Translation with Pervasive Attention

Maha Elbayad$^{1,2}$  Laurent Besacier$^1$  Jakob Verbeek$^2$

$^1$ Université Grenoble Alpes  $^2$ INRIA Grenoble, France

Conference on Computational Natural Language Learning, 2018
Machine translation

- Given pairs of aligned sentences \((x, y)\) (source, target)
- Model the conditional distribution \(\log p(y|x)\)

\[
f = (\text{La, croissance, économique, s'est, ralentie, ces, dernières, années, .})
\]

\[
e = (\text{Economic, growth, has, slowed, down, in, recent, years, .})
\]

source: Kyunghyun Cho, NYU.

- Translation model: \(p(y|x)\)
- Language model: \(p(y)\)
Neural machine translation

- RNN encoder-decoder models
  [Kalchbrenner and Blunsom, 2013, Cho et al., 2014, Sutskever et al., 2014]

\[
p(y_1:T | x_1:L) = \prod_{t=1}^{T} p(y_t | y_{<t}, C(x_1:L)) \tag{1}
\]
Recurrent neural encoder

1. One-hot encoding: (sub)words tokens
2. Vector representation $s_t = Wx_t$, $W \in \mathbb{R}^{d \times V}$
3. Recursion: $h_t = f_\theta(h_{t-1}, s_t)$
4. Code: $C(x_1:L)$
Recurrent neural decoder

1. Recursion: \( z_{t+1} = f_\theta(z_t, y_t, C(x_{1:L})) \)
2. Emission prob.: \( p(y_t|z_t) = \text{SoftMax}(Ez_t) \)
3. Generation: sampling, greedy, beam search

\( f = (\text{La, croissance, économique, s'est, ralentie, ces, dernières, années, .}) \)
\( e = (\text{Economic, growth, has, slowed, down, in, recent, years, .}) \)

Source: Kyunghyun Cho, NYU.
Performance vs. sentence length
“You can’t cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!”

Ray Mooney @ ACL Workshop on Semantic Parsing, 2014
“You can’t cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!”

Ray Mooney @ ACL Workshop on Semantic Parsing, 2014

▶ Ok, so how about cramming it into two vectors?!
“You can’t cram the meaning of a whole *%&!*$ing sentence into a single *$&!*ing vector!”

Ok, so how about cramming it into two vectors?!

Bi-directional RNN encoder [Schuster and Paliwal, 1997]
Attention [Bahdanau et al., 2015]

- Re-encode input given current decoder state $z_t$
- Use re-encoded input in decoder state update

$$z_{t+1} = f_\theta (z_t, y_t, C(x_{1:L}), A(x_{1:L}, y_{1:t})) \quad (2)$$
Attention [Bahdanau et al., 2015]

- Re-encode input given current decoder state $z_t$
- Use re-encoded input in decoder state update

\[ z_{t+1} = f_\theta (z_t, y_t, C(x_1:L), A(x_1:L, y_1:t)) \] (2)

\[ f = (\text{La, croissance, économique, s'est, ralentie, ces, dernières, années, .}) \]

\[ e = (\text{Economic, growth, has, slowed, down, in, recent, years, .}) \]

source: Kyunghyun Cho, NYU.
So far, so good...
So far, so good...

- Elements of state-of-the-art machine translation
So far, so good...

- Elements of state-of-the-art machine translation
  1. Bi-directional RNN encoder
So far, so good...

- Elements of state-of-the-art machine translation
  1. Bi-directional RNN encoder
  2. RNN decoder with beam-search
So far, so good...

- Elements of state-of-the-art machine translation
  1. Bi-directional RNN encoder
  2. RNN decoder with beam-search
  3. Attention mechanism
So far, so good...

- Elements of state-of-the-art machine translation
  1. Bi-directional RNN encoder
  2. RNN decoder with beam-search
  3. Attention mechanism

Now let’s try something else...

- No encoder
So far, so good...

- Elements of state-of-the-art machine translation

1. Bi-directional RNN encoder
2. RNN decoder with beam-search
3. Attention mechanism

Now let’s try something else...

- No encoder
- No decoder
So far, so good...

- Elements of state-of-the-art machine translation
  1. Bi-directional RNN encoder
  2. RNN decoder with beam-search
  3. Attention mechanism

Now let’s try something else...

- No encoder
- No decoder
- No attention (?)
Trading depth for parallelism

- **RNN** receptive field unlimited with depth 1
- **CNN** receptive field grows by 1 each layer

In NLP, eg. [Collobert and Weston, 2008, Kalchbrenner et al., 2014, Gehring et al., 2017]
Trading depth for parallelism

- **RNN** receptive field unlimited with depth 1
- **CNN** receptive field grows by 1 each layer

In NLP, eg. [Collobert and Weston, 2008, Kalchbrenner et al., 2014, Gehring et al., 2017]
Stop cramming a sentence into a vector...

- Joint coding: input N-grams given last M output tokens
Stop cramming a sentence into a vector...

- Joint coding: input N-grams given last M output tokens

<table>
<thead>
<tr>
<th>Source sequence</th>
<th>Alice</th>
<th>dit</th>
<th>a</th>
<th>Bob</th>
<th>que</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target sequence</td>
<td>&lt;start&gt;</td>
<td>Alice</td>
<td>told</td>
<td>Bob</td>
<td>that</td>
<td>Charlie</td>
</tr>
</tbody>
</table>

```
Alice
dit
` a
Bob
que
Charlie
<start>
Alice
told
Bob
that
Charlie
told
```
Stop cramming a sentence into a vector...

- Joint coding: input N-grams given last M output tokens
  - Receptive field: \((N,M) = 1 + (2,1) \times \text{depth}\)

### Target sequence

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>told</th>
<th>Bob</th>
<th>that</th>
<th>Charlie</th>
<th>told</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;start&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>que</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>told</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Source sequence

- Alice
- a
- dit
- a
- Bob
- que
- Charlie
Stop cramming a sentence into a vector...

- Joint coding: input $N$-grams given last $M$ output tokens
  - Receptive field: $(N,M) = 1 + (2,1) \times \text{depth}$
- Activation map: $a_{i,t} = A(x_{1:L}, y_{1:t})$

<table>
<thead>
<tr>
<th>Source sequence</th>
<th>Target sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charlie</td>
<td>Alice</td>
</tr>
<tr>
<td>que</td>
<td>told</td>
</tr>
<tr>
<td>Bob</td>
<td>that</td>
</tr>
<tr>
<td>a</td>
<td>told</td>
</tr>
<tr>
<td>dit</td>
<td>Bob</td>
</tr>
<tr>
<td>à</td>
<td>Alice</td>
</tr>
<tr>
<td>&lt;start&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Target sequence

$
\begin{array}{cccccc}
\text{Alice} & \text{told} & \text{Bob} & \text{that} & \text{Charlie} & \text{told} \\
\text{a} & \text{dit} & \text{à} & \text{Bob} & \text{que} & \text{Charlie} \\
\text{Alice} & \text{told} & \text{Bob} & \text{that} & \text{Charlie} & \text{told} \\
\end{array}
$
## Pervasive attention

- Similar to “classic” attention: re-coding input given output
- Present in every layer, rather than an “afterthought”

### Target sequence

<table>
<thead>
<tr>
<th>&lt;start&gt;</th>
<th>Alice</th>
<th>told</th>
<th>Bob</th>
<th>that</th>
<th>Charlie</th>
<th>told</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>à</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>que</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Network architecture

- Input tensor $X_{i,j} = [v_i, w_j]$ concatenates word embeddings
- 2D masked CNN layers, e.g. DenseNet [Huang et al., 2017]
- Soft-max to predict next token at every target position

<table>
<thead>
<tr>
<th>Target sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;start&gt;</td>
</tr>
<tr>
<td>Alice</td>
</tr>
<tr>
<td>a</td>
</tr>
<tr>
<td>dit</td>
</tr>
<tr>
<td>à</td>
</tr>
<tr>
<td>Bob</td>
</tr>
<tr>
<td>que</td>
</tr>
<tr>
<td>Charlie</td>
</tr>
</tbody>
</table>

Source sequence
Collapsing source dimension

- Max-pool over variable-length source dimension
- Generates one vector per target position

$$M_j = [\max_i X_{ij}^1, \ldots \max_i X_{ij}^D] \quad (3)$$
Implicit sentence alignments from max-pooling

- Look at where the max-pool is picking up its values

\[ M_j = [\max_i X_{ij}^1, \ldots \max_i X_{ij}^D] \]  \hspace{1cm} (4)
Experiments: IWSLT’14

- Translation of TED and TEDx talks
- 160k German-to-English train pairs
Experiments: IWSLT’14

- Translation of TED and TEDx talks
- 160k German-to-English train pairs

<table>
<thead>
<tr>
<th>Words</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet</td>
<td>26.81</td>
</tr>
<tr>
<td>ResNet, Gated</td>
<td>28.11</td>
</tr>
<tr>
<td>DenseNet</td>
<td>29.39</td>
</tr>
<tr>
<td>DenseNet, Gated</td>
<td>29.35</td>
</tr>
<tr>
<td>DenseNet</td>
<td>27.43</td>
</tr>
<tr>
<td>DenseNet</td>
<td>28.58</td>
</tr>
<tr>
<td>DenseNet</td>
<td>29.21</td>
</tr>
<tr>
<td>DenseNet</td>
<td>29.56</td>
</tr>
</tbody>
</table>

**BPE**

<table>
<thead>
<tr>
<th>Words</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>DenseNet</td>
<td>30.24</td>
</tr>
<tr>
<td>DenseNet, Max &amp; Attention</td>
<td>30.56</td>
</tr>
</tbody>
</table>
## Experiments: IWSLT’14

<table>
<thead>
<tr>
<th>BPE</th>
<th>BLEU</th>
<th>FLOPS</th>
<th># prms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-LSTM enc-dec [Bahdanau et al., 2015]</td>
<td>31.02 ± 0.05</td>
<td>1.79</td>
<td>6M</td>
</tr>
<tr>
<td>Transformer* [Vaswani et al., 2017]</td>
<td>29.58 ± 0.04</td>
<td>1.47</td>
<td>12M</td>
</tr>
<tr>
<td>ConvS2S** (MLE) [Gehring et al., 2017]</td>
<td>31.56 ± 0.05</td>
<td>1.73</td>
<td>19M</td>
</tr>
<tr>
<td>ConvS2S (MLE+SLE) [Edunov et al., 2018]</td>
<td>32.84 ± 0.08</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>2DConvMP (ours, g=16, L=12)</td>
<td>31.08 ± 0.06</td>
<td>1.25</td>
<td>3M</td>
</tr>
<tr>
<td>2DConvMP (ours, g=32, L=20)</td>
<td>33.41 ± 0.05</td>
<td>2.20</td>
<td>6M</td>
</tr>
</tbody>
</table>

* Obtained using FairSeq.
** Obtained using author’s code = FairSeq.
Conclusion

- Joint-coding approach, alternative to encoder-decoder
  - 2D CNN with masked filters
  - Source-target interactions pervasive in architecture
  - Proper coding of source at every target token
- Max-pooling generates implicit sentence alignment
- Compares favorably to encoder-decoder models
  - Both in nr. of parameters and compute

Ongoing work:
- More efficient hybrid 1D-2D architectures
- Architectures for multiple language pairs
Conclusion

- Joint-coding approach, alternative to encoder-decoder
  - 2D CNN with masked filters
  - Source-target interactions pervasive in architecture
  - Proper coding of source at every target token
- Max-pooling generates implicit sentence alignment
- Compares favorably to encoder-decoder models
  - Both in nr. of parameters and compute

- Ongoing work:
  - More efficient hybrid 1D-2D architectures
  - Architectures for multiple language pairs
Thanks for your attention
Neural machine translation by jointly learning to align and translate.
In ICLR.

Learning phrase representations using RNN encoder-decoder for statistical machine translation.
In EMNLP.

In ICML.

Classical structured prediction losses for sequence to sequence learning.
In NAACL.

Convolutional sequence to sequence learning.
In ICML.

Densely connected convolutional networks.
In CVPR.

Recurrent continuous translation models.
In ACL.

A convolutional neural network for modelling sentences.
In ACL.
Bidirectional recurrent neural networks.

Sequence to sequence learning with neural networks.
In *NIPS*.

Attention is all you need.
In *NIPS*. 