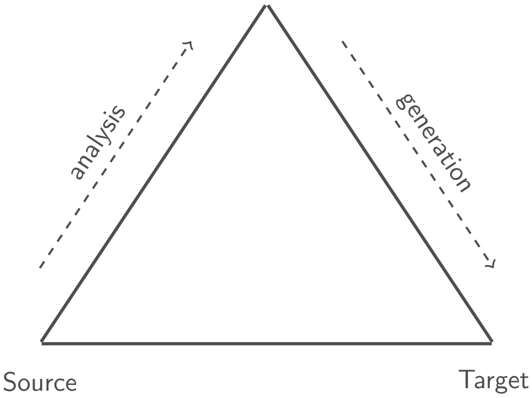
Neural Machine Translation

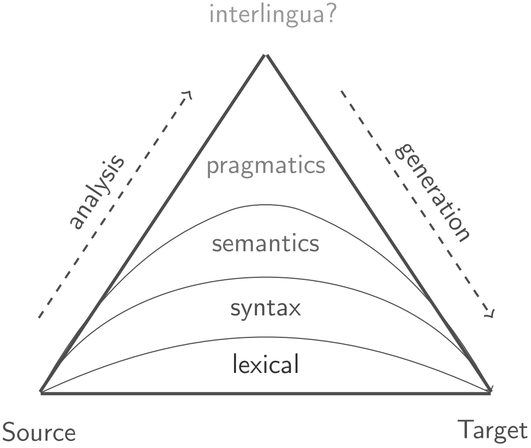
COMS W4705: Natural Language Processing

Kapil Thadani
kapil@cs.columbia.edu

Review: Machine Translation



Review: Machine Translation

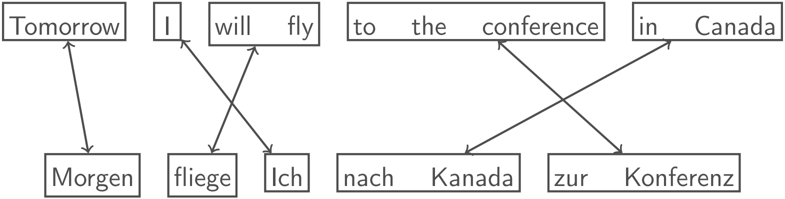


Review: Phrase-based MT

Tomorrow I will fly to the conference in Canada

Morgen fliege Ich nach Kanada zur Konferenz

Review: Phrase-based MT



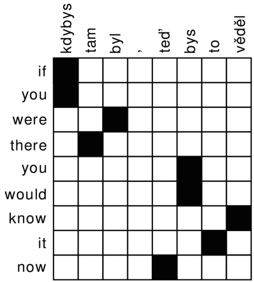
Review: Phrase-based MT

1. Collect bilingual dataset $\langle S_i, T_i \rangle \in \mathcal{D}$

2. Unsupervised phrase-based alignment
   - phrase table π

3. Unsupervised n-gram language modeling
   - language model ψ

4. Supervised decoder
   - parameters θ



$$\widehat{T} = \arg\max_T \; p(T|S)$$
$$= \arg\max_T \; p(S|T, \pi, \theta) \cdot p(T|\psi)$$

Neural MT

1. Collect bilingual dataset $\langle S_i, T_i \rangle \in \mathcal{D}$
2. Unsupervised phrase-based alignment
   - phrase table π
3. Unsupervised n-gram language modeling
   - language model ψ
4. Supervised encoder-decoder framework
   - parameters θ

Outline

- Encoder-decoder architectures
  - RNN encoders & decoders
  - Sequence-to-sequence models
  - LSTMs & GRUs
- Attention mechanism
  - Dynamic contexts
  - Induced alignments
- Scaling up
  - Google NMT
  - Sub-word units
  - Sequence-level training
  - Multilingual translation
- Transformers
  - Self-attention
  - Induced structure

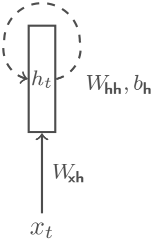
Outline

- Encoder-decoder architectures
  - RNN encoders & decoders
  - Sequence-to-sequence models
  - LSTMs & GRUs
- Attention mechanism
  - Dynamic contexts
  - Induced alignments
- Scaling up
  - Google NMT
  - Sub-word units
  - Sequence-level training
  - Multilingual translation
- Transformers
  - Self-attention
  - Induced structure

Notation: Basic recurrent unit

- Repeatedly apply a non-linear transformation to sequential inputs
- Optionally produce an output from hidden states

Hidden state



Input vector

$$h_t = \phi_{\mathbf{h}}(W_{\mathbf{xh}}^\top x_t + W_{\mathbf{hh}}^\top h_{t-1} + b_{\mathbf{h}})$$

(typically sigmoid or tanh)

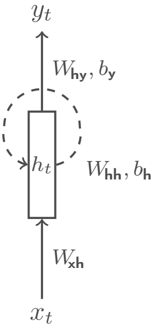
Notation: Basic recurrent unit

- Repeatedly apply a non-linear transformation to sequential inputs
- Optionally produce an output from hidden states

Output vector

Hidden state

Input vector



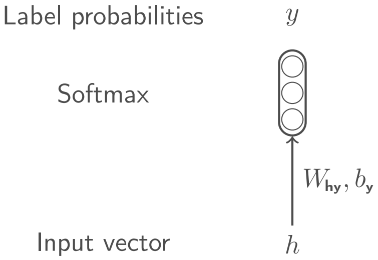
$$y_t = \phi_{\mathbf{y}}(W_{\mathbf{hy}}^\top h_t + b_{\mathbf{y}})$$

$$h_t = \phi_{\mathbf{h}}(W_{\mathbf{xh}}^\top x_t + W_{\mathbf{hh}}^\top h_{t-1} + b_{\mathbf{h}})$$

(typically sigmoid or tanh)

Notation: Softmax

- Typical output layer for multiclass classification
- Produces scores y such that $\sum_i y_i = 1$



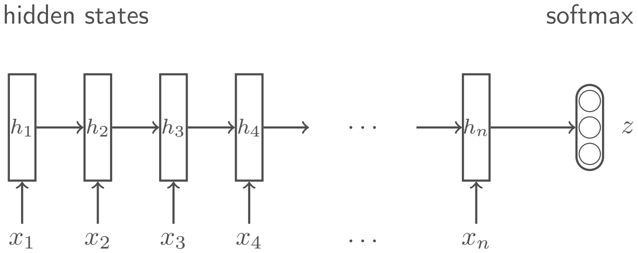
$$z = W_{\mathbf{hy}}^{\top} h + b_{\mathbf{y}}$$

$$\begin{aligned} y_i &= \frac{e^{z_i}}{\sum_j e^{z_j}} \\ &= p(\text{label} = i \mid h) \end{aligned}$$

RNN classifier

Input words $x_1, \ldots, x_n$

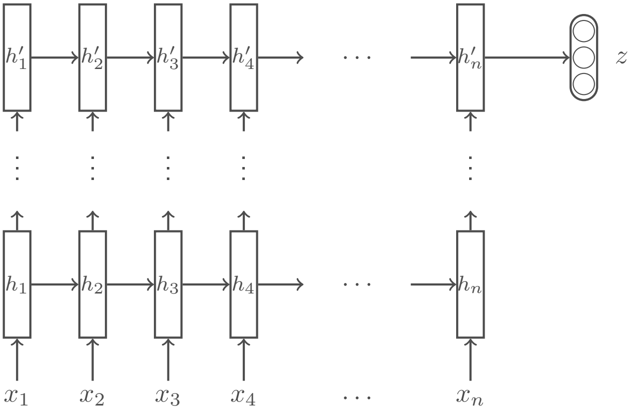
Output category label z



Deep RNN classifier

Input words $x_1, \ldots, x_n$

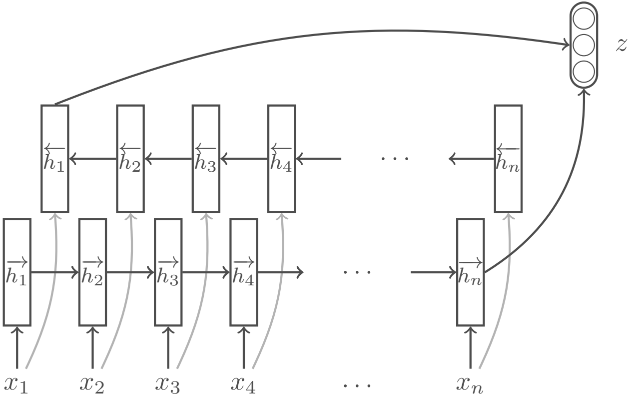
Output category label z



Bidirectional RNN classifier

Input words $x_1, \ldots, x_n$

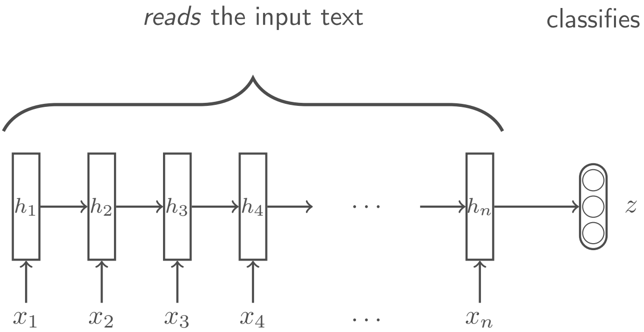
Output category label z



RNN classifier

Input words $x_1, \ldots, x_n$

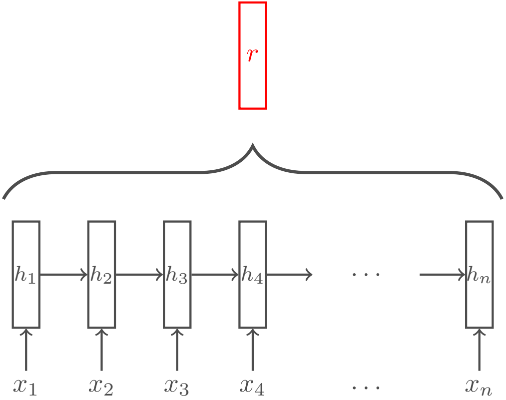
Output category label z



RNN encoder

Input words $x_1, \ldots, x_n$

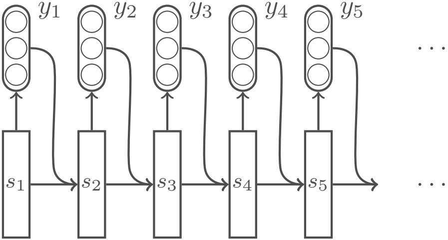
Output representation r



RNN language model

Input words $y_1, \dots, y_k$

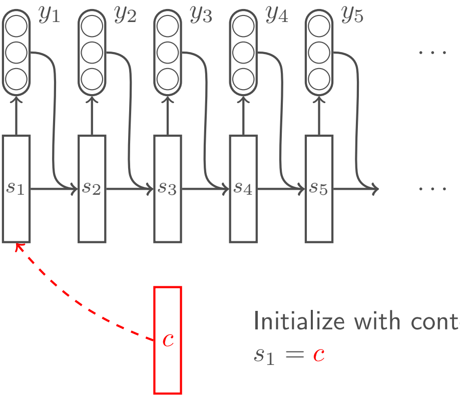
Output following words $y_k, \dots, y_m$



RNN decoder

Input context vector c

Output words $y_1, \ldots, y_m$



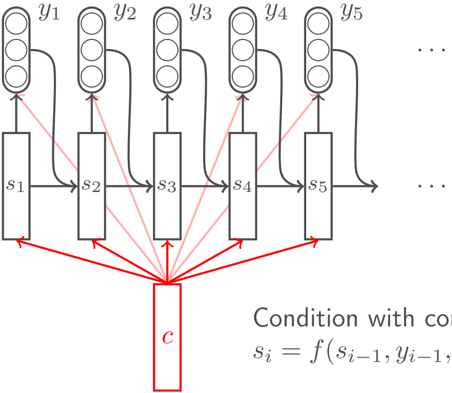
Initialize with context

$s_1 = c$

RNN decoder

Input context vector c

Output words $y_1, \ldots, y_m$



Condition with context
$s_i = f(s_{i-1}, y_{i-1}, c)$

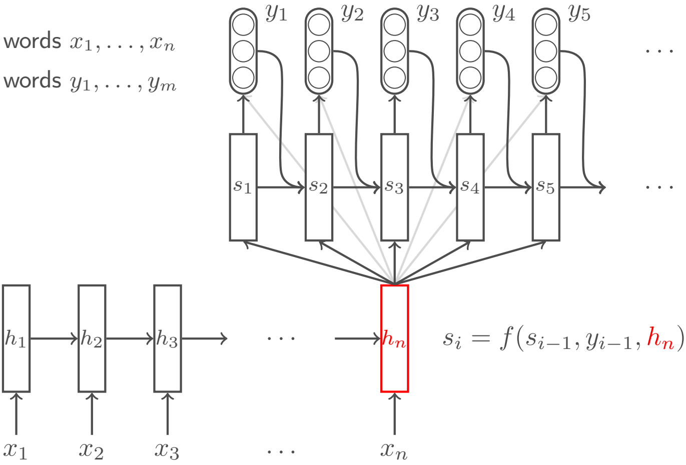
Sequence-to-sequence models

- Introduced in Sutskever et al. (2014) and Cho et al. (2014)
- Combine a sequence encoder for the source language with a sequence decoder for the target language
  1. Encode source language tokens until `<EOS>` obtained
  2. Use final encoder hidden state as context vector
  3. Decode target language tokens until `<EOS>` obtained
- Use gated units (LSTMs or GRUs) to overcome vanishing gradients
- Beam search decoding through softmax scores

Sequence-to-sequence learning

Input words $x_1, \ldots, x_n$

Output words $y_1, \ldots, y_m$



$$s_i = f(s_{i-1}, y_{i-1}, h_n)$$

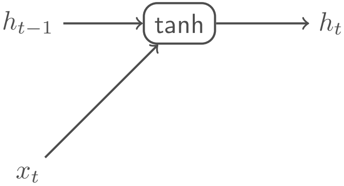
Long short-term memory (LSTM)

- Backpropagation through repeated non-linear transformations (sigmoid, tanh) leads to vanishing gradients
  - RNNs cannot easily model long-range dependencies
  - Performance degrades with longer sequences

- LSTM (Hochreiter & Schmidhuber, 1997) adds a memory cell which is only affected by linear interactions

- Gates with sigmoid activations are used to modulate:
  - additions from the current input (input gate)
  - contributions to the next hidden state (output gate)
  - the amount of memory decayed (forget gate) (Gers et al., 1999)

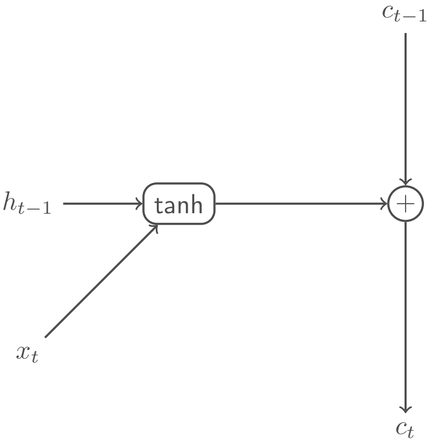
Long short-term memory (LSTM)



$$h_t = \mathsf{tanh}(W_{\mathbf{xh}}^\top x_t + W_{\mathbf{hh}}^\top h_{t-1})$$

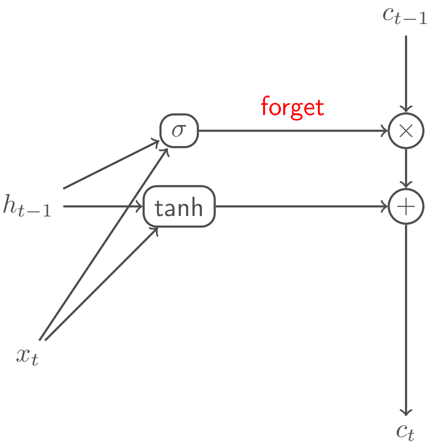
(normal RNN)

Long short-term memory (LSTM)



$$\tilde{c}_t = \mathsf{tanh}(W_{\mathbf{xh}}^{\top} x_t + W_{\mathbf{hh}}^{\top} h_{t-1})$$
$$c_t = c_{t-1} + \tilde{c}_t$$

Long short-term memory (LSTM)

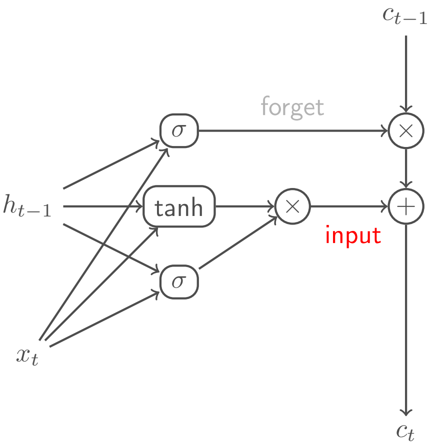


$$\textcolor{red}{f_t} = \sigma(W_{\mathbf{fx}}^\top x_t + W_{\mathbf{fh}}^\top h_{t-1})$$

$$\tilde{c}_t = \tanh(W_{\mathbf{xh}}^\top x_t + W_{\mathbf{hh}}^\top h_{t-1})$$

$$c_t = \textcolor{red}{f_t} \odot c_{t-1} + \tilde{c}_t$$

Long short-term memory (LSTM)



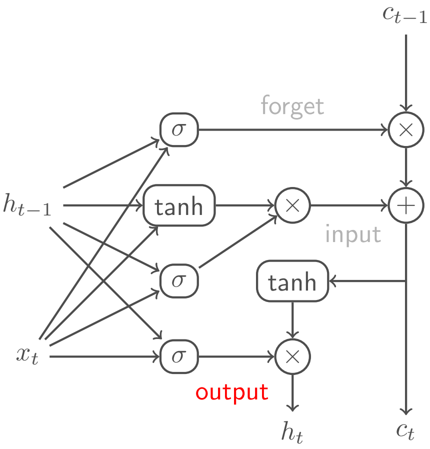
$$f_t = \sigma(W_{\mathsf{fx}}^\top x_t + W_{\mathsf{fh}}^\top h_{t-1})$$

$$i_t = \sigma(W_{\mathsf{ix}}^\top x_t + W_{\mathsf{ih}}^\top h_{t-1})$$

$$\tilde{c}_t = \mathrm{tanh}(W_{\mathsf{xh}}^\top x_t + W_{\mathsf{hh}}^\top h_{t-1})$$

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

Long short-term memory (LSTM)



$$f_t = \sigma(W_{\mathsf{fx}}^\top x_t + W_{\mathsf{fh}}^\top h_{t-1})$$

$$i_t = \sigma(W_{\mathsf{ix}}^\top x_t + W_{\mathsf{ih}}^\top h_{t-1})$$

$$o_t = \sigma(W_{\mathsf{ox}}^\top x_t + W_{\mathsf{oh}}^\top h_{t-1})$$

$$\tilde{c}_t = \mathsf{tanh}(W_{\mathsf{xh}}^\top x_t + W_{\mathsf{hh}}^\top h_{t-1})$$

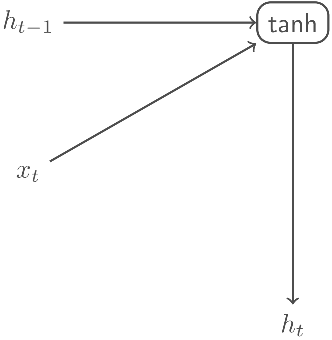
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot \mathsf{tanh}(c_t)$$

Gated Recurrent Unit (GRU)

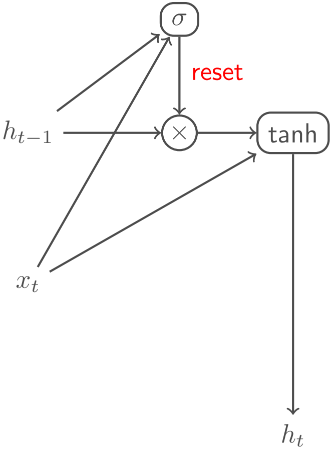
- Inspired by LSTM but with no memory cell (Cho et al., 2014)
- Gates with sigmoid activations are used to control:
  - contributions of the previous hidden state to a new state (reset gate)
  - the balance between previous and new states for the next hidden state (update gate)
- Requires fewer parameters but performs similarly to LSTM in practice (Chung et al., 2014)

Gated Recurrent Unit (GRU)



$$\tilde{h}_t = \mathsf{tanh}(W_{\mathbf{xh}}^\top x_t + W_{\mathbf{hh}}^\top h_{t-1})$$
$$h_t = \tilde{h}_t$$

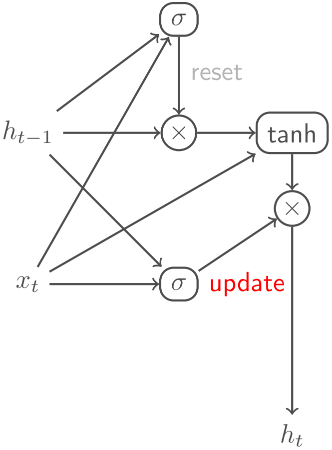
Gated Recurrent Unit (GRU)



$$r_t = \sigma(W_{\mathbf{rx}}^\top x_t + W_{\mathbf{rh}}^\top h_{t-1})$$

$$\tilde{h}_t = \tanh(W_{\mathbf{xh}}^\top x_t + W_{\mathbf{hh}}^\top (r_t \odot h_{t-1}))$$
$$h_t = \tilde{h}_t$$

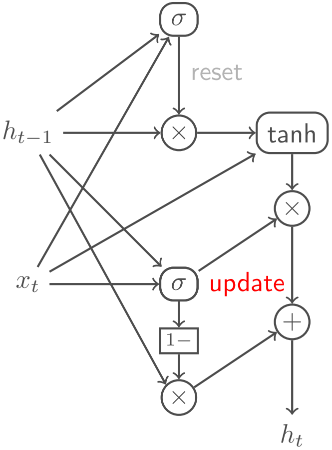
Gated Recurrent Unit (GRU)



$$r_t = \sigma(W_{\mathsf{rx}}^\top x_t + W_{\mathsf{rh}}^\top h_{t-1})$$
$$z_t = \sigma(W_{\mathsf{zx}}^\top x_t + W_{\mathsf{zh}}^\top h_{t-1})$$

$$\tilde{h}_t = \tanh(W_{\mathsf{xh}}^\top x_t + W_{\mathsf{hh}}^\top (r_t \odot h_{t-1}))$$
$$h_t = z_t \odot \tilde{h}_t$$

Gated Recurrent Unit (GRU)

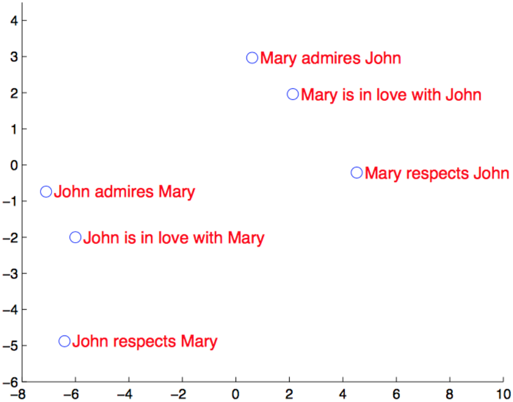


$$r_t = \sigma(W_{\mathbf{rx}}^\top x_t + W_{\mathbf{rh}}^\top h_{t-1})$$
$$z_t = \sigma(W_{\mathbf{zx}}^\top x_t + W_{\mathbf{zh}}^\top h_{t-1})$$

$$\tilde{h}_t = \tanh(W_{\mathbf{xh}}^\top x_t + W_{\mathbf{hh}}^\top (r_t \odot h_{t-1}))$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

Sentence embeddings

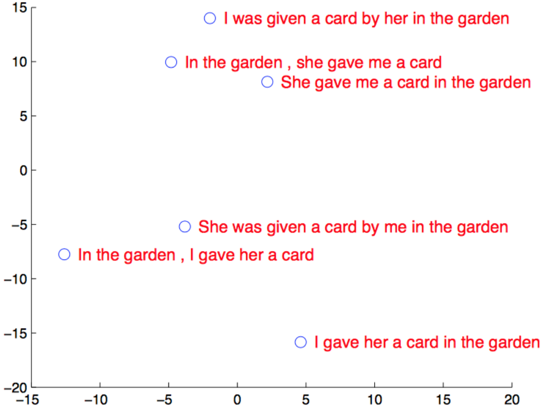
2-D PCA projections of encoded vectors for sentences



Sutskever et al. (2014)

Sentence embeddings

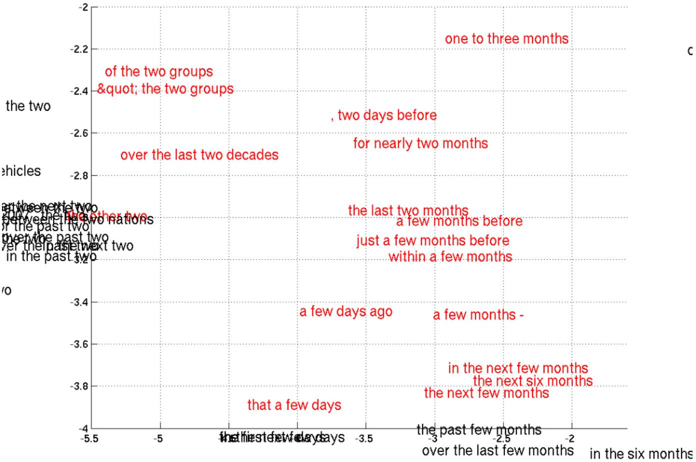
2-D PCA projections of encoded vectors for sentences



Sutskever et al. (2014)

Phrase embeddings

2-D Barnes-Hut projections of encoded vectors for phrases



Cho et al. (2014)

Sequence-to-sequence models

- + First end-to-end neural architecture for machine translation
- + No alignments required, just parallel data
- + Encoders produce meaningful sentence embeddings

- – Does not outperform phrase-based MT techniques
- – Performance degrades for longer sentences
- – Need to reverse the input for better performance

Method	**test BLEU score (ntst14)**
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59

Outline

- Encoder-decoder architectures
  - RNN encoders & decoders
  - Sequence-to-sequence models
  - LSTMs & GRUs
- Attention mechanism
  - Dynamic contexts
  - Induced alignments
- Scaling up
  - Google NMT
  - Sub-word units
  - Sequence-level training
  - Multilingual translation
- Transformers
  - Self-attention
  - Induced structure

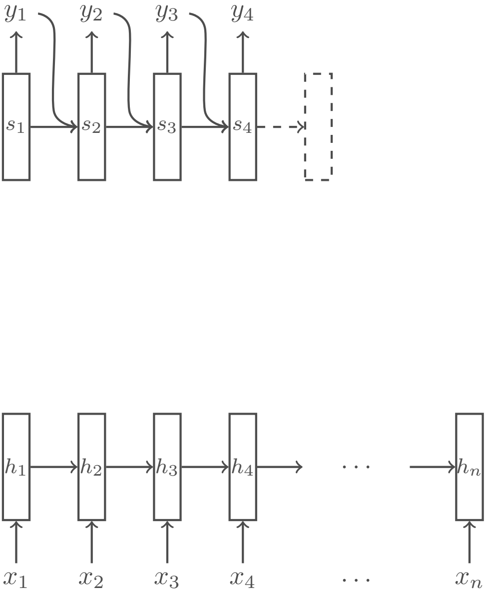
Attention mechanism

- Fixed context vector is a bottleneck for performance in encoder-decoder architectures

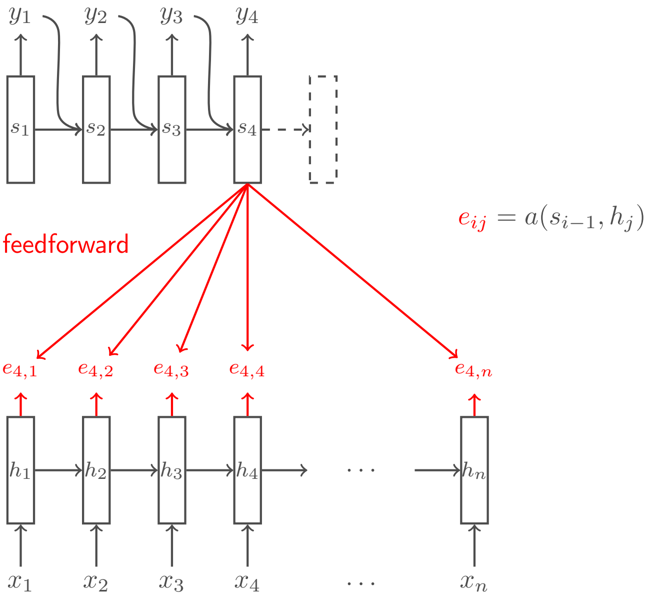
- Bahdanau et al. (2015) introduce a dynamic context vector that changes with each decoder timestep
  - Weighted average over all encoder hidden states
  - Weights ("attention") conditioned on current decoder hidden state

- Allows gradients to flow directly from decoding errors to relevant encoder hidden states, thus robust to vanishing gradients

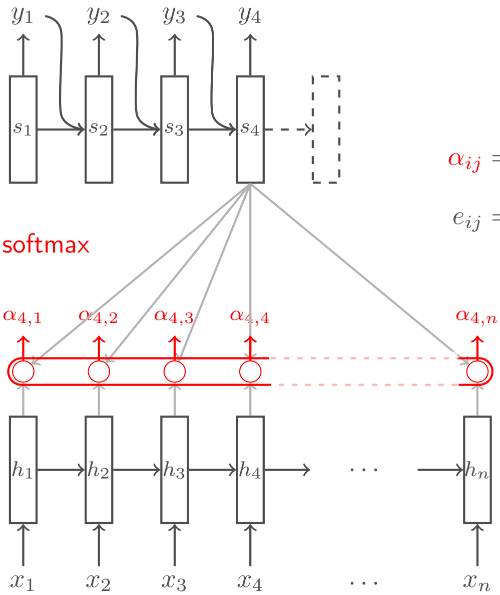
Attention-based translation



Attention-based translation



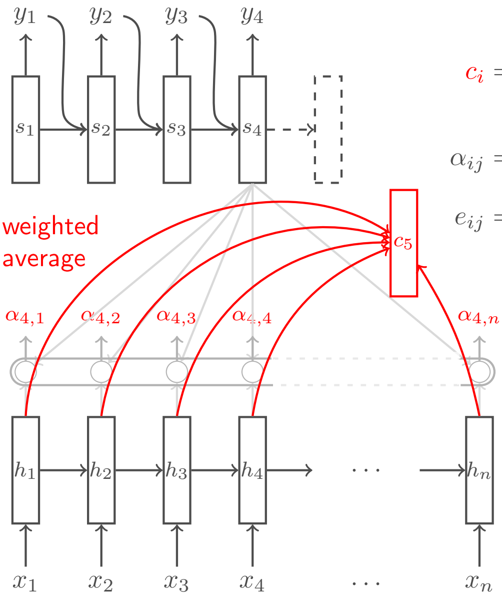
Attention-based translation



$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Attention-based translation

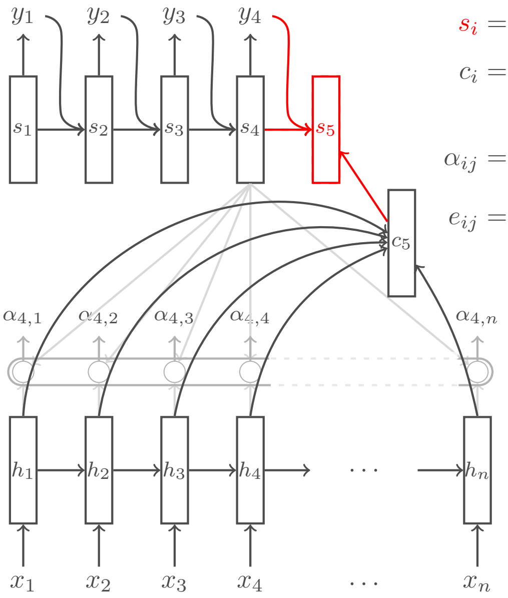


$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Attention-based translation



$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

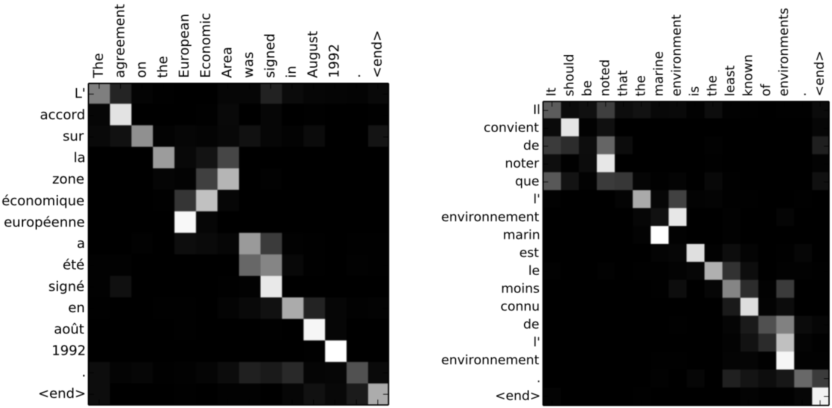
$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Induced alignments

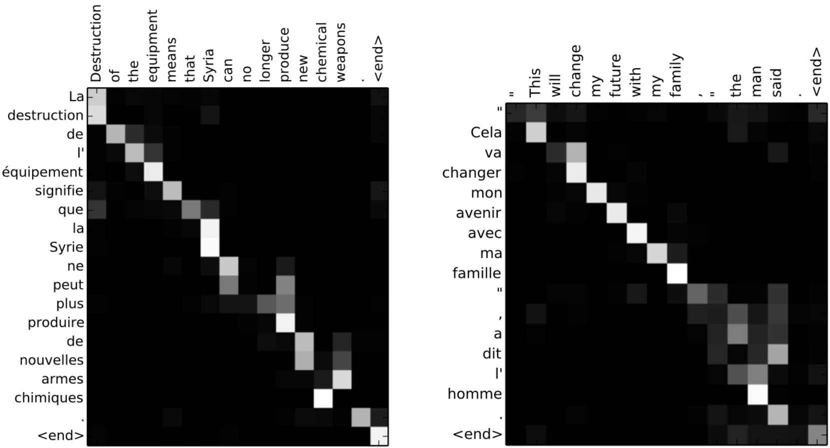
Attention weights α_{ij} reveal alignments between source & target words



Bahdanau et al. (2015)

Induced alignments

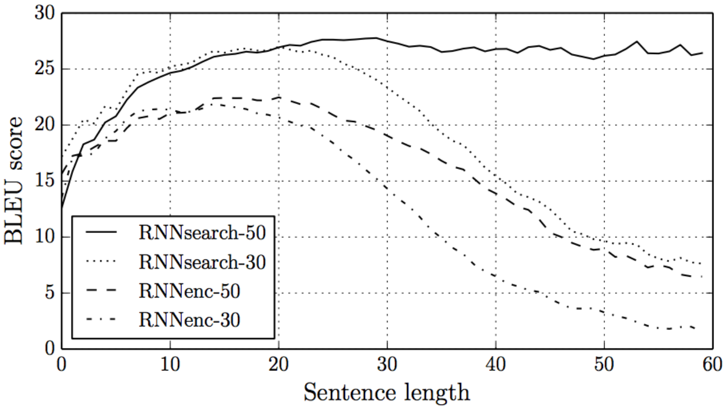
Attention weights α_{ij} reveal alignments between source & target words



Bahdanau et al. (2015)

Attention-based translation

Consistent performance as sentence length increases



Bahdanau et al. (2015)

Attention-based translation

+ Gradients can be backpropagated directly to attended regions, avoiding vanishing gradients with long sequences
+ Attention weights α_{ij} can be visualized to diagnose errors
+ Performance competitive with phrase-based MT

Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

– Runtime for inference is $\mathcal{O}(mn)$ instead of $\mathcal{O}(m+n)$ without attention

Outline

- Encoder-decoder architectures
  - RNN encoders & decoders
  - Sequence-to-sequence models
  - LSTMs & GRUs
- Attention mechanism
  - Dynamic contexts
  - Induced alignments
- Scaling up
  - Google NMT
  - Sub-word units
  - Sequence-level training
  - Multilingual translation
- Transformers
  - Self-attention
  - Induced structure

Scaling up

- Practical translation systems typically rely on phrase-based MT
  - NMT does scale easily to large vocabularies and rare words
  - Slower inference for large neural networks
  - NMT sometimes fails to fully translate all of the input

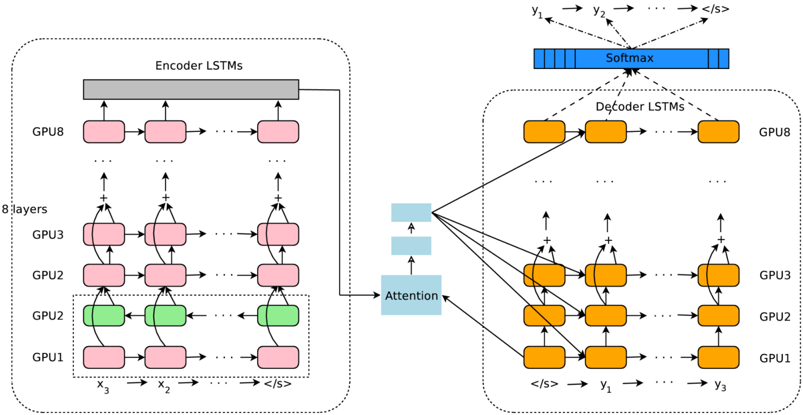
- Wu et al. (2016) describes a production-grade NMT system evaluated against phrase-based MT for Google Translate

	PBMT	GNMT	Human	Relative Improvement
English $\rightarrow$ Spanish	4.885	5.428	5.504	87%
English $\rightarrow$ French	4.932	5.295	5.496	64%
English $\rightarrow$ Chinese	4.035	4.594	4.987	58%
Spanish $\rightarrow$ English	4.872	5.187	5.372	63%
French $\rightarrow$ English	5.046	5.343	5.404	83%
Chinese $\rightarrow$ English	3.694	4.263	4.636	60%

Scaling up: GNMT

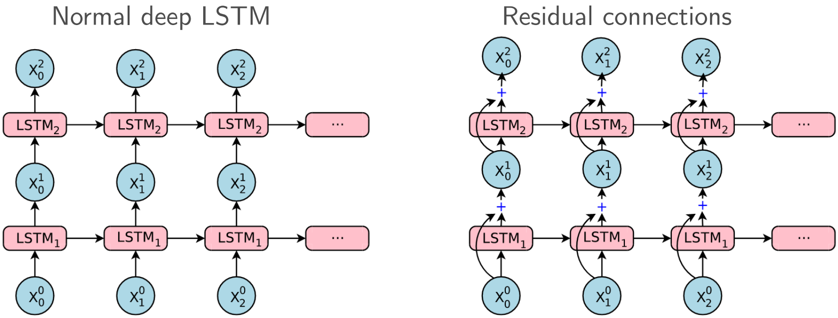
- Sequence-to-sequence model with attention (Wu et al., 2016)
  Encoder: 8 LSTM layers; bottom layer bidirectional
  Decoder: 8 LSTM layers; bottom layer provides attention context

- All layers loaded on separate GPUs



Scaling up: Residual connections

- Stacked LSTMs with residual connections (He et al., 2015)
  - Layer inputs added element-wise to outputs
  - Activations model *differences* between layer inputs and targets
  - More robust to vanishing gradients in deep architectures



Scaling up: Sub-word units

- Infrequent words replaced with sub-words to reduce vocabulary

Jet makers feud over seat width with big orders at stake
$\Downarrow$
_J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

- Various corpus-based techniques to identify sub-words including
  - WordPieceModel (Schuster & Nakajima, 2012)
  - Byte Pair Encoding (Sennrich et al., 2016)

- Available implementations:
  - sentencepiece
  - subword-nmt

Scaling up: Sequence-level training

- NMT models are trained on the word level with cross-entropy loss but evaluated with *sequence-level* metrics like BLEU, which are non-differentiable

- Model parameters θ can also be refined against any non-differentiable measure $R(x, y)$ using reinforcement learning

$$\begin{aligned}
\nabla_\theta \, \mathbb{E}_{\mathcal{D}} \left[R(x, y) \right] &= \sum_{\langle x,y \rangle \in \mathcal{D}} R(x, y) \cdot \nabla_\theta \, p(y|x;\theta) \\
&= \sum_{\langle x,y \rangle \in \mathcal{D}} R(x, y) \cdot \nabla_\theta \, p(y|x;\theta) \cdot \frac{p(y|x;\theta)}{p(y|x;\theta)} \\
&= \sum_{\langle x,y \rangle \in \mathcal{D}} R(x, y) \cdot \nabla_\theta \log p(y|x;\theta) \cdot p(y|x;\theta) \\
&= \mathbb{E}_{\mathcal{D}} \left[R(x, y) \cdot \nabla_\theta \log p(y|x;\theta) \right]
\end{aligned}$$

Scaling up: Sequence-level training

- NMT models are trained on the word level with cross-entropy loss but evaluated with *sequence-level* metrics like BLEU, which are non-differentiable
- Model parameters θ can also be refined against any non-differentiable measure $R(x, y)$ using reinforcement learning
- GNMT: improvement in BLEU scores (but not human judgments)

Dataset	Trained with log-likelihood	Refined with RL
En→Fr	38.95	39.92
En→De	24.67	24.60

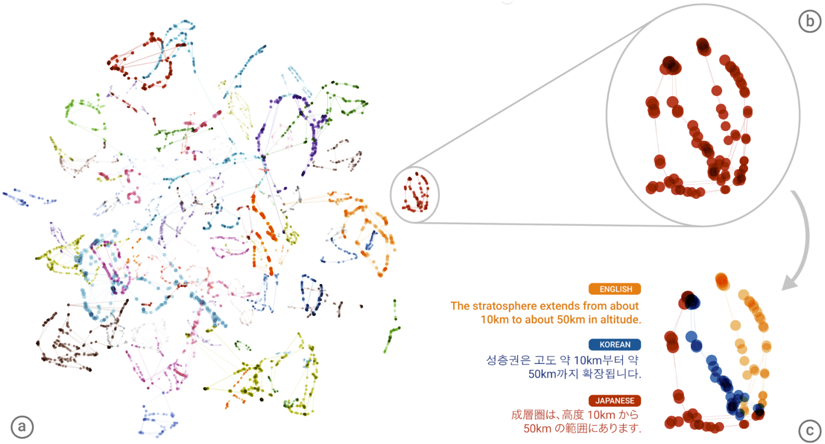
Scaling up: Multilingual MT

- Johnson et al. (2016) proposes a simple change to translate between *multiple* languages with a single NMT model
  - A token is added to the input sequence to indicate the target language for translation
  - Vocabulary and parameters are shared across languages

+ Can improve translation for low-resource languages with little parallel data
+ Enables *zero-shot* translation for language pairs with no parallel data

Scaling up: Multilingual MT

t-SNE projections of learned representations of 74 sentences and different translations in English, Japanese and Korean



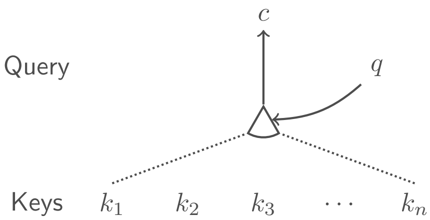
Johnson et al. (2016)

Outline

- Encoder-decoder architectures
  - RNN encoders & decoders
  - Sequence-to-sequence models
  - LSTMs & GRUs
- Attention mechanism
  - Dynamic contexts
  - Induced alignments
- Scaling up
  - Google NMT
  - Sub-word units
  - Sequence-level training
  - Multilingual translation
- Transformers
  - Self-attention
  - Induced structure

Notation: Attention

- Attend over keys $k_1 \ldots k_n$ conditioned on query q



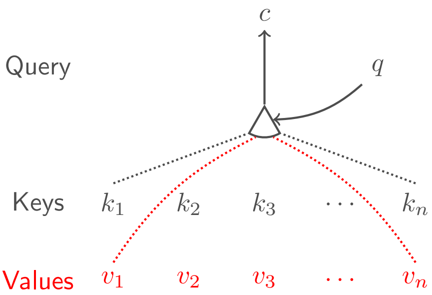
$$c = \sum_i \alpha_i k_i$$

$$\alpha_i = \text{softmax}(e_i)$$

$$e_i = \text{score}(q, k_i)$$

Notation: Attention

- Attend over values $v_1 \dots v_n$ for keys $k_1 \dots k_n$ conditioned on query q



$$c = \sum_i \alpha_i v_i$$

$$\alpha_i = \text{softmax}(e_i)$$

$$e_i = \text{score}(q, k_i)$$

Scaled dot-product attention

- The original additive attention (Bahdanau et al., 2015) is a single-layer feed-forward network over a concatenated query and key.

$$\text{score}(q, k) = u_{\mathbf{qk}}^{\top} \tanh(W_{\mathbf{qk}}^{\top}[q; k])$$

- Scaled dot-product attention (Vaswani et al., 2017) instead uses a simple dot product between the projected query and key (after a linear projection), normalized by the key dimensionality d_k

$$\text{score}(q, k) = \frac{q^{\top} k}{\sqrt{d_k}}$$

where $q = W_{\mathbf{q}}^{\top} q'$ and $k = W_{\mathbf{k}}^{\top} k'$

Note: values are projected separately $v = W_{\mathbf{v}}^{\top} v'$

Transformer

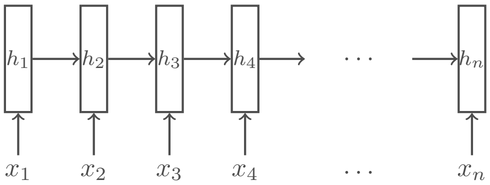
- The sequential computation of RNNs prevents parallelization for inference and also de-emphasizes long-range dependencies

- Vaswani et al., (2017) introduces a sequence model with recurrent connections replaced by *self-attention*
  - Hidden states for each input token are produced by attending to the input sequence using the token as a query
  - Information about word positions must by injected via *position embeddings* in the input

- Recurrent layers are replaced by self-attention layers which can be stacked, each with
  - Scaled dot-product attention
  - Multiple attention heads, projected down to the input dimensionality
  - Unseen tokens masked out (in the decoder)

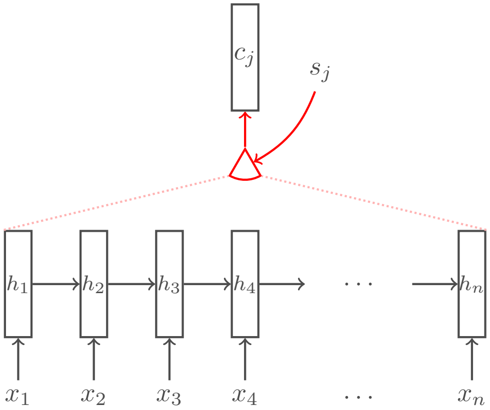
Transformer

RNN encoder



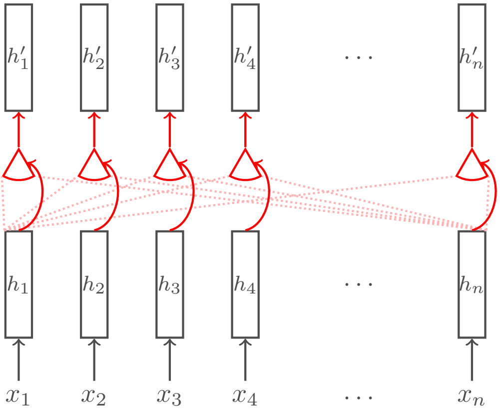
Transformer

RNN encoder with attention



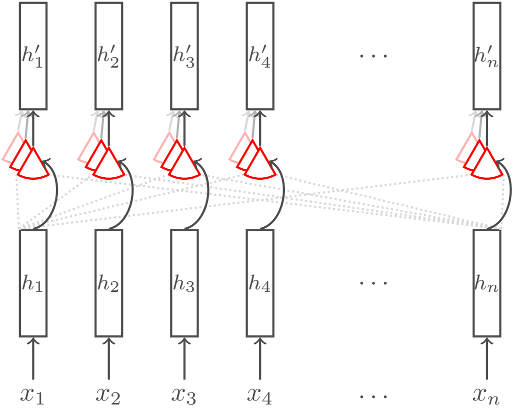
Transformer

Deep encoder with self-attention

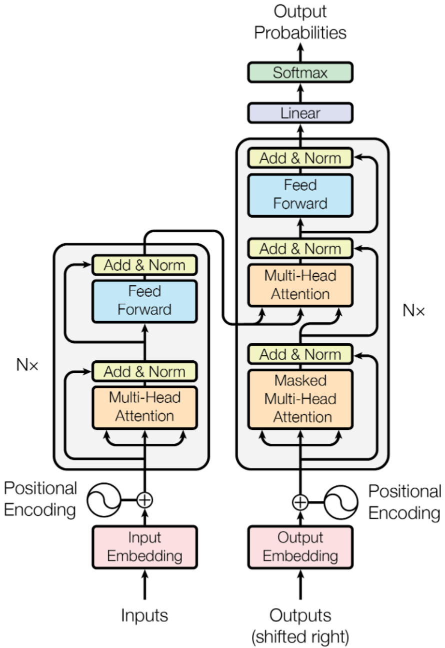


Transformer

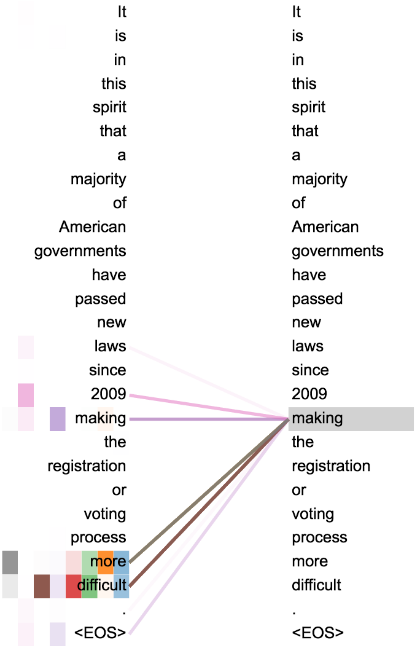
Deep encoder with multi-head self-attention (Vaswani et al., 2017)



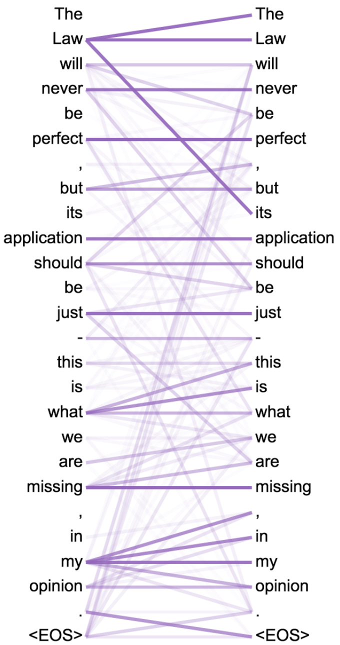
Transformer

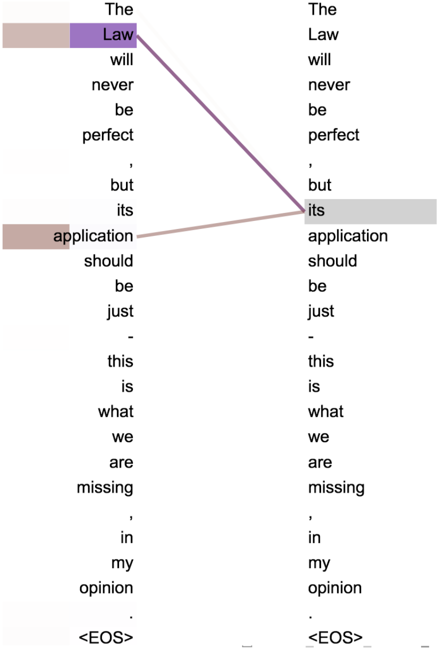


Self-attention: Long-range dependencies

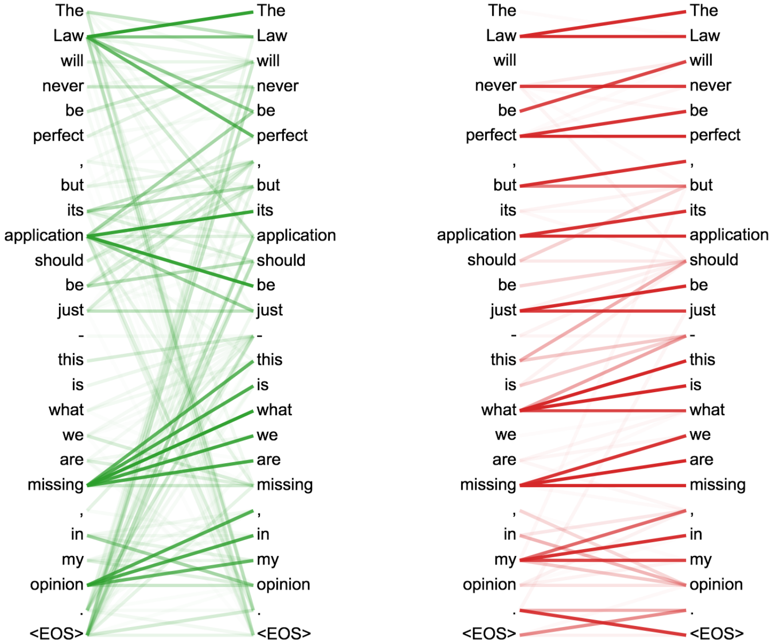


Self-attention: Anaphora resolution





Self-attention: Clause structure



Transformer

+ No recurrence, so inference can be parallelized
+ Improved runtime and performance on translation + other tasks
+ Scaled dot-product attention is efficient
+ Self-attention layers appear to capture some linguistic structure

– $\mathcal{O}(n^2)$ comparisons for each layer (unless restricted)
– Positional embeddings are necessary to account for ordering of input

Resources

- OpenNMT provides implementations of NMT models

	OpenNMT-py	OpenNMT-tf
ConvS2S	✓	
DeepSpeech2	✓	
GPT-2		✓
Im2Text	✓	
Listen, Attend and Spell		✓
RNN with attention	✓	✓
Transformer	✓	✓

+ Available for PyTorch and TensorFlow
+ Actively maintained and used