Neural Machine Translation
COMS W4705: Natural Language Processing

Kapil Thadani
kapil@cs.columbia.edu
Review: Machine Translation

Source

analysis

Target

generation
Review: Machine Translation

Source

interlingua?

generation

Target

lexical

syntax

semantics

pragmatics

analysis
Review: Phrase-based MT

Tomorrow I will fly to the conference in Canada

Morgen fliege Ich nach Kanada zur Konferenz
Review: Phrase-based MT

Tomorrow I will fly to the conference in Canada

Morgen fliege ich nach Kanada zur Konferenz
Review: Phrase-based MT

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

2. Unsupervised phrase-based alignment
   - phrase table \( \pi \)

3. Unsupervised n-gram language modeling
   - language model \( \psi \)

4. Supervised decoder
   - parameters \( \theta \)

\[
\hat{T} = \arg \max_T p(T|S) = \arg \max_T p(S|T, \pi, \theta) \cdot p(T|\psi)
\]
Neural MT

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

2. Unsupervised phrase-based alignment
   - phrase table $\pi$

3. Unsupervised n-gram language modeling
   - language model $\psi$

4. Supervised encoder-decoder framework
   - parameters $\theta$
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

\[
h_t = \phi_h (W_{xh}^T x_t + W_{hh}^T h_{t-1} + b_h)
\]
(typically sigmoid or tanh)
Notation: Basic recurrent unit

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

\[
\begin{align*}
    y_t &= \phi_y(W_{hy}^T h_t + b_y) \\
    h_t &= \phi_h(W_{xh}^T x_t + W_{hh}^T h_{t-1} + b_h) \\
    & \quad \text{(typically sigmoid or tanh)}
\end{align*}
\]
Notation: Softmax

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

Label probabilities

Softmax $y$

Input vector $h$

\[ z = W_{hy}^T h + b_y \]
\[ y_i = \frac{e^{zi}}{\sum_j e^{z_j}} \]
\[ = p(\text{label} = i | h) \]
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 $z$

- $h'_1 \rightarrow h'_2 \rightarrow h'_3 \rightarrow h'_4 \rightarrow \cdots \rightarrow h'_n \rightarrow z$
- $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow \cdots \rightarrow x_n$
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 $z$

*reads* the input text

*classifies*
RNN encoder

Input  words $x_1, \ldots, x_n$

Output  representation $r$
RNN language model

**Input**  words $y_1, \ldots, y_k$

**Output**  following words $y_k, \ldots, y_m$
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 $y_1, \ldots, y_m$

\[ s_i = f(s_{i-1}, y_{i-1}, h_n) \]
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)

\[ h_t = \tanh(W_{xh}^T x_t + W_{hh}^T h_{t-1}) \]  
(normal RNN)
Long short-term memory (LSTM)

\[ \tilde{c}_t = \tanh(W_{xh}^T x_t + W_{hh}^T h_{t-1}) \]

\[ c_t = c_{t-1} + \tilde{c}_t \]
Long short-term memory (LSTM)

\[ \begin{align*}
    f_t &= \sigma(W_{fx}^T x_t + W_{fh}^T h_{t-1}) \\
    \tilde{c}_t &= \tanh(W_{xh}^T x_t + W_{hh}^T h_{t-1}) \\
    c_t &= f_t \odot c_{t-1} + \tilde{c}_t
\end{align*} \]
Long short-term memory (LSTM)

\[
f_t = \sigma(W_{fx}^T x_t + W_{fh}^T h_{t-1})
\]

\[
i_t = \sigma(W_{ix}^T x_t + W_{ih}^T h_{t-1})
\]

\[
\tilde{c}_t = \tanh(W_{xh}^T x_t + W_{hh}^T h_{t-1})
\]

\[
c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t
\]
Long short-term memory (LSTM)

\[
\begin{align*}
f_t &= \sigma(W_{fx}^T x_t + W_{fh}^T h_{t-1}) \\
i_t &= \sigma(W_{ix}^T x_t + W_{ih}^T h_{t-1}) \\
o_t &= \sigma(W_{ox}^T x_t + W_{oh}^T h_{t-1}) \\
\tilde{c}_t &= \tanh(W_{xh}^T x_t + W_{hh}^T h_{t-1}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t)
\end{align*}
\]
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)

\[ \tilde{h}_t = \tanh(W_{xh}^\top x_t + W_{hh}^\top h_{t-1}) \]

\[ h_t = \tilde{h}_t \]
**Gated Recurrent Unit (GRU)**

\[ r_t = \sigma(W_{rx}^T x_t + W_{rh}^T h_{t-1}) \]

\[ \tilde{h}_t = \tanh(W_{xh}^T x_t + W_{hh}^T (r_t \odot h_{t-1})) \]

\[ h_t = \tilde{h}_t \]
Gated Recurrent Unit (GRU)

\[ r_t = \sigma(W_{rx}^T x_t + W_{rh}^T h_{t-1}) \]
\[ z_t = \sigma(W_{zx}^T x_t + W_{zh}^T h_{t-1}) \]
\[ \tilde{h}_t = \tanh(W_{xh}^T x_t + W_{hh}^T (r_t \odot h_{t-1})) \]
\[ h_t = z_t \odot \tilde{h}_t \]
Gated Recurrent Unit (GRU)

\[
\begin{align*}
    r_t &= \sigma(W_{rx}^T x_t + W_{rh}^T h_{t-1}) \\
    z_t &= \sigma(W_{zx}^T x_t + W_{zh}^T h_{t-1}) \\
    \tilde{h}_t &= \tanh(W_{xh}^T x_t + W_{hh}^T (r_t \odot h_{t-1})) \\
    h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{align*}
\]
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

<table>
<thead>
<tr>
<th>Method</th>
<th>test BLEU score (ntst14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline System [29]</td>
<td>33.30</td>
</tr>
<tr>
<td>Single forward LSTM, beam size 12</td>
<td>26.17</td>
</tr>
<tr>
<td>Single reversed LSTM, beam size 12</td>
<td>30.59</td>
</tr>
</tbody>
</table>
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

\[ e_{ij} = a(s_{i-1}, h_j) \]
Attention-based translation

softmax

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

\[ e_{ij} = a(s_{i-1}, h_j) \]
Attention-based translation

\[ y_1, y_2, y_3, y_4 \]

\[ s_1, s_2, s_3, s_4 \]

\[ h_1, h_2, h_3, h_4 \]

\[ x_1, x_2, x_3, x_4 \]

\[ \alpha_{4,1}, \alpha_{4,2}, \alpha_{4,3}, \alpha_{4,4} \]

\[ \alpha_{4,n} \]

\[ c_i = \sum_j \alpha_{ij} h_j \]

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

\[ e_{ij} = a(s_{i-1}, h_j) \]

weighted average
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 $\alpha_{ij}$ reveal alignments between source & target words

Bahdanau et al. (2015)
Induced alignments

Attention weights $\alpha_{ij}$ 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 $\alpha_{ij}$ can be visualized to diagnose errors
+ Performance competitive with phrase-based MT

<table>
<thead>
<tr>
<th>Model</th>
<th>All</th>
<th>No UNK°</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNNencdec-30</td>
<td>13.93</td>
<td>24.19</td>
</tr>
<tr>
<td>RNNsearch-30</td>
<td>21.50</td>
<td>31.44</td>
</tr>
<tr>
<td>RNNencdec-50</td>
<td>17.82</td>
<td>26.71</td>
</tr>
<tr>
<td>RNNsearch-50</td>
<td>26.75</td>
<td>34.16</td>
</tr>
<tr>
<td>RNNsearch-50*</td>
<td>28.45</td>
<td>36.15</td>
</tr>
<tr>
<td>Moses</td>
<td>33.30</td>
<td>35.63</td>
</tr>
</tbody>
</table>

- Runtime for inference is $O(mn)$ instead of $O(m + n)$ 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

<table>
<thead>
<tr>
<th></th>
<th>PBMT</th>
<th>GNMT</th>
<th>Human</th>
<th>Relative Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>English → Spanish</td>
<td>4.885</td>
<td>5.428</td>
<td>5.504</td>
<td>87%</td>
</tr>
<tr>
<td>English → French</td>
<td>4.932</td>
<td>5.295</td>
<td>5.496</td>
<td>64%</td>
</tr>
<tr>
<td>English → Chinese</td>
<td>4.035</td>
<td>4.594</td>
<td>4.987</td>
<td>58%</td>
</tr>
<tr>
<td>Spanish → English</td>
<td>4.872</td>
<td>5.187</td>
<td>5.372</td>
<td>63%</td>
</tr>
<tr>
<td>French → English</td>
<td>5.046</td>
<td>5.343</td>
<td>5.404</td>
<td>83%</td>
</tr>
<tr>
<td>Chinese → English</td>
<td>3.694</td>
<td>4.263</td>
<td>4.636</td>
<td>60%</td>
</tr>
</tbody>
</table>
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

Jet makers feud 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 $\theta$ can also be refined against any non-differentiable measure $R(x, y)$ using reinforcement learning

\[
\nabla_\theta \mathbb{E}_D [R(x, y)] = \sum_{\langle x, y \rangle \in \mathcal{D}} R(x, y) \cdot \nabla_\theta p(y|x; \theta)
\]

\[
= \sum_{\langle x, y \rangle \in \mathcal{D}} R(x, y) \cdot \nabla_\theta p(y|x; \theta) \cdot \frac{p(y|x; \theta)}{p(y|x; \theta)}
\]

\[
= \sum_{\langle x, y \rangle \in \mathcal{D}} R(x, y) \cdot \nabla_\theta \log p(y|x; \theta) \cdot p(y|x; \theta)
\]

\[
= \mathbb{E}_D [R(x, y) \cdot \nabla_\theta \log p(y|x; \theta)]
\]
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 $\theta$ can also be refined against any non-differentiable measure $R(x, y)$ using reinforcement learning.

- GNMT: improvement in BLEU scores (but not human judgments)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Trained with log-likelihood</th>
<th>Refined with RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>En→Fr</td>
<td>38.95</td>
<td>39.92</td>
</tr>
<tr>
<td>En→De</td>
<td>24.67</td>
<td>24.60</td>
</tr>
</tbody>
</table>
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 $v_1 \ldots v_n$ for keys $k_1 \ldots k_n$ 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.

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

- 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 \(q = W_q^\top q'\) and \(k = W_k^\top k'\)

Note: values are projected separately \(v = W_v^\top v'\)
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 be 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

\[ x_1 \rightarrow h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow h_4 \rightarrow \cdots \rightarrow h_n \]

\[ x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow \cdots \rightarrow x_n \]
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

It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult. <EOS>
Self-attention: Anaphora resolution
Self-attention: Clause structure

<table>
<thead>
<tr>
<th>The</th>
<th>The</th>
</tr>
</thead>
<tbody>
<tr>
<td>Law</td>
<td>Law</td>
</tr>
<tr>
<td>will</td>
<td>will</td>
</tr>
<tr>
<td>never</td>
<td>never</td>
</tr>
<tr>
<td>be</td>
<td>be</td>
</tr>
<tr>
<td>perfect</td>
<td>perfect</td>
</tr>
<tr>
<td>,</td>
<td>,</td>
</tr>
<tr>
<td>but</td>
<td>but</td>
</tr>
<tr>
<td>its</td>
<td>its</td>
</tr>
<tr>
<td>application</td>
<td>application</td>
</tr>
<tr>
<td>should</td>
<td>should</td>
</tr>
<tr>
<td>be</td>
<td>be</td>
</tr>
<tr>
<td>just</td>
<td>just</td>
</tr>
<tr>
<td>this</td>
<td>this</td>
</tr>
<tr>
<td>is</td>
<td>is</td>
</tr>
<tr>
<td>what</td>
<td>what</td>
</tr>
<tr>
<td>we</td>
<td>we</td>
</tr>
<tr>
<td>are</td>
<td>are</td>
</tr>
<tr>
<td>missing</td>
<td>missing</td>
</tr>
<tr>
<td>,</td>
<td>,</td>
</tr>
<tr>
<td>in</td>
<td>in</td>
</tr>
<tr>
<td>my</td>
<td>my</td>
</tr>
<tr>
<td>opinion</td>
<td>opinion</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The</th>
<th>The</th>
</tr>
</thead>
<tbody>
<tr>
<td>Law</td>
<td>Law</td>
</tr>
<tr>
<td>will</td>
<td>will</td>
</tr>
<tr>
<td>never</td>
<td>never</td>
</tr>
<tr>
<td>be</td>
<td>be</td>
</tr>
<tr>
<td>perfect</td>
<td>perfect</td>
</tr>
<tr>
<td>,</td>
<td>,</td>
</tr>
<tr>
<td>but</td>
<td>but</td>
</tr>
<tr>
<td>its</td>
<td>its</td>
</tr>
<tr>
<td>application</td>
<td>application</td>
</tr>
<tr>
<td>should</td>
<td>should</td>
</tr>
<tr>
<td>be</td>
<td>be</td>
</tr>
<tr>
<td>just</td>
<td>just</td>
</tr>
<tr>
<td>this</td>
<td>this</td>
</tr>
<tr>
<td>is</td>
<td>is</td>
</tr>
<tr>
<td>what</td>
<td>what</td>
</tr>
<tr>
<td>we</td>
<td>we</td>
</tr>
<tr>
<td>are</td>
<td>are</td>
</tr>
<tr>
<td>missing</td>
<td>missing</td>
</tr>
<tr>
<td>,</td>
<td>,</td>
</tr>
<tr>
<td>in</td>
<td>in</td>
</tr>
<tr>
<td>my</td>
<td>my</td>
</tr>
<tr>
<td>opinion</td>
<td>opinion</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

<EOS>  <EOS>
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

− $O(n^2)$ comparisons for each layer (unless restricted)
− Positional embeddings are necessary to account for ordering of input
Resources

- **OpenNMT** provides implementations of NMT models

<table>
<thead>
<tr>
<th></th>
<th>OpenNMT-py</th>
<th>OpenNMT-tf</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvS2S</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>DeepSpeech2</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>GPT-2</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Im2Text</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Listen, Attend and Spell</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>RNN with attention</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Transformer</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

- Available for **PyTorch** and **TensorFlow**
- Actively maintained and used