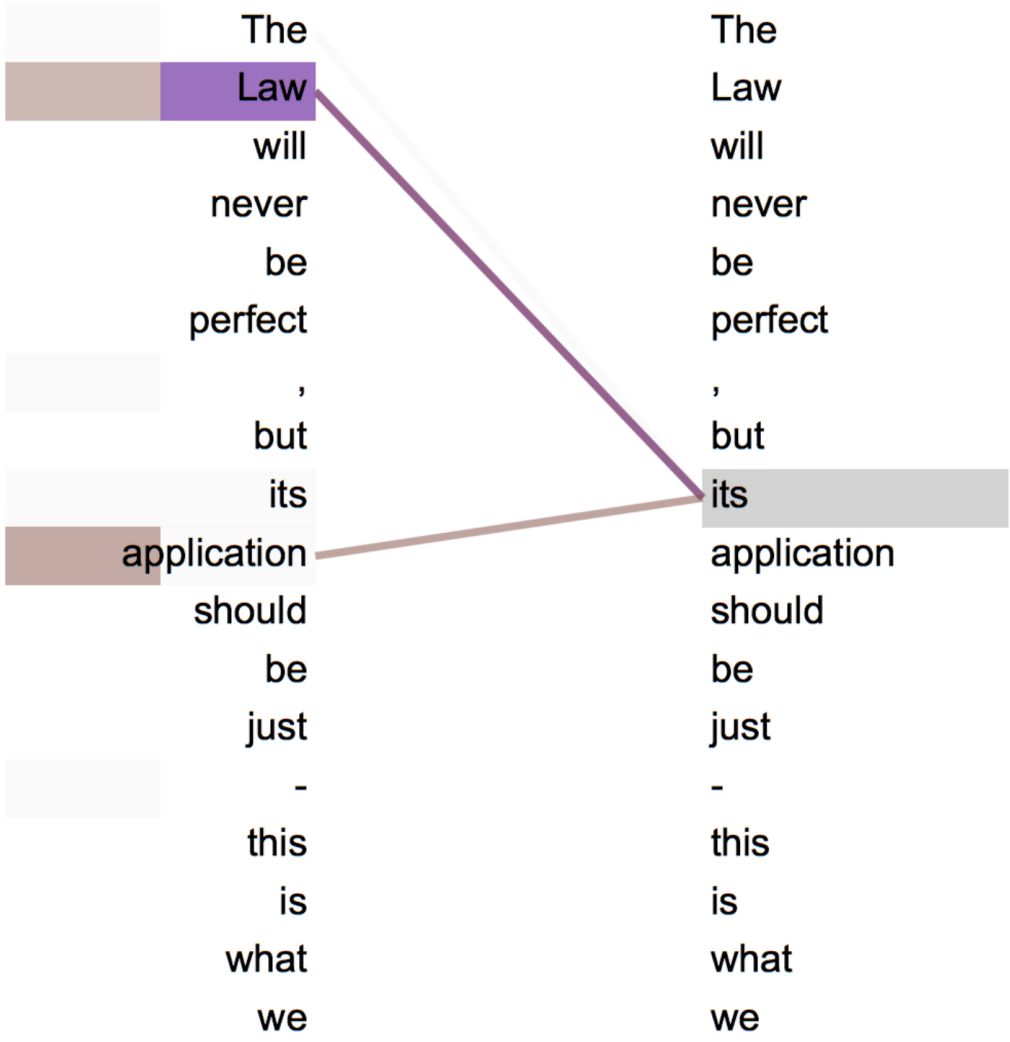
different colors = different attention heads

saturation: amount of attention

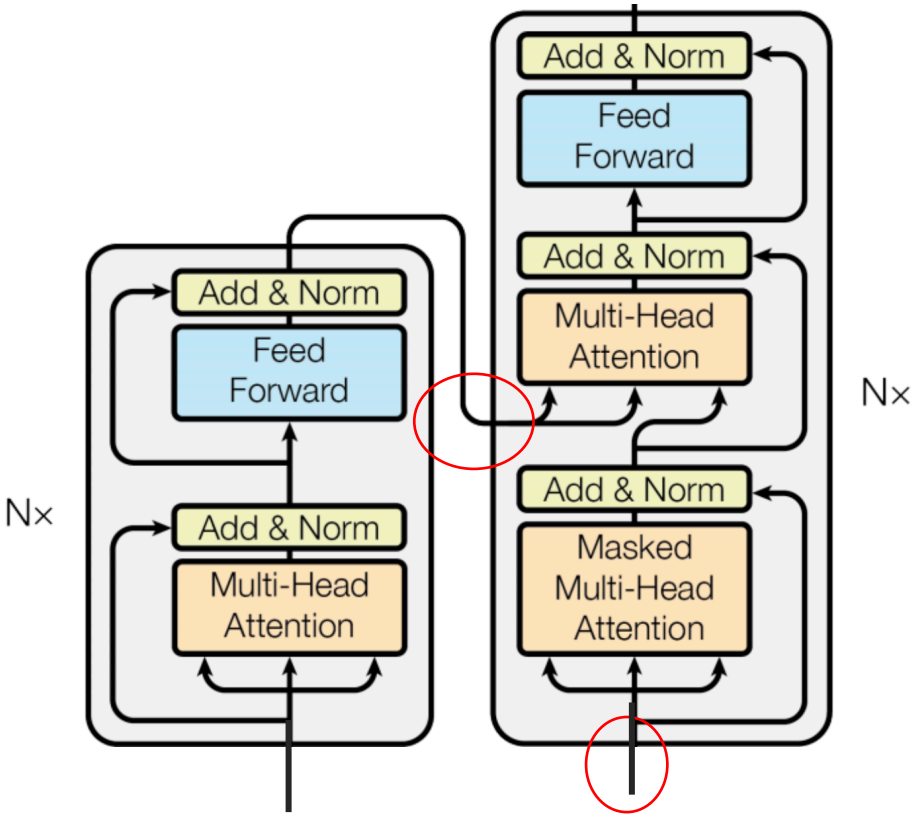


Attention Visualization [3]

- Isolated attentions from just the word ‘its’ for attention heads 5 and 6.
- attentions are very sharp for this word.

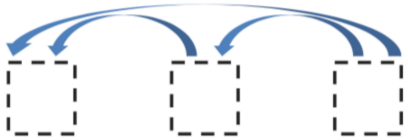


Decoder [3]



encoder decoder attention : queries from previous decoder layer; keys & values from encoder output

decoder self attention on previously generated outputs; masked to prevent information flow from future to past (ensure auto-regressive property)



Transformer Network: Overall[3]

- good performance: better BLEU scores than previous (☺) SOTA on 2014 NMT task;
  nice scores on constituency parsing (close to previous SOTA)
- [2]'s verdict: overall hard to optimize, unlike LSTMs don't work out of the box, don't yet go well together with other blocks in more complicated architectures
- detail methods of Transformer include
  - byte-pair encodings
  - checkpoint averaging (during training, average models of various times ("checkpoints" of the learning process)
  - beam search on decoder output
  - dropout regularization during training in each layer (before adding residuals)
  - label smoothing
  - fine tuning of related hyperparameters (☺)

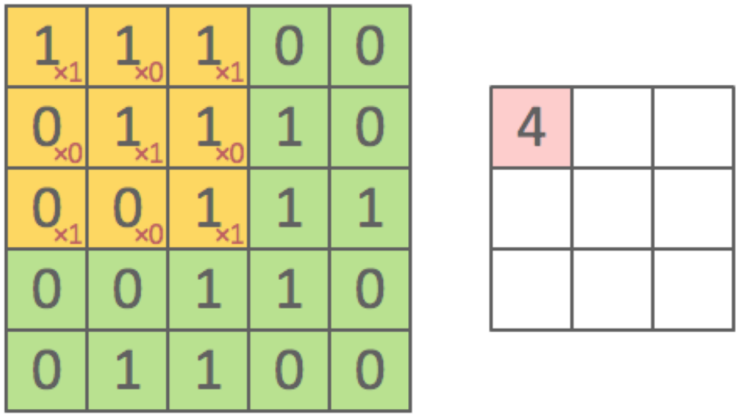
Convolutional NN for Deep NLP

- example idea: single word embeddings → embeddings for all possible phrases (computed in parallel)
- example: *the country of my birth* → 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,…*
- Not very linguistically or cognitively plausible but very fast!
- how to do that? use CNN

1d convolution:

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

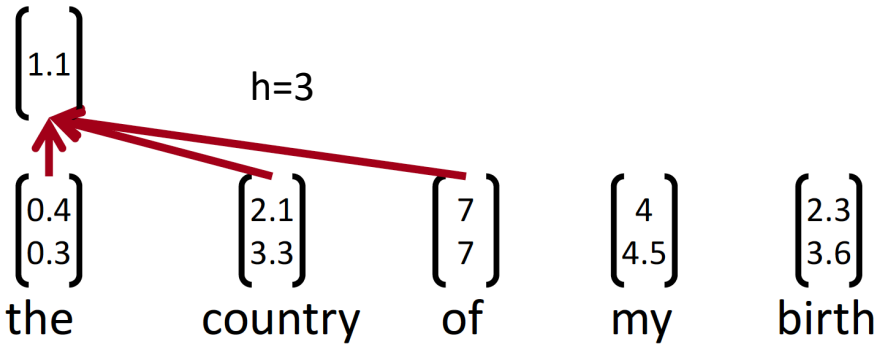
2d convolution:



Single Layer CNN [5][6]

[6]: task: sentence classification:

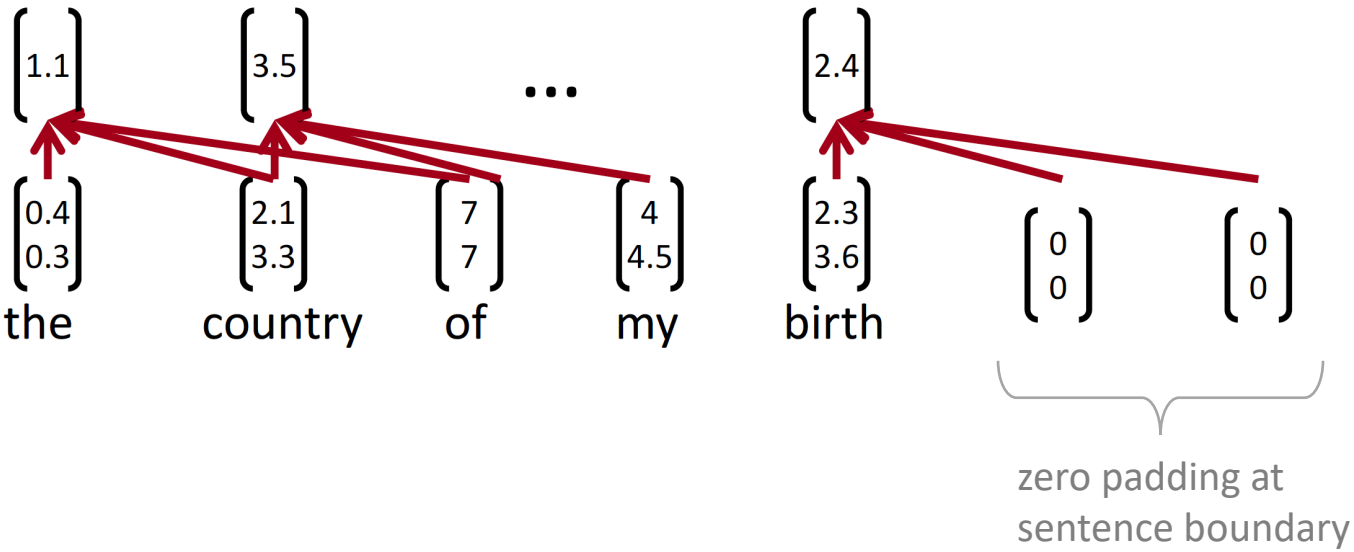
- word vectors $x_i \in \mathbb{R}^k$
- sentence: $x_{i:n}$
- phrase ("window") of length h in sentence: $x_{i:i+h-1}$
  (word vectors concatenated → vector $\boldsymbol{x}_{i:i+h-1}$)
- 1d convolutional filter: $\boldsymbol{w} \in \mathbb{R}^{hk}$ (a vector of convolution weights)
- → "features" $c_i = f(\boldsymbol{w}^T \boldsymbol{x}_{i:i+h-1} + b)$



[5]: Collobert et al.: Natural language processing (almost) from scratch. Journal of Machine Learning Research. 2011;12(Aug):2493-537.
[6]: Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Single Layer CNN [5][6]

- all possible windows of length h: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- → feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$



Single Layer CNN: Pooling [5][6]

- all possible windows of length h: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- → feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$
- pooling layer: idea: capture most important activation over time
- simple variant: max pooling: $\hat{c} = \max\{\mathbf{c}\}$

Single Layer CNN: Pooling [5][6]

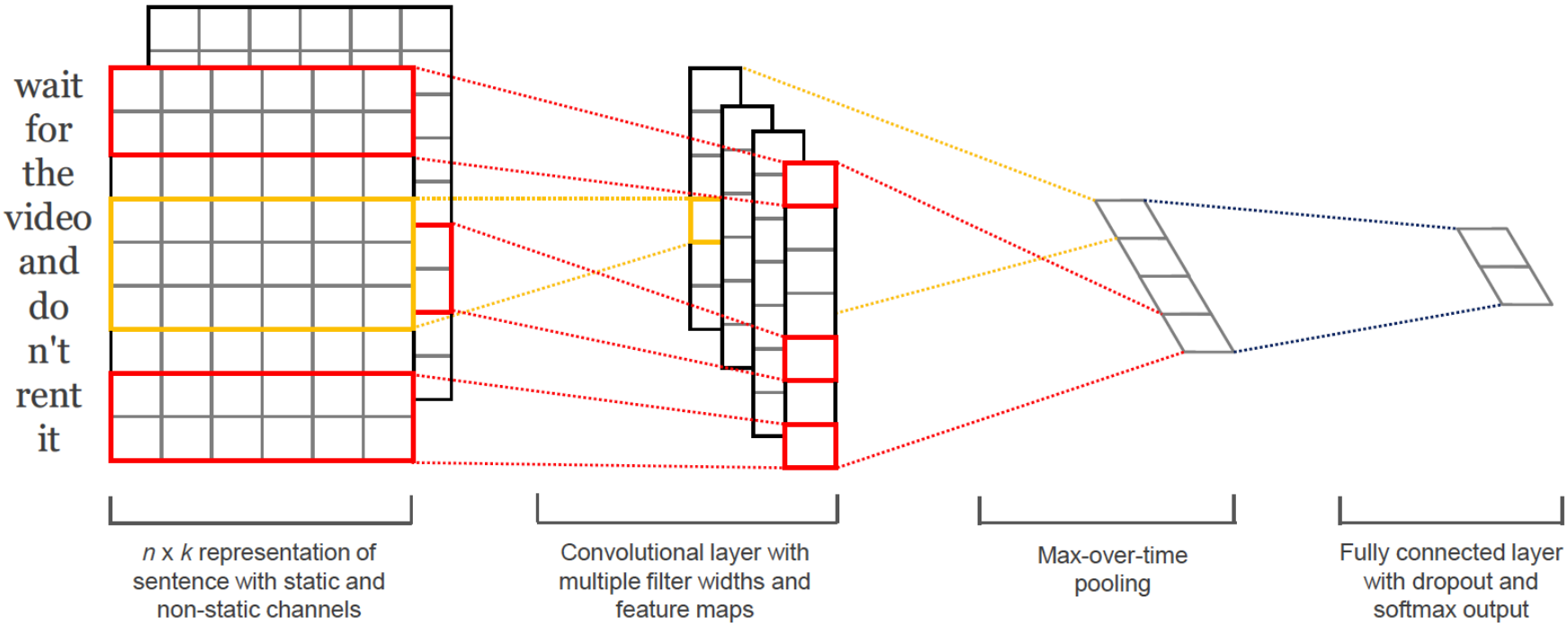
- all possible windows of length h: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- → feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$
- pooling layer: idea: capture most important activation over time
- simple variant: max pooling: $\hat{c} = \max\{\mathbf{c}\}$
- more features? → use more than one filter vector $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots$. (each with its own corresponding window length h) (e.g. filters that look at unigrams, bi-grams, 4-grams, etc.)
- → multiple feature maps $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots$. → multiple pooled features $\hat{c}_1, \hat{c}_2, \ldots$.

Multi Channel Idea + Classification [6]

- initialize with pre-trained word vectors (e.g. GloVe)
- start with two copies, backprop only to only one set, keep other "static" → two "channels"
- use both channels to compute the $\boldsymbol{c}_i$ before pooling
- simple softmax classification using the time-max-pooled features $z = [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_m]$ from the m filters:

$$y = softmax\left(W^{(S)}z + b\right)$$

Multi Channel Idea + Classification [6]



[6]

Dropout Regularization [6]

- dropout: prevents co-adaptation (overfitting to seeing specific feature constellations) (see [7])

- Idea: at training time, randomly mask/dropout/set to 0 some of the feature weights z: create masking vector r of Bernoulli random variables with probability p (a hyperparameter) of being 1

$$y = softmax\left(W^{(S)}(r \circ z) + b\right)$$

- → backpropagate only through those elements of z vector for which $r_i = 1$

- [6] reports 2 – 4% improved accuracy and ability to use very large networks without overfitting

Performance [6]

- hyperparameter set used in [6]:
  - Find hyperparameters based on dev set
  - Nonlinearity: reLu
  - Window filter sizes h = 3,4,5
  - Each filter size has 100 feature maps
  - Dropout p = 0.5
  - L2 constraint s for rows of softmax s = 3
  - Mini batch size for SGD training: 50
  - Word vectors: pre-trained with word2vec, k = 300
- experiments [6]:

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**	–	–

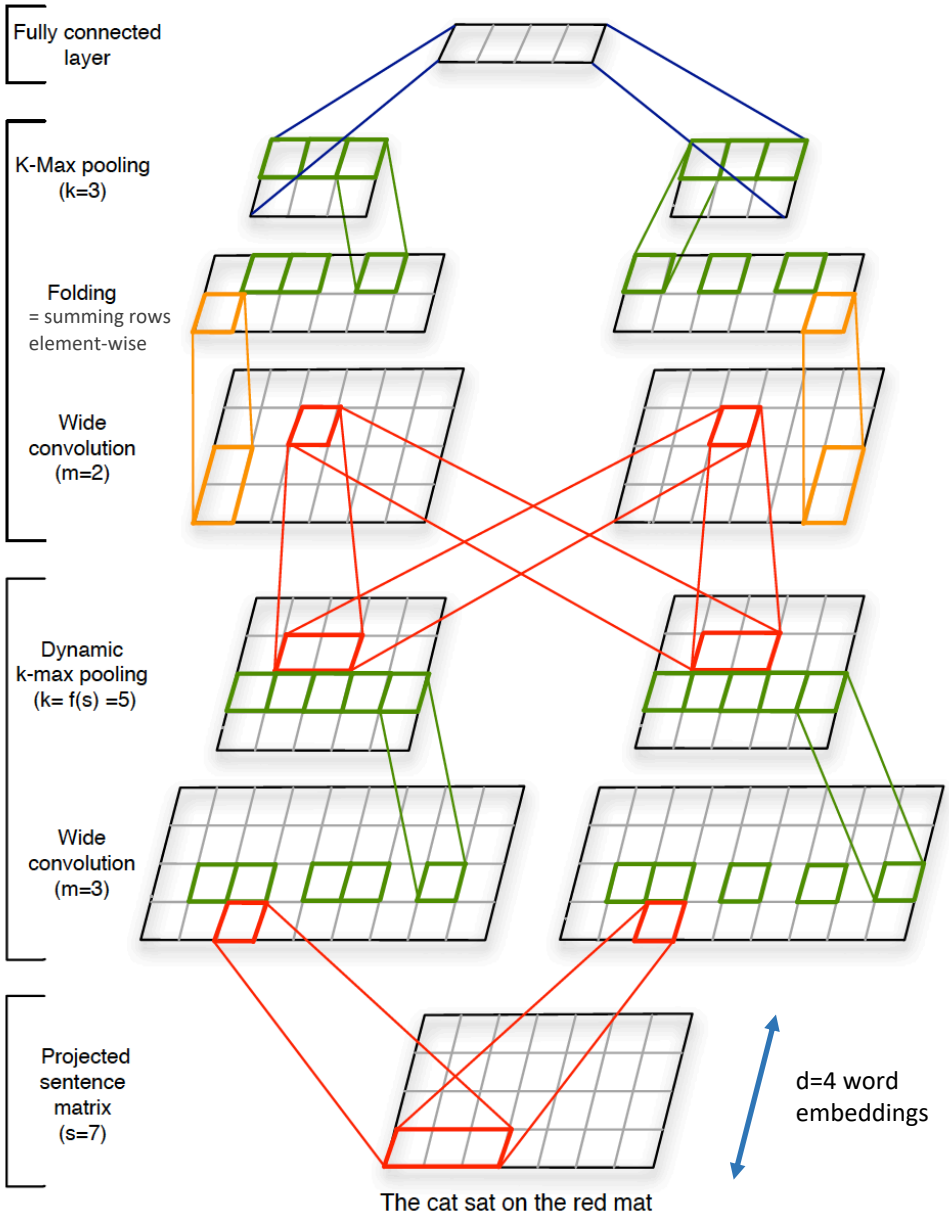
- **MR**: Movie reviews with one sentence per review. Classification involves detecting positive/negative reviews (Pang and Lee, 2005).[3]
- **SST-1**: Stanford Sentiment Treebank—an extension of MR but with train/dev/test splits provided and fine-grained labels (very positive, positive, neutral, negative, very negative), re-labeled by Socher et al. (2013).[4]
- **SST-2**: Same as SST-1 but with neutral reviews removed and binary labels.
- **Subj**: Subjectivity dataset where the task is to classify a sentence as being subjective or objective (Pang and Lee, 2004).
- **TREC**: TREC question dataset—task involves classifying a question into 6 question types (whether the question is about person, location, numeric information, etc.) (Li and Roth, 2002).[5]
- **CR**: Customer reviews of various products (cameras, MP3s etc.). Task is to predict positive/negative reviews (Hu and Liu, 2004).[6]

- **CNN-rand**: Our baseline model where all words are randomly initialized and then modified during training.
- **CNN-static**: A model with pre-trained vectors from `word2vec`. All words—including the unknown ones that are randomly initialized—are kept static and only the other parameters of the model are learned.
- **CNN-non-static**: Same as above but the pre-trained vectors are fine-tuned for each task.
- **CNN-multichannel**: A model with two sets of word vectors. Each set of vectors is treated as a 'channel' and each filter is applied to both channels, but gradients are back-propagated only through one of the channels. Hence the model is able to fine-tune one set of vectors while keeping the other static. Both channels are initialized with `word2vec`.

CNN Alternatives [8]

- narrow convolution (do not pad with zeros at beginning or end of sentence) vs wide convolution (use zero padding)

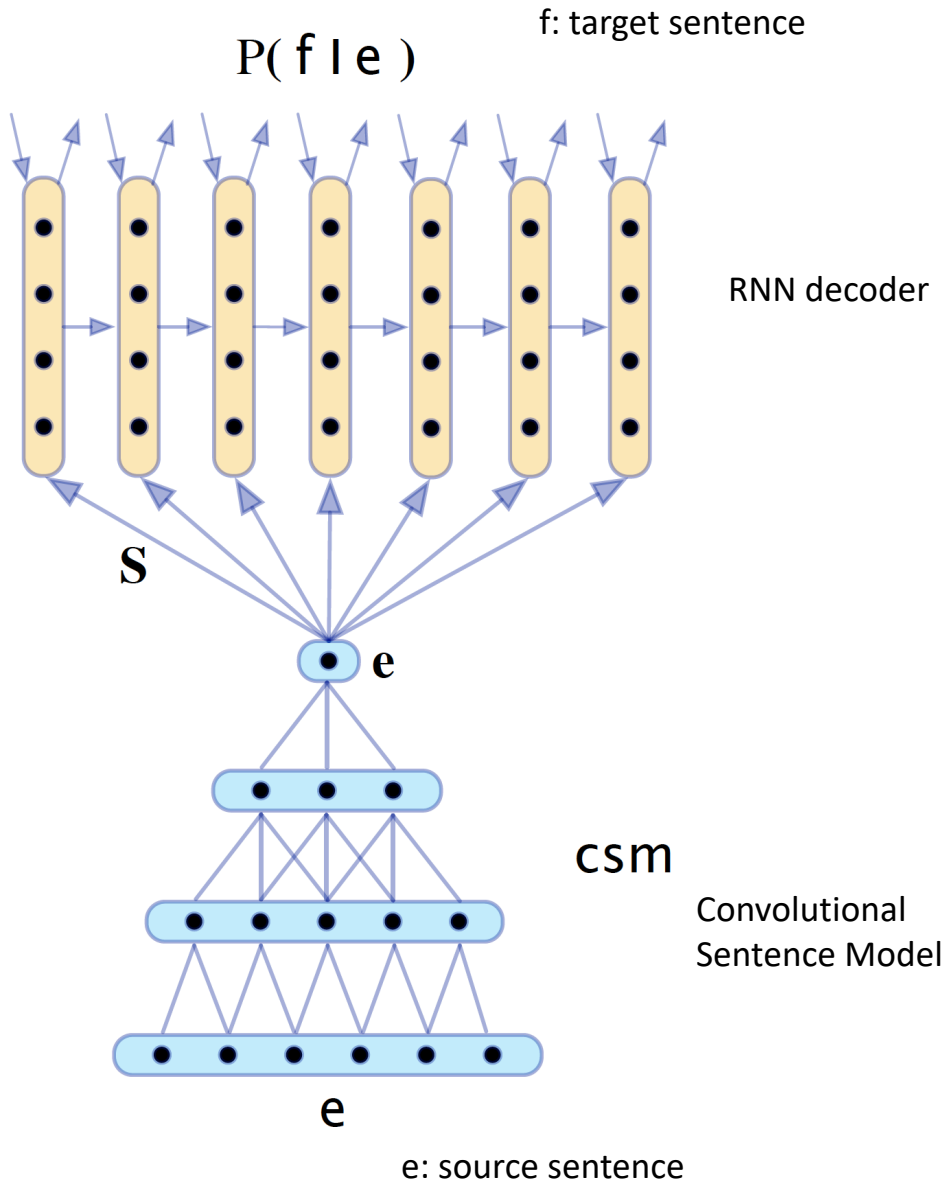
- complex pooling schemes
  - (k-max-pooling: select the k largest features
  - dynamic k-max pooling: k is a function of sentence length and number of layers)
- and/or deeper convolutional layers



[8]: Kalchbrenner, et al.. "A convolutional neural network for modelling sentences." *arXiv preprint arXiv:1404.2188* (2014).

CNN Application [9]

- [9]: One of the first successful NMT efforts
- Uses CNN for encoding and RNN for decoding



[9]: Kalchbrenner, N. and Blunsom, P., 2013. Recurrent continuous translation models. (2013)

Bibliography

(2) Richard Socher et al: “CS224n: Natural Language Processing with Deep Learning”, Lecture Materials (slides and links to background reading) http://web.stanford.edu/class/cs224n/ (URL, May 2018), 2018

(3) Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention is all you need. In Advances in Neural Information Processing Systems (pp. 5998-6008) and arXiv:1706.03762v5

(4) Ba, J. L., Kiros, J. R. and Hinton, G. E. (2016) Layer normalization. arXiv preprint arXiv:1607.06450

(5) Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural language processing (almost) from scratch. Journal of Machine Learning Research. 2011;12(Aug):2493-537.

(6) Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

(7) Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580. 2012 Jul 3.

Bibliography

(8) Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences." *arXiv preprint arXiv:1404.2188* (2014).

(9) Kalchbrenner, N. and Blunsom, P., 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1700-1709).

(10) J. Alammar: The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/ (URL, Dec. 2018), Blog Article, 2018.

(11) A. Rush et al.: The Annotated Transformer; https://nlp.seas.harvard.edu/2018/04/03/attention.html (URL, Jan 2020), Blog Article with PyTorch code, 2018

Recommendations for Studying

- minimal approach:

  work with the slides and understand their contents! Think beyond instead of merely memorizing the contents

- standard approach:

  minimal approach + read the corresponding paragraphs of the papers corresponding to the discussed example systems:

  - [3]* sections 1-4
    or [10]
  - [6] section 1, section 2
  - [8] section 3

  *: especially important

- interested / deeply interested student's approach:

  standard approach + study the few omitted elements of the corresponding lecture slides from [2] + read all of the recommended background reading of [2] for lecture 12