Machine Translation

Elisabetta Cavallo, Ben Dodge

COS598C - Deep Learning for Natural Language Processing
February 25th, 2020
Sequence to Sequence Learning with Neural Networks

Sutskever I., Vinyals O., V. Le Q. (2014)
Sequence to Sequence Learning with Neural Networks

Goal: I am a student → Je suis étudiant

With:

Ilya Sutskever
Google
ilyas@gmail.com

Oriol Vinyals
Google
vinyals@gmail.com

Quoc V. Le
Google
qvl@gmail.com
Background

- Statistical Machine Translation (SMT)
- RNN to rescore baseline translations
- Encode and decode a fixed-size vector
  - Using CNN: Kalchbrenner and Blunsom (2013)
  - Integrating into SMT: Cho et al. (2014)
  - Using Attention: Bahdanau et al. (2014)
The Basic Model

4 layers
1000 cells
1000d embeddings
380M parameters
Extra Bits

- Reverse order of source words
- Separate encoder/decoder parameters
- Beam search decoding
- Ensemble of models
## Empirical Results

### Neural Network Only (except SMT baseline)

<table>
<thead>
<tr>
<th>Method</th>
<th>test BLEU score (ntst14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bahdanau et al. [2]</td>
<td>28.45</td>
</tr>
<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>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 1</td>
<td>33.00</td>
</tr>
<tr>
<td>Ensemble of 2 reversed LSTMs, beam size 12</td>
<td>33.27</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 2</td>
<td>34.50</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 12</td>
<td>34.81</td>
</tr>
</tbody>
</table>

### Neural Network Rescoring SMT

<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>Cho et al. [5]</td>
<td>34.54</td>
</tr>
<tr>
<td>State of the art [9]</td>
<td>37.0</td>
</tr>
<tr>
<td>Rescoring the baseline 1000-best with a single forward LSTM</td>
<td>35.61</td>
</tr>
<tr>
<td>Rescoring the baseline 1000-best with a single reversed LSTM</td>
<td>35.85</td>
</tr>
<tr>
<td>Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs</td>
<td>36.5</td>
</tr>
<tr>
<td>Oracle Rescoring of the Baseline 1000-best lists</td>
<td>~45</td>
</tr>
</tbody>
</table>
Qualitative Results

Model learns the meaning of sentences, even with complex reordering
Machine Translation Datasets!

• Workshop on Statistical Machine Translation
  – WMT ‘14 English to French (36M sentence pairs)
  – Mostly from the Europarl corpus
  – Also has En↔De, En↔Hi, En↔Cs, En↔Ru, ...

• International Workshop on Spoken Language Translation (IWSLT)

• “Google-internal production datasets” (Wu et al., 2016)
Non-Autoregressive Neural Machine Translation

Gu J., Bradbury J., Xiong C., O.K. Li V., Socher R. (2018)
Non-Autoregressive Neural Machine Translation (Gu et Al., 2018)

**Autoregressive models**

**Idea:** sequential model which decodes the output depending on the input words the words previously generated (i.e. translated).

- In formulas, given a source sentence $X = \{x_1, ..., x_T\}$ (e.g. I am a student) and an output sentence $Y = \{y_1, ..., y_T\}$ (e.g. je suis étudiant):

$$p_{\text{AR}}(Y|X; \theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T}; \theta)$$

**Pros:** usually fluent as it corresponds to the word-by-word nature of human language production, easy to train, state-of-the-art performance on large-scale corpora, but...
Non-Autoregressive Neural Machine Translation (Gu et al., 2018)

Autoregressive models

\[ \text{Cons: unidirectional conditioning on previously translated words, beam search suffers from diminishing returns with respect to beam size, NOT parallelizable at inference, etc.} \]

\[ p_{\text{NA}}(Y|X; \theta) = p_L(T|x_{1:T'}; \theta) \cdot \prod_{t=1}^{T} p(y_t|x_{1:T'}; \theta) \]

\[ \text{Assumption: target sequence length } T \text{ can be modelled with a conditional distribution } p_L \]

\[ \text{Probability of a token conditionally independent of previous tokens} \]

Naive solution: why not generating output words independent of previously translated words?

Multimodality problem: complete conditional independence is a poor approximation to the true target distribution (i.e. some translation could be equally likely while not both correct).

\[ \rightarrow \text{e.g. } P(\text{Danke schon} | \text{Thank you}) = P(\text{Vielen Dank} | \text{Thank you}) = \ldots P(\text{Danke dank} | \text{Thank you}) \text{ which isn't German} \ldots \]
Non-Autoregressive Neural Machine Translation (Gu et Al., 2018)

State of Art AR Neural Model before Gu et Al.

Neural Machine Translation with Transformers and Self-Attention by Vaswani et al., 2017

Transformer

Self-attention

\[ \text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \]

where:
- \( Q \) = query (vector representation of one word in the sequence)
- \( K \) = keys (matrix representations of all the words in the sequence)
- \( V \) = values (matrix representations of all the words in the sequence)
- \( d_k \) = dimension of queries and keys
Non-Autoregressive Neural Machine Translation (Gu et Al., 2018)

State of Art AR Neural Model before Gu et Al.

Non-Autoregressive Neural Machine Translation (Gu et Al., 2018)

Non-Autoregressive Transformer (NAT) by Gu et Al.

Idea (Gu et Al., 2018): Non-autoregressive translation model based on a Transformer network (Vaswani et al., 2017), with modified encoder to predict fertilities.

→ can produce translations of an entire sentence at a time in a fully parallel way.

Figure 2: The architecture of the NAT, where the black solid arrows represent differentiable connections and the purple dashed arrows are non-differentiable operations. Each sublayer inside the encoder and decoder stacks also includes layer normalization and a residual connection.
Non-Autoregressive Neural Machine Translation (Gu et Al., 2018)

**Encoder Stack**

**Structure:** Multi-Head Self-Attention modules + Feedforward NN (MLP)  
(same as in Vaswani et al., 2017)

→ no RNN = no inherent requirement for sequential execution.
Non-Autoregressive Neural Machine Translation (Gu et Al., 2018)

Encoder Stack → Fertility predictor

The encoder + fertility predictor have two jobs:

1) understanding and interpreting the input sentence,

2) predicting a sequence of numbers (e.g. [1, 1, 2, 0, 1]) called fertilities that are used as input to the parallel decoder.
**What are fertilities?**

**Fertilities** are latent variables which represent how many output words each input word generates.

*E.g. “Please” has fertility 4 for French translation (“S’il te plait”)*

Fertility $p_F(f_t' | x_{1:T})$ is modelled at each position independently using a one-layer neural network with a softmax classifier on top of the output of the last encoder layer.

→ fertility values are a property of each input word while depending on information and context from the entire sentence.

→ but... what about reordering of words?

$$p_{NA}(Y|X; \theta) = \sum_{f_1, \ldots, f_{T'} \in \mathcal{F}} \left( \prod_{t'=1}^{T'} p_F(f_t' | x_{1:T'}; \theta) \cdot \prod_{t=1}^{T} p(y_t | x_1 \{f_1\}, \ldots, x_{T'} \{f_{T'}\}; \theta) \right)$$

(token for $x_i$ repeated $f_i$ times)
Non-Autoregressive Neural Machine Translation (Gu et al., 2018)

Encoder Stack → Fertility predictor → Decoder Stack

Structure: Copied source inputs from the encoder side using *fertilities* + Self-attention layers and MLP (Non-causal + Positional + Inter-Attention)
Non-Autoregressive Neural Machine Translation (Gu et Al., 2018)

Encoder Stack → Fertility predictor → Decoder Stack: in Action!

Animation from:
https://blog.einstein.ai/fully-parallel-text-generation-for-neural-machine-translation/?fbclid=IwAR3VX1ZCn3ArjBnAcWmKxvELOknMvth9Hfd-rzEa0ovYmi_OFCrTJ3R5AQ
Fertility Training

• Why can’t we train the fertilities end-to-end?
  – Cannot flow gradients
  – Need separate supervision

• Loss function

\[
\mathcal{L}_{\text{ML}} = \log p_{\mathcal{NA}}(Y|X; \theta) = \log \sum_{f_{1:T}, \in \mathcal{F}} p_F(f_{1:T'}|x_{1:T'}; \theta) \cdot p(y_{1:T}|x_{1:T'}, f_{1:T'}; \theta)
\]

\[
\geq \mathbb{E}_{f_{1:T'}, \sim q} \left( \sum_{t=1}^{T} \log p(y_t|x_1{f_1}, \ldots, x_{T'}{f_{T'}}; \theta) + \sum_{t'=1}^{T'} \log p_F(f_t'|x_{1:T'}; \theta) \right) + \mathcal{H}(q)
\]

Translation Loss

Fertility Loss
IBM Model 2

- Idea is to use alignments from SMT model
  - Easy to translate into fertilities

\[
p(a|e, m) = \prod_{j=1}^{m} q(a_j|j, l, m)\]

\[
p(f, a|e, m) = \prod_{j=1}^{m} q(a_j|j, l, m) t(f_j|e_{a_j})\]

- Trained with expectation maximization (EM) on data
  - Allows model to learn alignments that are not observed in the data

\( l = 6, m = 7 \)
\( e = \text{And the program has been implemented} \)
\( f = \text{Le programme a été mis en application} \)

\{2, 3, 4, 5, 6, 6, 6\}
Is this a good latent variable?

• Their criteria
  – Easy to infer from training data
  – Should account for correlations across time (so each output is almost conditionally independent)
  – Should not convey too much information about target translation so that decoder still has something to learn

"Including both fertilities and reordering in the latent variable would provide complete alignment statistics. This would make the decoding function trivially easy to approximate given the latent variable and force all of the modeling complexity into the encoder. Using fertilities alone allows the decoder to take some of this burden off of the encoder."
Non-Autoregressive Neural Machine Translation (Gu et Al., 2018)

Knowledge Distillation

**Idea** (Hinton et al., 2015, Kim and Rush, 2016): distilling the knowledge of a teacher model (e.g. larger model or ensemble of models) into a student model.

**Upside:** can train on less data while still learning how to generalise well, much faster during inference, outputs are less noisy. **Downside:** output of the teacher model are lower in quality than original data.
Fine Tuning

Problem: our training is not end-to-end (translation model and fertilities predictor are trained separately).
→ after training the Non-AR Transformer to convergence: add a fine-tuning step

\[
\mathcal{L}_{\text{FT}} = \lambda \left( \mathbb{E}_{f_1:T' \sim p_F} \left( \mathcal{L}_{\text{RKL}} (f_1:T') - \mathcal{L}_{\text{RKL}} (\tilde{f}_1:T') \right) + \mathbb{E}_{f_1:T' \sim q} \left( \mathcal{L}_{\text{RKL}} (f_1:T') \right) \right) + (1 - \lambda) \mathcal{L}_{\text{KD}}
\]

- \( \mathcal{L}_{\text{RKL}} \) is obtained from word-knowledge distillation based on KL divergence with the teacher model:

\[
\mathcal{L}_{\text{RKL}} (f_1:T'; \theta) = \sum_{t=1}^{T} \sum_{y_t} \log p_{\text{AR}} (y_t|\hat{y}_{1:t-1}, x_{1:T'}) \cdot p_{\text{NA}} (y_t|x_{1:T'}, f_{1:T'}; \theta)
\]

→ trained with Reinforcement Learning

→ trained with Backprop
Decoding Process

- **Argmax decoding**

  \[ \hat{Y}_{\text{argmax}} = G(x_{1:T'}, \hat{f}_{1:T'}; \theta), \text{ where } \hat{f}_{t'} = \arg\max_f p_F(f_{t'}|x_{1:T'}; \theta) \]

- **Average decoding**

  \[ \hat{Y}_{\text{average}} = G(x_{1:T'}, \hat{f}_{1:T'}; \theta), \text{ where } \hat{f}_{t'} = \text{Round} \left( \sum_{f_{t'}=1}^L p_F(f_{t'}|x_{1:T'}; \theta) f_{t'} \right) \]

- **Noisy parallel decoding (NPD)**

  \[ \hat{Y}_{\text{NPD}} = G(x_{1:T'}, \arg\max_{f_{t'} \sim p_F} p_{AR}(G(x_{1:T'}, f_{1:T'}; \theta)|X; \theta); \theta) \]
Noisy Parallel Decoding

- Non-autoregressive model can leverage autoregressive teacher during inference as well

- “Autoregressive” teacher can run very fast while evaluating candidate translation
  - Does not have to consume previous input
  - Operates off of candidate translation tokens (like teacher forcing during training)

- Sample size trades off speed and accuracy
  - We will see this in a moment
Example of NPD

Decoder input (copied by fertilities)

se lucreaza la solutii de genul acesta.
se la solutii de genul acesta.
se lucreaza la solutii de acesta.
se se lucreaza la solutii de acesta.
se lucreaza lucreaza la solutii de acesta.
se se lucreaza lucreaza la solutii de acesta.
se se lucreaza lucreaza la solutii de de acesta.
se se lucreaza lucreaza la solutii de genul acesta.

Decoder output

solutions on this kind are done.
work done on solutions like this.
solutions on this kind is done.
work is done on solutions like this.
work is done on solutions like this.
work is being done on solutions like this.
work is being done on solutions such as this.
work is being done on solutions such this kind.

AR favorite
Comparison of Decoding Methods

- So is NAT-NPD better than autoregressive?
- Could this model ever even theoretically outperform autoregressive models?
## Results

Bottom line: sometimes very competitive BLEU with significant speedup

<table>
<thead>
<tr>
<th>Models</th>
<th>WMT14</th>
<th>WMT16</th>
<th>IWSLT16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>En→De</td>
<td>De→En</td>
<td>En→Ro</td>
</tr>
<tr>
<td>NAT</td>
<td>17.35</td>
<td>20.62</td>
<td>26.22</td>
</tr>
<tr>
<td>NAT (+FT)</td>
<td>17.69</td>
<td>21.47</td>
<td>27.29</td>
</tr>
<tr>
<td>NAT (+FT + NPD s = 10)</td>
<td>18.66</td>
<td>22.41</td>
<td>29.02</td>
</tr>
<tr>
<td>NAT (+FT + NPD s = 100)</td>
<td>19.17</td>
<td>23.20</td>
<td>29.79</td>
</tr>
<tr>
<td>Autoregressive (b = 1)</td>
<td>22.71</td>
<td>26.39</td>
<td>31.35</td>
</tr>
<tr>
<td>Autoregressive (b = 4)</td>
<td>23.45</td>
<td>27.02</td>
<td>31.91</td>
</tr>
</tbody>
</table>
## Ablation Study

<table>
<thead>
<tr>
<th>Distillation</th>
<th>Decoder Inputs</th>
<th>Fine-tuning</th>
<th>BLEU</th>
<th>BLEU (T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b=1$</td>
<td>+uniform</td>
<td>+$L_{KD}$</td>
<td>≈ 2</td>
<td>16.51</td>
</tr>
<tr>
<td></td>
<td>+fertility</td>
<td>+$L_{BP}$</td>
<td>18.87</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+PosAtt</td>
<td>+$L_{RL}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b=4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+uniform</td>
<td>+$L_{KD}$</td>
<td>20.72</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+fertility</td>
<td>+$L_{BP}$</td>
<td>21.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+PosAtt</td>
<td>+$L_{RL}$</td>
<td>24.02</td>
<td>43.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>25.20</td>
<td>45.41</td>
</tr>
<tr>
<td></td>
<td>+uniform</td>
<td>+$L_{KD}$</td>
<td>22.44</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+fertility</td>
<td>+$L_{BP}$</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>+PosAtt</td>
<td>+$L_{RL}$</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>25.76</td>
<td>46.11</td>
</tr>
<tr>
<td></td>
<td>+uniform</td>
<td>+$L_{KD}$</td>
<td>26.52</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+fertility</td>
<td>+$L_{BP}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+PosAtt</td>
<td>+$L_{RL}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Really need to copy source*

*Teacher distillation is very helpful*

*External fertility actually contributes a lot*

*Fine-tuning process gives another percent*

*guess what this means?*
Non-Autoregressive Neural Machine Translation (Gu et Al., 2018)

Quiz time

Can you think of any limitations of the proposed approach in (Gu et al, 2018)?
Non-Autoregressive Neural Machine Translation (Gu et Al., 2018)

Quiz time

Can you think of any limitations of the proposed approach in (Gu et al, 2018)?

- Non-differentiable component when copying fertilities into the decoder → model cannot be trained-end-to-end (and fine-tuning leads to only negligible improvements).

- Still relies on autoregressive to train the teacher model → does not beat AR “on its own”.

- Very slow at training → it has to train both the teacher (larger model = computationally expensive) and the student

- Fertilities do no cope with the problem of re-ordering of words in the translated sentence → they are just an alignment between number of words in the input and number of words in the output.
Bonus Paper!

Bonus Paper

Mask-Predict: Parallel Decoding of Conditional Masked Language Models

Marjan Ghazvininejad* Omer Levy* Yinhan Liu* Luke Zettlemoyer
Facebook AI Research
Seattle, WA

Pros: Newer Better Prettier
Cutting to the chase

- Train to predict **masked** target tokens given source sequence and unmasked target tokens.
- Encoder also predicts **length** of target sequence based on source.

\[
\begin{array}{ll}
src & \text{Je suis étudiant} \\
[M] [M] [M] [M] & \text{predicted } L=4 \\
\end{array}
\]

\[
\begin{array}{ll}
t = 0 & \text{I am studying} \\
\end{array}
\]

\[
\begin{array}{ll}
t = 1 & \text{I am am student} \\
\end{array}
\]

\[
\begin{array}{ll}
t = 2 & \text{I am a student} \\
\end{array}
\]

generated all masked tokens

replaced least certain tokens

arrived at final translation

---

\[
\begin{array}{ll}
src & \text{Der Abzug der französischen Kampftruppen wurde am 20. November abgeschlossen} \\
\end{array}
\]

\[
\begin{array}{ll}
t = 0 & \text{The departure of the French combat completed completed on 20 November} \\
\end{array}
\]

\[
\begin{array}{ll}
t = 1 & \text{The departure of French combat troops was completed on 20 November} \\
\end{array}
\]

\[
\begin{array}{ll}
t = 2 & \text{The withdrawal of French combat troops was completed on November 20th} \\
\end{array}
\]
Conclusions

- Outperforms other parallel decoding schemes
- Linear-ish trade-off between speed and performance
- Still heavily reliant on knowledge distillation
Thank you

Any questions?