Practical Neural Machine Translation

Rico Sennrich, Barry Haddow

Institute for Language, Cognition and Computation
University of Edinburgh

April 4, 2017
(Last Updated: April 28, 2017)
Introduction

2 Neural Networks — Basics

3 Language Models using Neural Networks

4 Attention-based NMT Model

5 Edinburgh’s WMT16 System

6 Analysis: Why does NMT work so well?

7 Building and Improving NMT Systems

8 Resources, Further Reading and Wrap-Up
1987  Early encoder-decoder, with vocabulary size 30-40 [Allen, 1987]

2013  Pure neural MT system presented [Kalchbrenner and Blunsom, 2013]

2014  Competitive encoder-decoder for large-scale MT
       [Bahdanau et al., 2015, Luong et al., 2014]

2015  NMT systems in shared tasks – perform well in WMT, state-of-the-art at IWSLT

2016  NMT systems top most language pairs in WMT

2016  Commercial deployments of NMT launched
NMT now state-of-the-art

### WMT16 EN→DE

<table>
<thead>
<tr>
<th>system</th>
<th>BLEU</th>
<th>official rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>uedin-nmt</td>
<td>34.2</td>
<td>1</td>
</tr>
<tr>
<td>metamind</td>
<td>32.3</td>
<td>2</td>
</tr>
<tr>
<td>uedin-syntax</td>
<td>30.6</td>
<td>3</td>
</tr>
<tr>
<td>NYU-UMontreal</td>
<td>30.8</td>
<td>4</td>
</tr>
<tr>
<td>online-B</td>
<td>29.4</td>
<td>5-10</td>
</tr>
<tr>
<td>KIT/LIMSI</td>
<td>29.1</td>
<td>5-10</td>
</tr>
<tr>
<td>cambridge</td>
<td>30.6</td>
<td>5-10</td>
</tr>
<tr>
<td>online-A</td>
<td>29.9</td>
<td>5-10</td>
</tr>
<tr>
<td>promt-rule</td>
<td>23.4</td>
<td>5-10</td>
</tr>
<tr>
<td>KIT</td>
<td>29.0</td>
<td>6-10</td>
</tr>
<tr>
<td>jhu-syntax</td>
<td>26.6</td>
<td>11-12</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>28.3</td>
<td>11-12</td>
</tr>
<tr>
<td>uedin-pbmt</td>
<td>28.4</td>
<td>13-14</td>
</tr>
<tr>
<td>online-F</td>
<td>19.3</td>
<td>13-15</td>
</tr>
<tr>
<td>online-G</td>
<td>23.8</td>
<td>14-15</td>
</tr>
</tbody>
</table>

### WMT16 DE→EN

<table>
<thead>
<tr>
<th>system</th>
<th>BLEU</th>
<th>official rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>uedin-nmt</td>
<td>38.6</td>
<td>1</td>
</tr>
<tr>
<td>online-B</td>
<td>35.0</td>
<td>2-5</td>
</tr>
<tr>
<td>online-A</td>
<td>32.8</td>
<td>2-5</td>
</tr>
<tr>
<td>uedin-syntax</td>
<td>34.4</td>
<td>2-5</td>
</tr>
<tr>
<td>KIT</td>
<td>33.9</td>
<td>2-6</td>
</tr>
<tr>
<td>uedin-pbmt</td>
<td>35.1</td>
<td>5-7</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>34.5</td>
<td>6-7</td>
</tr>
<tr>
<td>online-G</td>
<td>30.1</td>
<td>8</td>
</tr>
<tr>
<td>jhu-syntax</td>
<td>31.0</td>
<td>9</td>
</tr>
<tr>
<td>online-F</td>
<td>20.2</td>
<td>10</td>
</tr>
</tbody>
</table>

WMT16 DE→EN
NMT now state-of-the-art

<table>
<thead>
<tr>
<th>system</th>
<th>BLEU</th>
<th>official rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>uedin-nmt</td>
<td>34.2</td>
<td>1</td>
</tr>
<tr>
<td>metamind</td>
<td>32.3</td>
<td>2</td>
</tr>
<tr>
<td>uedin-syntax</td>
<td>30.6</td>
<td>3</td>
</tr>
<tr>
<td>NYU-UMontreal</td>
<td>30.8</td>
<td>4</td>
</tr>
<tr>
<td>online-B</td>
<td>29.4</td>
<td>5-10</td>
</tr>
<tr>
<td>KIT/LIMSI</td>
<td>29.1</td>
<td>5-10</td>
</tr>
<tr>
<td>cambridge</td>
<td>30.6</td>
<td>5-10</td>
</tr>
<tr>
<td>online-A</td>
<td>29.9</td>
<td>5-10</td>
</tr>
<tr>
<td>prompt-rule</td>
<td>23.4</td>
<td>5-10</td>
</tr>
<tr>
<td>KIT</td>
<td>29.0</td>
<td>6-10</td>
</tr>
<tr>
<td>jhu-syntax</td>
<td>26.6</td>
<td>11-12</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>28.3</td>
<td>11-12</td>
</tr>
<tr>
<td>uedin-pbmt</td>
<td>28.4</td>
<td>13-14</td>
</tr>
<tr>
<td>online-F</td>
<td>19.3</td>
<td>13-15</td>
</tr>
<tr>
<td>online-G</td>
<td>23.8</td>
<td>14-15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>system</th>
<th>BLEU</th>
<th>official rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>uedin-nmt</td>
<td>38.6</td>
<td>1</td>
</tr>
<tr>
<td>online-B</td>
<td>35.0</td>
<td>2-5</td>
</tr>
<tr>
<td>online-A</td>
<td>32.8</td>
<td>2-5</td>
</tr>
<tr>
<td>uedin-syntax</td>
<td>34.4</td>
<td>2-5</td>
</tr>
<tr>
<td>KIT</td>
<td>33.9</td>
<td>2-6</td>
</tr>
<tr>
<td>uedin-pbmt</td>
<td>35.1</td>
<td>5-7</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>34.5</td>
<td>6-7</td>
</tr>
<tr>
<td>online-G</td>
<td>30.1</td>
<td>8</td>
</tr>
<tr>
<td>jhu-syntax</td>
<td>31.0</td>
<td>9</td>
</tr>
<tr>
<td>online-F</td>
<td>20.2</td>
<td>10</td>
</tr>
</tbody>
</table>

WMT16 EN→DE

- pure NMT

WMT16 DE→EN
NMT now state-of-the-art

<table>
<thead>
<tr>
<th>system</th>
<th>BLEU</th>
<th>official rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>uedin-nmt</td>
<td>34.2</td>
<td>1</td>
</tr>
<tr>
<td>metamind</td>
<td>32.3</td>
<td>2</td>
</tr>
<tr>
<td>uedin-syntax</td>
<td>30.6</td>
<td>3</td>
</tr>
<tr>
<td>NYU-UMontreal</td>
<td>30.8</td>
<td>4</td>
</tr>
<tr>
<td>online-B</td>
<td>29.4</td>
<td>5-10</td>
</tr>
<tr>
<td>KIT/LIMSI</td>
<td>29.1</td>
<td>5-10</td>
</tr>
<tr>
<td>cambridge</td>
<td>30.6</td>
<td>5-10</td>
</tr>
<tr>
<td>online-A</td>
<td>29.9</td>
<td>5-10</td>
</tr>
<tr>
<td>promt-rule</td>
<td>23.4</td>
<td>5-10</td>
</tr>
<tr>
<td>KIT</td>
<td>29.0</td>
<td>6-10</td>
</tr>
<tr>
<td>jhu-syntax</td>
<td>26.6</td>
<td>11-12</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>28.3</td>
<td>11-12</td>
</tr>
<tr>
<td>uedin-pbmt</td>
<td>28.4</td>
<td>13-14</td>
</tr>
<tr>
<td>online-F</td>
<td>19.3</td>
<td>13-15</td>
</tr>
<tr>
<td>online-G</td>
<td>23.8</td>
<td>14-15</td>
</tr>
</tbody>
</table>

WMT16 EN→DE

<table>
<thead>
<tr>
<th>system</th>
<th>BLEU</th>
<th>official rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>uedin-nmt</td>
<td>38.6</td>
<td>1</td>
</tr>
<tr>
<td>online-B</td>
<td>35.0</td>
<td>2-5</td>
</tr>
<tr>
<td>online-A</td>
<td>32.8</td>
<td>2-5</td>
</tr>
<tr>
<td>uedin-syntax</td>
<td>34.4</td>
<td>2-5</td>
</tr>
<tr>
<td>KIT</td>
<td>33.9</td>
<td>2-6</td>
</tr>
<tr>
<td>uedin-pbmt</td>
<td>35.1</td>
<td>5-7</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>34.5</td>
<td>6-7</td>
</tr>
<tr>
<td>online-G</td>
<td>30.1</td>
<td>8</td>
</tr>
<tr>
<td>jhu-syntax</td>
<td>31.0</td>
<td>9</td>
</tr>
<tr>
<td>online-F</td>
<td>20.2</td>
<td>10</td>
</tr>
</tbody>
</table>

WMT16 DE→EN

- pure NMT
- NMT component
Sennrich, Haddow

Practical Neural Machine Translation

3 / 109

NMT now state-of-the-art

<table>
<thead>
<tr>
<th>System</th>
<th>Score</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>uedin-nmt</td>
<td>25.8</td>
<td>1</td>
</tr>
<tr>
<td>NYU-UMontreal</td>
<td>23.6</td>
<td>2</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>23.6</td>
<td>3</td>
</tr>
<tr>
<td>cu-chimera</td>
<td>21.0</td>
<td>4-5</td>
</tr>
<tr>
<td>cu-tamchyna</td>
<td>20.8</td>
<td>4-5</td>
</tr>
<tr>
<td>uedin-cu-syntax</td>
<td>20.9</td>
<td>6-7</td>
</tr>
<tr>
<td>online-B</td>
<td>22.7</td>
<td>6-7</td>
</tr>
<tr>
<td>online-A</td>
<td>19.5</td>
<td>15</td>
</tr>
<tr>
<td>cu-TectoMT</td>
<td>14.7</td>
<td>16</td>
</tr>
<tr>
<td>cu-mergedtrees</td>
<td>8.2</td>
<td>18</td>
</tr>
</tbody>
</table>

WMT16 EN→CS

<table>
<thead>
<tr>
<th>System</th>
<th>Score</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>online-B</td>
<td>39.2</td>
<td>1-2</td>
</tr>
<tr>
<td>uedin-nmt</td>
<td>33.9</td>
<td>1-2</td>
</tr>
<tr>
<td>uedin-pbmt</td>
<td>35.2</td>
<td>3</td>
</tr>
<tr>
<td>uedin-syntax</td>
<td>33.6</td>
<td>4-5</td>
</tr>
<tr>
<td>online-A</td>
<td>30.8</td>
<td>4-6</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>32.2</td>
<td>5-7</td>
</tr>
<tr>
<td>LIMSI</td>
<td>31.0</td>
<td>6-7</td>
</tr>
</tbody>
</table>

WMT16 CS→EN

<table>
<thead>
<tr>
<th>System</th>
<th>Score</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>uedin-nmt</td>
<td>31.4</td>
<td>1</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>30.4</td>
<td>2</td>
</tr>
<tr>
<td>online-B</td>
<td>28.6</td>
<td>3</td>
</tr>
<tr>
<td>PJATK</td>
<td>28.3</td>
<td>8-10</td>
</tr>
<tr>
<td>online-A</td>
<td>25.7</td>
<td>11</td>
</tr>
<tr>
<td>cu-mergedtrees</td>
<td>13.3</td>
<td>12</td>
</tr>
</tbody>
</table>

WMT16 RO→EN

<table>
<thead>
<tr>
<th>System</th>
<th>Score</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>QT21-HimL-SysComb</td>
<td>28.9</td>
<td>1-2</td>
</tr>
<tr>
<td>KIT</td>
<td>25.8</td>
<td>3-7</td>
</tr>
<tr>
<td>uedin-pbmt</td>
<td>26.8</td>
<td>3-7</td>
</tr>
<tr>
<td>online-B</td>
<td>25.4</td>
<td>3-7</td>
</tr>
<tr>
<td>uedin-lmu-hiero</td>
<td>25.9</td>
<td>3-7</td>
</tr>
<tr>
<td>RWTH-SYSCOMB</td>
<td>27.1</td>
<td>3-7</td>
</tr>
<tr>
<td>LIMSI</td>
<td>23.9</td>
<td>8-10</td>
</tr>
<tr>
<td>lmu-cuni</td>
<td>24.3</td>
<td>8-10</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>23.5</td>
<td>8-11</td>
</tr>
<tr>
<td>usfd-rescoring</td>
<td>23.1</td>
<td>10-12</td>
</tr>
<tr>
<td>online-A</td>
<td>19.2</td>
<td>11-12</td>
</tr>
</tbody>
</table>

WMT16 EN→RO
## NMT now state-of-the-art

<table>
<thead>
<tr>
<th>System</th>
<th>Score</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROMT-rule</td>
<td>22.3</td>
<td>1</td>
</tr>
<tr>
<td>amu-uedin</td>
<td>25.3</td>
<td>2-4</td>
</tr>
<tr>
<td>online-B</td>
<td>23.8</td>
<td>2-5</td>
</tr>
<tr>
<td>uedin-nmt</td>
<td>26.0</td>
<td>2-5</td>
</tr>
<tr>
<td>online-G</td>
<td>26.2</td>
<td>3-5</td>
</tr>
<tr>
<td>NYU-UMontreal</td>
<td>23.1</td>
<td>6</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>24.0</td>
<td>7-8</td>
</tr>
<tr>
<td>LIMSI</td>
<td>23.6</td>
<td>7-10</td>
</tr>
<tr>
<td>online-A</td>
<td>20.2</td>
<td>8-10</td>
</tr>
<tr>
<td>AFRL-MITLL-phr</td>
<td>23.5</td>
<td>9-10</td>
</tr>
<tr>
<td>AFRL-MITLL-verb</td>
<td>20.9</td>
<td>11</td>
</tr>
<tr>
<td>online-F</td>
<td>8.6</td>
<td>12</td>
</tr>
</tbody>
</table>

### WMT16 EN→RU

<table>
<thead>
<tr>
<th>System</th>
<th>Score</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>amu-uedin</td>
<td>29.1</td>
<td>1-2</td>
</tr>
<tr>
<td>online-G</td>
<td>28.7</td>
<td>1-3</td>
</tr>
<tr>
<td>NRC</td>
<td>29.1</td>
<td>2-4</td>
</tr>
<tr>
<td>online-B</td>
<td>28.1</td>
<td>3-5</td>
</tr>
<tr>
<td>uedin-nmt</td>
<td>28.0</td>
<td>4-5</td>
</tr>
<tr>
<td>online-A</td>
<td>25.7</td>
<td>6-7</td>
</tr>
<tr>
<td>AFRL-MITLL-phr</td>
<td>27.6</td>
<td>6-7</td>
</tr>
<tr>
<td>AFRL-MITLL-contrast</td>
<td>27.0</td>
<td>8-9</td>
</tr>
<tr>
<td>PROMT-rule</td>
<td>20.4</td>
<td>8-9</td>
</tr>
<tr>
<td>online-F</td>
<td>13.5</td>
<td>10</td>
</tr>
</tbody>
</table>

### WMT16 FI→EN

<table>
<thead>
<tr>
<th>System</th>
<th>Score</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>online-G</td>
<td>15.4</td>
<td>1-3</td>
</tr>
<tr>
<td>abumatra-nmt</td>
<td>17.2</td>
<td>1-4</td>
</tr>
<tr>
<td>online-B</td>
<td>14.4</td>
<td>1-4</td>
</tr>
<tr>
<td>abumatran-combo</td>
<td>17.4</td>
<td>3-5</td>
</tr>
<tr>
<td>UH-opus</td>
<td>16.3</td>
<td>4-5</td>
</tr>
<tr>
<td>NYU-UMontreal</td>
<td>15.1</td>
<td>6-8</td>
</tr>
<tr>
<td>abumatran-pbsmt</td>
<td>14.6</td>
<td>6-8</td>
</tr>
<tr>
<td>online-A</td>
<td>13.0</td>
<td>6-8</td>
</tr>
<tr>
<td>jhu-pbmt</td>
<td>13.8</td>
<td>9-10</td>
</tr>
<tr>
<td>UH-factored</td>
<td>12.8</td>
<td>9-12</td>
</tr>
<tr>
<td>aalto</td>
<td>11.6</td>
<td>10-12</td>
</tr>
<tr>
<td>jhu-hltcoe</td>
<td>11.9</td>
<td>10-13</td>
</tr>
<tr>
<td>UUT</td>
<td>11.6</td>
<td>11-13</td>
</tr>
</tbody>
</table>

### WMT16 RU→EN

### WMT16 EN→FI
SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market.
SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

Google announces Neural Machine Translation to improve Google Translate
SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market.

Google announces Neural Machine Translation to improve Google Translate.

WIPO goes Neural

Oct 4, 2016 590 views 31 Likes 3 Comments
SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market.

Google announces Neural Machine Translation to improve Google Translate.

WIPO goes Neural

Microsoft Translator launching Neural Network based translations for all its speech languages.
SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market.

Baidu Says They Were First in NMT: Language Industry News Roundup

Google announces Neural Machine Translation to improve Google Translate

WIPO goes Neural

Microsoft Translator launching Neural Network based translations for all its speech languages
SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

Baidu Says They Were First in NMT: Language Industry News Roundup
by Marion Marking on January 16, 2017

Google announces Neural Machine Translation to improve Google Translate

WIPO goes Neural
Oct 4, 2016  590 views  41 Likes  3 Comments

Microsoft Translator launching Neural Network based translations for all its speech languages

Rate this article ★★★★★

Microsoft Translator  November 15, 2016

TraMOOC
Translation for Massive Open Online Courses
SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market.

Google announces Neural Machine Translation to improve Google Translate.

WIPO goes Neural

October 4, 2016 590 views  41 likes  3 comments

Microsoft Translator launching Neural Network based translations for all its speech languages.

Rate this article 5 stars
SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market.

Google announces Neural Machine Translation to improve Google Translate.

WIPO goes Neural
Oct 4, 2016  |  590 views  |  41 Likes  |  3 Comments

Microsoft Translator launching Neural Network based translations for all its speech languages.

Baidu Says They Were First in NMT: Language Industry News Roundup
by Marion Marking on January 16, 2017

Andrew Ng, Stanford Adjunct Professor, Coursera co-Founder, and Chief Scientist at Chinese search giant Baidu, recently made a bold claim that may raise a few eyebrows in the NMT community.
Course Goals

At the end of this tutorial, you will

- have a basic theoretical understanding of models/algorithms in NMT
- understand strengths and weaknesses of NMT
- know techniques that help to build state-of-the-art NMT systems
- know practical tips for various problems you may encounter:
  - training and decoding efficiency
  - domain adaptation
  - ways to further improve translation quality
  - ...

no hands-on coding/training in tutorial, but helpful resources are provided
What is a Neural Network?

- A complex non-linear function which:
  - is built from simpler units (neurons, nodes, gates, ...)
  - maps vectors/matrices to vectors/matrices
  - is parameterised by vectors/matrices

Why is this useful?
- very expressive
- can represent (e.g.) parameterised probability distributions
- evaluation and parameter estimation can be built up from components
What is a Neural Network?

- A complex non-linear function which:
  - is built from simpler units (neurons, nodes, gates, ...)
  - maps vectors/matrices to vectors/matrices
  - is parameterised by vectors/matrices

- Why is this useful?
  - very expressive
  - can represent (e.g.) parameterised probability distributions
  - evaluation and parameter estimation can be built up from components
A Simple Neural Network Classifier

- $x$ is a vector input, $y$ is a scalar output
- $w$ and $b$ are the parameters ($b$ is a bias term)
- $g$ is a non-linear activation function

$$g(w \cdot x + b)$$
Functions like XOR cannot be separated by a linear function

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(neurons arranged in layers, and fire if input is $\geq 1$)
Why Non-linearity?

Functions like XOR cannot be separated by a linear function

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₂</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(neurons arranged in layers, and fire if input is ≥ 1)
Functions like XOR cannot be separated by a linear function.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Why Non-linearity?

Functions like XOR cannot be separated by a linear function.

XOR Truth table

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(neurons arranged in layers, and fire if input is $\geq 1$)
Activation functions

- desirable:
  - differentiable (for gradient-based training)
  - monotonic (for better training stability)
  - non-linear (for better expressivity)
More Complex Architectures

Convolutional

- Fully connected layer
- K-Max pooling (k=3)
- Folding
- Wide convolution (m=2)
- Dynamic k-max pooling (k=f(s)=5)
- Wide convolution (m=3)
- Projected sentence matrix (s=7)

Recurrent

- Target chars: “e” “i” “l” “o”
- Output layer
  - “e”:
    - 1.0
    - 2.2
    - -3.0
    - 4.1
  - “i”:
    - 0.5
    - 0.3
    - -1.0
    - 1.2
  - “l”:
    - 0.1
    - 0.5
    - 1.9
    - -1.1
  - “o”:
    - 0.2
    - -1.5
    - -0.1
    - 2.2
- Hidden layer
  - “e”:
    - 0.3
    - -0.1
    - 0.9
  - “i”:
    - 1.0
    - 0.3
    - 0.1
  - “l”:
    - 0.1
    - -0.5
    - -0.3
  - “o”:
    - 0.2
    - -1.5
    - -0.1
    - 2.2
- Input layer
  - “h”:
    - 1
    - 0
    - 0
  - “e”:
    - 0
    - 1
    - 0
  - “l”:
    - 0
    - 0
    - 1
  - “o”:
    - 0
    - 0
    - 1

Andrej Karpathy

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[Kalchbrenner et al., 2014]
Training of Neural Networks

- Parameter estimation
  - Use gradient descent
  - Requires labelled training data . . .
    . . . and differentiable objective function

- Network structure enables efficient computation
  - Forward pass to compute network output
  - Backpropogation, i.e. backward pass using chain rule, to calculate gradient

- Normally train stochastically using mini-batches
Practical Considerations

- hyperparameters:
  - number and size of layers
  - minibatch size
  - learning rate
  - ...
- initialisation of weight matrices
- stopping criterion
- regularization (dropout)
- bias units (always-on input)
<table>
<thead>
<tr>
<th>Toolkits for Neural Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>What does a Toolkit Provide</td>
</tr>
<tr>
<td>- Multi-dimensional matrices (tensors)</td>
</tr>
<tr>
<td>- Automatic differentiation</td>
</tr>
<tr>
<td>- Efficient GPU routines for tensor operations</td>
</tr>
</tbody>
</table>

- Torch: [http://torch.ch/](http://torch.ch/)
- TensorFlow: [https://www.tensorflow.org/](https://www.tensorflow.org/)
- Theano: [http://deeplearning.net/software/theano/](http://deeplearning.net/software/theano/)

There are many more!
1 Introduction
2 Neural Networks — Basics
3 Language Models using Neural Networks
4 Attention-based NMT Model
5 Edinburgh’s WMT16 System
6 Analysis: Why does NMT work so well?
7 Building and Improving NMT Systems
8 Resources, Further Reading and Wrap-Up
a sentence $T$ of length $n$ is a sequence $w_1, \ldots, w_n$

$$p(T) = p(w_1, \ldots, w_n)$$

$$= \prod_{i=1}^{n} p(w_i | w_0, \ldots, w_{i-1}) \quad \text{(chain rule)}$$

$$\approx \prod_{i=1}^{n} p(w_i | w_{i-k}, \ldots, w_{i-1}) \quad \text{(Markov assumption: n-gram model)}$$
n-gram NNLM [Bengio et al., 2003]

- **Input:** context of n-1 previous words
- **Output:** probability distribution for next word
- Linear **embedding layer** with shared weights
- One or several **hidden layers**
Representing words as vectors

**One-hot encoding**

- **example vocabulary:** 'man', 'runs', 'the', '.'
- **input/output for** \(p(\text{runs}|\text{the man})\):

\[
\begin{align*}
x_0 &= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} & x_1 &= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} & y_{\text{true}} &= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}
\end{align*}
\]

- **size of input/output vector:** vocabulary size
- **embedding layer is lower-dimensional and dense**
  - smaller weight matrices
  - network learns to group similar words to similar point in vector space
Softmax activation function

\[
p(y = j | x) = \frac{e^{x_j}}{\sum_k e^{x_k}}
\]

- softmax function normalizes output vector to probability distribution → computational cost linear to vocabulary size (!)
- ideally: probability 1 for correct word; 0 for rest
- SGD with softmax output minimizes cross-entropy (and hence perplexity) of neural network
Feedforward neural language model: math

\[ h_1 = \varphi W_1(E x_1, E x_2) \]
\[ y = \text{softmax}(W_2 h_1) \]

[Vaswani et al., 2013]
Feedforward neural language model in SMT

**FFNLM**

- can be integrated as a feature in the log-linear SMT model [Schwenk et al., 2006]
- costly due to matrix multiplications and softmax solutions:
  - n-best reranking
  - variants of softmax (hierarchical softmax, self-normalization [NCE])
  - shallow networks; premultiplication of hidden layer
- scales well to many input words
  → models with source context [Devlin et al., 2014]

\[
\begin{align*}
\text{S:} & \quad \text{我} \quad \text{就} \quad \text{取} \quad \text{钱} \quad \text{给} \quad \text{了} \quad \text{她们} \\
& \quad \text{i} \quad \text{will} \quad \text{get} \quad \text{money} \quad \text{to} \quad \text{perf.} \quad \text{them} \\
\text{T:} & \quad \text{will} \quad \text{get} \quad \text{the} \quad \text{money} \quad \text{to} \quad \text{them} \\
P(\text{the} \mid \text{get, will, i, 就, 取, 钱, 给, 了})
\end{align*}
\]
Recurrent neural network language model (RNNLM)

RNNLM [Mikolov et al., 2010]

- motivation: condition on arbitrarily long context
  → no Markov assumption
- we read in one word at a time, and update hidden state incrementally
- hidden state is initialized as empty vector at time step 0
- parameters:
  - embedding matrix $E$
  - feedforward matrices $W_1, W_2$
  - recurrent matrix $U$

\[
\begin{align*}
  h_i &= \begin{cases} 
    0, & \text{if } i = 0 \\
    \tanh(W_1 x_i + U h_{i-1}), & \text{if } i > 0 
  \end{cases} \\
  y_i &= \text{softmax}(W_2 h_{i-1})
\end{align*}
\]
RNN variants

Gated units

- alternative to plain RNN
- sigmoid layers $\sigma$ act as “gates” that control flow of information
- allows passing of information over long time
  $\rightarrow$ avoids vanishing gradient problem
- strong empirical results
- popular variants:
  - Long Short Term Memory (LSTM) (shown)
  - Gated Recurrent Unit (GRU)
RNN variants

gated units

- alternative to plain RNN
- sigmoid layers $\sigma$ act as “gates” that control flow of information
- allows passing of information over long time
  → avoids vanishing gradient problem
- strong empirical results
- popular variants:
  - Long Short Term Memory (LSTM) (shown)
  - Gated Recurrent Unit (GRU)
1 Introduction
2 Neural Networks — Basics
3 Language Models using Neural Networks
4 Attention-based NMT Model
5 Edinburgh’s WMT16 System
6 Analysis: Why does NMT work so well?
7 Building and Improving NMT Systems
8 Resources, Further Reading and Wrap-Up
Suppose that we have:
- a source sentence $S$ of length $m \ (x_1, \ldots, x_m)$
- a target sentence $T$ of length $n \ (y_1, \ldots, y_n)$

We can express translation as a probabilistic model

\[
T^* = \arg \max_T p(T|S)
\]

Expanding using the chain rule gives

\[
p(T|S) = p(y_1, \ldots, y_n|x_1, \ldots, x_m)
\]

\[
= \prod_{i=1}^{n} p(y_i|y_1, \ldots, y_{i-1}, x_1, \ldots, x_m)
\]
Differences Between Translation and Language Model

- **Target-side language model:**
  \[
  p(T) = \prod_{i=1}^{n} p(y_i | y_1, \ldots, y_{i-1})
  \]

- **Translation model:**
  \[
  p(T|S) = \prod_{i=1}^{n} p(y_i | y_1, \ldots, y_{i-1}, x_1, \ldots, x_m)
  \]

- We could just treat sentence pair as one long sequence, but:
  - We do not care about \( p(S) \)
  - We may want different vocabulary, network architecture for source text
Differences Between Translation and Language Model

- Target-side language model:

\[ p(T) = \prod_{i=1}^{n} p(y_i | y_1, \ldots, y_{i-1}) \]

- Translation model:

\[ p(T|S) = \prod_{i=1}^{n} p(y_i | y_1, \ldots, y_{i-1}, x_1, \ldots, x_m) \]

- We could just treat sentence pair as one long sequence, but:
  - We do not care about \( p(S) \)
  - We may want different vocabulary, network architecture for source text

→ Use separate RNNs for source and target.
of course john has fun
Encoder-Decoder for Translation

Decoder

\[ y_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5 \rightarrow y_5 \]

Encoder

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

Text:

of course john has fun

natürlich hat john spaß
• Last encoder hidden-state “summarises” source sentence

• With multilingual training, we can potentially learn language-independent meaning representation
Summary vector as information bottleneck

Problem: Sentence Length

- Fixed sized representation degrades as sentence length increases
- Reversing source brings some improvement [Sutskever et al., 2014]

![Graph showing BLEU score vs. sentence length]

[Cho et al., 2014]
Summary vector as information bottleneck

Problem: Sentence Length
- Fixed sized representation degrades as sentence length increases
- Reversing source brings some improvement [Sutskever et al., 2014]

Solution: Attention
- Compute *context vector* as weighted average of source hidden states
- Weights computed by feed-forward network with softmax activation

[Cho et al., 2014]
Encoder-Decoder with Attention

Decoder

\[ y_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5 \]

Encoder

\[ h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow h_4 \]

\[ x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \]

\[ 0.7 + 0.1 + 0.1 = 1.0 \]

\[ \text{natürlich} \rightarrow \text{hat} \rightarrow \text{john} \rightarrow \text{spaß} \]

\[ 0.7 + 0.1 + 0.1 = 1.0 \]
Encoder-Decoder with Attention

Decoder

\[ y_1 \rightarrow s_1 \rightarrow y_2 \rightarrow s_2 \rightarrow y_3 \rightarrow s_3 \rightarrow y_4 \rightarrow s_4 \rightarrow y_5 \rightarrow s_5 \]

Encoder

\[ x_1 \rightarrow h_1 \rightarrow x_2 \rightarrow h_2 \rightarrow x_3 \rightarrow h_3 \rightarrow x_4 \rightarrow h_4 \]

\begin{align*}
&\text{natürlich} & \text{hat} & \text{john} & \text{spaß} \\
&0.6 & 0.2 & 0.1 & 0.1
\end{align*}
Encoder-Decoder with Attention

Decoder

\[
\begin{align*}
&y_1 \\
&s_1
\end{align*}
\]

\[
\begin{align*}
&y_2 \\
&s_2
\end{align*}
\]

\[
\begin{align*}
&y_3 \\
&s_3
\end{align*}
\]

\[
\begin{align*}
&y_4 \\
&s_4
\end{align*}
\]

\[
\begin{align*}
&y_5 \\
&s_5
\end{align*}
\]

Encoder

\[
\begin{align*}
&h_1 \\
&x_1
\end{align*}
\]

\[
\begin{align*}
&h_2 \\
&x_2
\end{align*}
\]

\[
\begin{align*}
&h_3 \\
&x_3
\end{align*}
\]

\[
\begin{align*}
&h_4 \\
&x_4
\end{align*}
\]

\[
0.1 + 0.7 + 0.1 + 0.1 = 1.0
\]

\[
\begin{align*}
&\text{natürlich} \\
&\text{hat} \\
&\text{john} \\
&\text{spaß}
\end{align*}
\]

Sennrich, Haddow
Practical Neural Machine Translation
Encoder-Decoder with Attention

Decoder

\[
\begin{align*}
    y_1 &\rightarrow s_1 \\
    y_2 &\rightarrow s_2 \\
    y_3 &\rightarrow s_3 \\
    y_4 &\rightarrow s_4 \\
    y_5 &\rightarrow s_5
\end{align*}
\]

Encoder

\[
\begin{align*}
    h_1 &\rightarrow h_2 \\
    h_2 &\rightarrow h_3 \\
    h_3 &\rightarrow h_4
\end{align*}
\]

\[
\begin{align*}
    x_1 &\rightarrow + \\
    x_2 &\rightarrow + \\
    x_3 &\rightarrow + \\
    x_4 &\rightarrow +
\end{align*}
\]

\[
\begin{align*}
    0.1 &\rightarrow + \\
    0.7 &\rightarrow + \\
    0.1 &\rightarrow + \\
    0.1 &\rightarrow +
\end{align*}
\]

\[
\begin{align*}
    \text{natürlich} &\rightarrow h_1 \\
    \text{hat} &\rightarrow h_2 \\
    \text{john} &\rightarrow h_3 \\
    \text{spaß} &\rightarrow h_4
\end{align*}
\]

\[
\begin{align*}
    \text{of course} &\rightarrow y_1 \\
    \text{john} &\rightarrow y_3 \\
    \text{has} &\rightarrow y_4 \\
    \text{fun} &\rightarrow y_5
\end{align*}
\]
Encoder-Decoder with Attention

Decoder

Decoder

Encoder

Decoder

Sennrich, Haddow
Practical Neural Machine Translation 30 / 109
Attentional encoder-decoder: Maths

**simplifications of model by [Bahdanau et al., 2015] (for illustration)**

- plain RNN instead of GRU
- simpler output layer
- we do not show bias terms
- decoder follows *Look, Update, Generate* strategy [Sennrich et al., 2017]
- Details in [https://github.com/amunmt/amunmt/blob/master/contrib/notebooks/dl4mt.ipynb](https://github.com/amunmt/amunmt/blob/master/contrib/notebooks/dl4mt.ipynb)

**notation**

- $W$, $U$, $E$, $C$, $V$ are weight matrices (of different dimensionality)
  - $E$ one-hot to embedding (e.g. $50000 \cdot 512$)
  - $W$ embedding to hidden (e.g. $512 \cdot 1024$)
  - $U$ hidden to hidden (e.g. $1024 \cdot 1024$)
  - $C$ context (2x hidden) to hidden (e.g. $2048 \cdot 1024$)
  - $V_o$ hidden to one-hot (e.g. $1024 \cdot 50000$)

- separate weight matrices for encoder and decoder (e.g. $E_x$ and $E_y$)
- input $X$ of length $T_x$; output $Y$ of length $T_y$
Attentional encoder-decoder: Maths

encoder

\[
\begin{align*}
\overrightarrow{h}_j &= \begin{cases} 
0, & \text{if } j = 0 \\
\tanh(\overrightarrow{W}_x E_x x_j + \overrightarrow{U}_x h_{j-1}), & \text{if } j > 0
\end{cases}, \\
\overleftarrow{h}_j &= \begin{cases} 
0, & \text{if } j = T_x + 1 \\
\tanh(\overleftarrow{W}_x E_x x_j + \overleftarrow{U}_x h_{j+1}), & \text{if } j \leq T_x
\end{cases}, \\
h_j &= (\overrightarrow{h}_j, \overleftarrow{h}_j)
\end{align*}
\]
Attentional encoder-decoder: Maths

decoder

\[ s_i = \begin{cases} 
\tanh(W_s h_i), & \text{if } i = 0 \\
\tanh(W_y E_y y_{i-1} + U_y s_{i-1} + C c_i) & \text{if } i > 0 
\end{cases} \]

\[ t_i = \tanh(U_o s_i + W_o E_y y_{i-1} + C_o c_i) \]

\[ y_i = \text{softmax}(V_o t_i) \]

attention model

\[ e_{ij} = v_a^\top \tanh(W_a s_{i-1} + U_a h_j) \]

\[ \alpha_{ij} = \text{softmax}(e_{ij}) \]

\[ c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \]
Attention model

**attention model**

- side effect: we obtain alignment between source and target sentence
- information can also flow along recurrent connections, so there is no guarantee that attention corresponds to alignment
- applications:
  - visualisation
  - replace unknown words with back-off dictionary [Jean et al., 2015]
  - ...

---

Economic growth has slowed down in recent years.

Das Wirtschaftswachstum hat sich in den letzten Jahren verlangsamt.

La croissance économique s'est ralentie ces dernières années.

---

Kyunghyun Cho


Sennrich, Haddow

Practical Neural Machine Translation
attention model also works with images:

\[ f = (a, \text{ man, is, jumping, into, a, lake, .}) \]
Fig. 5. Examples of the attention-based model attending to the correct object (white indicates the attended regions, underlines indicated the corresponding word) [22]
Application of Encoder-Decoder Model

Scoring (a translation)

\[ p(\text{La, croissance, économique, s’est, ralentie, ces, dernières, années, .} \mid \text{Economic, growth, has, slowed, down, in, recent, year, .}) = ? \]

Decoding (a source sentence)

Generate the most probable translation of a source sentence

\[ y^* = \arg \max_y p(y \mid \text{Economic, growth, has, slowed, down, in, recent, year, .}) \]
Decoding

**exact search**
- generate every possible sentence $T$ in target language
- compute score $p(T|S)$ for each
- pick best one

- intractable: $|\text{vocab}|^N$ translations for output length $N$
  $\rightarrow$ we need approximative search strategy
approximative search/1: greedy search

- at each time step, compute probability distribution $P(y_i|S, y_{<i})$
- select $y_i$ according to some heuristic:
  - sampling: sample from $P(y_i|S, y_{<i})$
  - greedy search: pick $\arg\max_y p(y_i|S, y_{<i})$
- continue until we generate $<eos>$

- efficient, but suboptimal
Decoding

approximative search/2: beam search

- maintain list of $K$ hypotheses (beam)
- at each time step, expand each hypothesis $k$: $p(y^k_i | S, y^k_{<i})$
- select $K$ hypotheses with highest total probability:

$$\prod_i p(y^k_i | S, y^k_{<i})$$

- relatively efficient . . . beam expansion parallelisable
- currently default search strategy in neural machine translation
- small beam ($K \approx 10$) offers good speed-quality trade-off

$$K = 3$$
Ensembles

- at each timestep, combine the probability distribution of $M$ different ensemble components.
- combine operator: typically average (log-)probability

$$\log P(y_i | S, y_{<i}) = \frac{\sum_{m=1}^{M} \log P_m(y_i | S, y_{<i})}{M}$$

- requirements:
  - same output vocabulary
  - same factorization of $Y$
- internal network architecture may be different
- source representations may be different
  (extreme example: ensemble-like model with different source languages [Junczys-Dowmunt and Grundkiewicz, 2016])
1. Introduction
2. Neural Networks — Basics
3. Language Models using Neural Networks
4. Attention-based NMT Model
5. Edinburgh’s WMT16 System
6. Analysis: Why does NMT work so well?
7. Building and Improving NMT Systems
8. Resources, Further Reading and Wrap-Up
Basic encoder-decoder-with-attention, plus:

1. Subword models to allow translation of rare/unknown words
   → since networks have small, fixed vocabulary
2. Back-translated monolingual data as additional training data
   → allows us to make use of extensive monolingual resources
3. Combination of left-to-right and right-to-left models
   → Reduces “label-bias” problem
4. Bayesian dropout
   → Improves generalisation performance with small training data
Subwords for NMT: Motivation

MT is an open-vocabulary problem

- compounding and other productive morphological processes
  - they charge a *carry-on bag fee*.
  - *sie erheben eine Handgepäckgebühr*.
- names
  - *Obama* (English; German)
  - *Обама* (Russian)
  - *オバマ* (o-ba-ma) (Japanese)
- technical terms, numbers, etc.

... but Neural MT architectures have small and fixed vocabulary
Subword units

segmentation algorithms: wishlist

- **open-vocabulary NMT**: encode *all* words through small vocabulary
- encoding generalizes to unseen words
- small text size
- good translation quality

our experiments

- after preliminary experiments, we use:
  - character n-grams (with shortlist of unsegmented words)
  - segmentation via *byte pair encoding*
bottom-up character merging

- starting point: character-level representation
  → computationally expensive
- compress representation based on information theory
  → byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair ('A', 'B') with 'AB'
- hyperparameter: when to stop
  → controls vocabulary size

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>vocabulary:</th>
</tr>
</thead>
<tbody>
<tr>
<td>'l o w &lt;/w&gt;'</td>
<td>5</td>
<td>low ernstid</td>
</tr>
<tr>
<td>'l o w e r &lt;/w&gt;'</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>'n e w e s t &lt;/w&gt;'</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>'w i d e s t &lt;/w&gt;'</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Byte pair encoding for word segmentation

bottom-up character merging

- starting point: character-level representation → computationally expensive
- compress representation based on information theory → byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair (‘A’, ‘B’) with ‘AB’
- hyperparameter: when to stop → controls vocabulary size

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>vocabulary:</th>
</tr>
</thead>
<tbody>
<tr>
<td>'low &lt;/w&gt;'</td>
<td>5</td>
<td>low &lt;/w&gt; erntid</td>
</tr>
<tr>
<td>'lower &lt;/w&gt;'</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>'newest &lt;/w&gt;'</td>
<td>6</td>
<td>es</td>
</tr>
<tr>
<td>'widest &lt;/w&gt;'</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Byte pair encoding for word segmentation

**bottom-up character merging**

- starting point: character-level representation → computationally expensive
- compress representation based on information theory → byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair (’A’,’B’) with ’AB’
- hyperparameter: when to stop → controls vocabulary size

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>vocabulary:</th>
</tr>
</thead>
<tbody>
<tr>
<td>’l o w &lt;/w&gt;’</td>
<td>5</td>
<td>low &lt;/w&gt; e r n s t i d</td>
</tr>
<tr>
<td>’l o w e r &lt;/w&gt;’</td>
<td>2</td>
<td>l o w &lt;/w&gt; e r n s t i d</td>
</tr>
<tr>
<td>’n e w est &lt;/w&gt;’</td>
<td>6</td>
<td>es est</td>
</tr>
<tr>
<td>’w i d est &lt;/w&gt;’</td>
<td>3</td>
<td>es est</td>
</tr>
</tbody>
</table>
Byte pair encoding for word segmentation

**bottom-up character merging**

- starting point: character-level representation
  → computationally expensive
- compress representation based on information theory
  → byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair ('A', 'B') with 'AB'
- hyperparameter: when to stop
  → controls vocabulary size

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>vocabulary:</th>
</tr>
</thead>
<tbody>
<tr>
<td>'l o w &lt;/w&gt;'</td>
<td>5</td>
<td>l o w &lt;/w&gt; e r n s t i d</td>
</tr>
<tr>
<td>'l o w e r &lt;/w&gt;'</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>'n e w est&lt;/w&gt;'</td>
<td>6</td>
<td>es est est&lt;/w&gt;</td>
</tr>
<tr>
<td>'w i d est&lt;/w&gt;'</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Byte pair encoding for word segmentation

bottom-up character merging

- starting point: character-level representation → computationally expensive
- compress representation based on information theory → byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair (’A’,’B’) with ’AB’
- hyperparameter: when to stop → controls vocabulary size

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>vocabulary:</th>
</tr>
</thead>
<tbody>
<tr>
<td>’lo w &lt;/w&gt;’</td>
<td>5</td>
<td>low</td>
</tr>
<tr>
<td>’lo w e r &lt;/w&gt;’</td>
<td>2</td>
<td>more words</td>
</tr>
<tr>
<td>’n e w est&lt;/w&gt;’</td>
<td>6</td>
<td>est</td>
</tr>
<tr>
<td>’w i d est&lt;/w&gt;’</td>
<td>3</td>
<td>lo</td>
</tr>
</tbody>
</table>
Byte pair encoding for word segmentation

bottom-up character merging

- starting point: character-level representation  
  → computationally expensive
- compress representation based on information theory  
  → byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair ('A', 'B') with 'AB'
- hyperparameter: when to stop  
  → controls vocabulary size

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>'low &lt;/w&gt;'</td>
<td>5</td>
</tr>
<tr>
<td>'low er &lt;/w&gt;'</td>
<td>2</td>
</tr>
<tr>
<td>'new est&lt;/w&gt;'</td>
<td>6</td>
</tr>
<tr>
<td>'wid est&lt;/w&gt;'</td>
<td>3</td>
</tr>
</tbody>
</table>

vocabulary: low er n s t i d  es est est</w> lo low
why BPE?

- open-vocabulary: operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency
  → trade-off between text length and vocabulary size

\[
\begin{align*}
\text{'lo\,w\,e\,s\,t</w>}' & \rightarrow \text{es} \\
\text{es\,t} & \rightarrow \text{est} \\
\text{est</w>}' & \rightarrow \text{est</w>}' \\
\text{l\,o\,w\,w} & \rightarrow \text{low}
\end{align*}
\]
Byte pair encoding for word segmentation

why BPE?

- open-vocabulary: operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency
  → trade-off between text length and vocabulary size

'llow est</w>'

\[
\begin{align*}
\text{l o w} & \rightarrow \text{lo} \\
\text{es} & \rightarrow \text{es} \\
\text{est} & \rightarrow \text{est} \\
\text{es} & \rightarrow \text{es} \\
\text{est} & \rightarrow \text{est}</w> \\
\text{l o} & \rightarrow \text{lo} \\
\text{lo w} & \rightarrow \text{low}
\end{align*}
\]
why BPE?

- open-vocabulary: operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency
  → trade-off between text length and vocabulary size

'low est </w>’

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>es</td>
<td>→ es</td>
</tr>
<tr>
<td>est</td>
<td>→ est</td>
</tr>
<tr>
<td>est &lt;/w&gt;</td>
<td>→ est&lt;/w&gt;</td>
</tr>
<tr>
<td>lo</td>
<td>→ lo</td>
</tr>
<tr>
<td>lo w</td>
<td>→ low</td>
</tr>
</tbody>
</table>
Byte pair encoding for word segmentation

why BPE?

- open-vocabulary: operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency → trade-off between text length and vocabulary size

'l o w est</w>’

<table>
<thead>
<tr>
<th>Original</th>
<th>BPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>'l o w est'</td>
<td>es est es</td>
</tr>
<tr>
<td></td>
<td>est est est</td>
</tr>
</tbody>
</table>
why BPE?

- open-vocabulary: operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency
  → trade-off between text length and vocabulary size

\[
\begin{align*}
'lo w est'</lo w est> & \rightarrow es \\
es t & \rightarrow est \\
est </w> & \rightarrow est</w> \\
lo & \rightarrow lo \\
lo w & \rightarrow low
\end{align*}
\]
Byte pair encoding for word segmentation

why BPE?
- open-vocabulary: operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency → trade-off between text length and vocabulary size

’e s’ → ‘es’
es t → ‘est’
est ’<w>’ → ‘est</w>’
’l o’ → ‘lo’
’lo w’ → ‘low’
## Evaluation: data and methods

<table>
<thead>
<tr>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMT 15 English→German and English→Russian</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>model</th>
</tr>
</thead>
<tbody>
<tr>
<td>attentional encoder–decoder neural network</td>
</tr>
<tr>
<td>parameters and settings as in [Bahdanau et al, 2014]</td>
</tr>
</tbody>
</table>
Subword NMT: Translation Quality

![Bar chart showing translation quality comparison between SMT, word-level NMT (with back-off), subword-level NMT: character bigrams, and subword-level NMT: BPE for EN-DE and EN-RU pairs.]

- **SMT** [Sennrich and Haddow, 2015, Haddow et al., 2015]
- **word-level NMT (with back-off)** [Jean et al., 2015]
- **subword-level NMT: character bigrams**
- **subword-level NMT: BPE**

The chart illustrates the BLEU scores for translation quality, with higher scores indicating better performance.
Subword NMT: Translation Quality

NMT Results EN-RU

unigram $F_1$

training set frequency rank

- subword-level NMT: BPE
- subword-level NMT: char bigrams
- word-level (with back-off)
- word-level (no back-off)
<table>
<thead>
<tr>
<th>system</th>
<th>sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>health research institutes</td>
</tr>
<tr>
<td>reference</td>
<td>Gesundheitsforschungsinstitute</td>
</tr>
<tr>
<td>word-level (with back-off)</td>
<td>Forschungsinstitute</td>
</tr>
<tr>
<td>character bigrams</td>
<td>Forschungsinstitute</td>
</tr>
<tr>
<td>BPE</td>
<td>Gesundheitsforschungsinstitute</td>
</tr>
<tr>
<td></td>
<td>rakfisk</td>
</tr>
<tr>
<td>source</td>
<td>rakfisk</td>
</tr>
<tr>
<td>reference</td>
<td>Rakfisk → UNK → rakfisk</td>
</tr>
<tr>
<td>word-level (with back-off)</td>
<td>Rakfisk → PaKFiSk (rakfiska)</td>
</tr>
<tr>
<td>character bigrams</td>
<td>Rakfisk → PaKFiSk (rakfiska)</td>
</tr>
<tr>
<td>BPE</td>
<td>Rakfisk → PaKFiSk (rakfiska)</td>
</tr>
</tbody>
</table>
BPE in WMT16 Systems

- Used **Joint** BPE
  - Just concatenate source and target, then train
  - Named-entities are split consistently
- Learn 89,500 merge operations
- Use ISO-9 transliteration for Russian:
  - transliterate Russian corpus into Latin script
  - learn BPE operations on concatenation of English and transliterated Russian corpus
  - transliterate BPE operations into Cyrillic
  - for Russian, apply both Cyrillic and Latin BPE operations → concatenate BPE files
- Set vocabulary size according to BPE vocabulary

Code available: [https://github.com/rsennrich/subword-nmt](https://github.com/rsennrich/subword-nmt)
### Why Monolingual Data for Phrase-based SMT?
- more training data ✓
- relax independence assumptions ✓
- more appropriate training data (domain adaptation) ✓

### Why Monolingual Data for NMT?
- more training data ✓
- relax independence assumptions ✗
- more appropriate training data (domain adaptation) ✓
encoder-decoder already conditions on previous target words

↓

no architecture change required to learn from monolingual data
## Monolingual Training Instances

### Output prediction

- \( p(y_i) \) is a function of hidden state \( s_i \), previous output \( y_{i-1} \), and source context vector \( c_i \)
- only difference to monolingual RNN: \( c_i \)

### Problem

we have no source context \( c_i \) for monolingual training instances
Output prediction

- $p(y_i)$ is a function of hidden state $s_i$, previous output $y_{i-1}$, and source context vector $c_i$
- only difference to monolingual RNN: $c_i$

Problem

we have no source context $c_i$ for monolingual training instances

Solutions

two methods to deal with missing source context:
- empty/dummy source context $c_i$
  → danger of unlearning conditioning on source
- produce synthetic source sentence via back-translation
  → get approximation of $c_i$
## Monolingual Training Instances

### Dummy source
- 1-1 mix of parallel and monolingual training instances
- Randomly sample from monolingual data each epoch
- Freeze encoder/attention layers for monolingual training instances

### Synthetic source
- 1-1 mix of parallel and monolingual training instances
- Randomly sample from back-translated data
- Training does not distinguish between real and synthetic parallel data
Evaluation: WMT 15 English $\rightarrow$ German

![Bar Chart]

<table>
<thead>
<tr>
<th>BLEU Score</th>
<th>Syntax-based</th>
<th>Parallel</th>
<th>+Monolingual</th>
<th>+Synthetic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>24.4</td>
<td>23.6</td>
<td>24.6</td>
<td>26.5</td>
</tr>
</tbody>
</table>

(NMT systems are ensemble of 4)
Evaluation: WMT 15 German → English

<table>
<thead>
<tr>
<th></th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBSMT</td>
<td>29.3</td>
</tr>
<tr>
<td>parallel</td>
<td>26.7</td>
</tr>
<tr>
<td>+synthetic</td>
<td>30.4</td>
</tr>
<tr>
<td>+synth-ens4</td>
<td>31.6</td>
</tr>
</tbody>
</table>
Why is monolingual data helpful?

- Domain adaptation effect
- Reduces over-fitting
- Improves fluency

(See [Sennrich et al., 2016] for more analysis.)
Target history is strong signal for next prediction

- History is reliable at training time, but not at test time
- Low-entropy output words lead to poor translation
- Similar to label bias problem

Reranking with reverse model can help

1. Train two models, one has target reversed
2. Generate $n$-best lists with one model
3. Rescore lists with second model
4. Rerank using combined scores

Consistent increase (0.5 – 1) in BLEU
Bayesian Dropout

- Dropout (randomly zeroing activations in training) prevents overfitting
- Follow [Gal, 2015] and repeat mask across timesteps
- Necessary for English↔Romanian (0.6M sentences)
- Masks of 0.1-0.2 provide gain of 4-5 BLEU
Checkpoint Ensembling

Ensembling improves **performance** and **stability**

- Checkpoint ensembling much cheaper than independent runs

\[ p(e|f) = p_1(e|f) \times p_2(e|f) \times p_3(e|f) \times p_4(e|f) \]
Putting it all together: WMT16 Results

![Graph showing BLEU scores for various language pairs and different data combinations.](chart)

- **Parallel Data**
- **+Synthetic Data**
- **+Ensemble**
- **+R2L Reranking**

Languages:
- **EN→CS**
- **EN→DE**
- **EN→RO**
- **EN→RU**
- **CS→EN**
- **DE→EN**
- **RO→EN**
- **RU→EN**

Scores range from 0.0 to 40.0 in increments of 10.0.
Comparison between phrase-based and neural MT

human analysis of NMT (reranking) [Neubig et al., 2015]

- NMT is more grammatical
  - word order
  - insertion/deletion of function words
  - morphological agreement
- minor degradation in lexical choice?
Comparison between phrase-based and neural MT

Analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- Human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

<table>
<thead>
<tr>
<th>System</th>
<th>HTER (no shift)</th>
<th>HTER (shift only)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>word</td>
<td>lemma</td>
</tr>
<tr>
<td>PBSMT [Ha et al., 2015]</td>
<td>28.3</td>
<td>23.2</td>
</tr>
<tr>
<td>NMT [Luong and Manning, 2015]</td>
<td>21.7</td>
<td>18.7</td>
</tr>
</tbody>
</table>

- Word-level is closer to lemma-level performance: better at inflection/agreement
- Improvement on lemma-level: better lexical choice
- Fewer shift errors: better word order
Comparison between phrase-based and neural MT

### Analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- Human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

<table>
<thead>
<tr>
<th>System</th>
<th>HTER (no shift)</th>
<th>HTER (shift only)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>word</td>
<td>lemma</td>
</tr>
<tr>
<td>PBSMT [Ha et al., 2015]</td>
<td>28.3</td>
<td>23.2</td>
</tr>
<tr>
<td>NMT [Luong and Manning, 2015]</td>
<td>21.7</td>
<td>18.7</td>
</tr>
</tbody>
</table>

- Word-level is closer to lemma-level performance: better at inflection/agreement
- Improvement on lemma-level: better lexical choice
- Fewer shift errors: better word order
Comparison between phrase-based and neural MT

**Analysis of IWSLT 2015 results [Bentivogli et al., 2016]**

- Human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

<table>
<thead>
<tr>
<th>System</th>
<th>HTER (no shift)</th>
<th>HTER (shift only)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>word</td>
<td>lemma</td>
</tr>
<tr>
<td>PBSMT [Ha et al., 2015]</td>
<td>28.3</td>
<td>23.2</td>
</tr>
<tr>
<td>NMT [Luong and Manning, 2015]</td>
<td>21.7</td>
<td>18.7</td>
</tr>
</tbody>
</table>

- Word-level is closer to lemma-level performance: better at inflection/agreement
- Improvement on lemma-level: better lexical choice
- Fewer shift errors: better word order
Comparison between phrase-based and neural MT

Analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- Human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

<table>
<thead>
<tr>
<th>System</th>
<th>HTER (no shift)</th>
<th>HTER (shift only)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>word</td>
<td>lemma</td>
</tr>
<tr>
<td>PBSMT [Ha et al., 2015]</td>
<td>28.3</td>
<td>23.2</td>
</tr>
<tr>
<td>NMT [Luong and Manning, 2015]</td>
<td>21.7</td>
<td>18.7</td>
</tr>
</tbody>
</table>

- Word-level is closer to lemma-level performance: better at inflection/agreement
- Improvement on lemma-level: better lexical choice
- Fewer shift errors: better word order
Adequacy vs. Fluency in WMT16 Evaluation

Figure: WMT16 direct assessment results
Human Evaluation in TraMOOC

- comparison of NMT and PBSMT for EN→{DE,EL,PT,RU}
- direct assessment:
  - NMT obtains higher fluency judgment than PBSMT: +10%
  - NMT only obtains small improvement in adequacy judgment: +1%
- post-editing:
  - NMT reduces technical effort (keystrokes): -13%
  - small reduction in post-editing time: -4%
  → NMT errors more difficult to identify

### Error Annotation

<table>
<thead>
<tr>
<th>category</th>
<th>SMT</th>
<th>NMT</th>
<th>difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>inflectional morphology</td>
<td>2274</td>
<td>1799</td>
<td>-21%</td>
</tr>
<tr>
<td>word order</td>
<td>1098</td>
<td>691</td>
<td>-37%</td>
</tr>
<tr>
<td>omission</td>
<td>421</td>
<td>362</td>
<td>-14%</td>
</tr>
<tr>
<td>addition</td>
<td>314</td>
<td>265</td>
<td>-16%</td>
</tr>
<tr>
<td>mistranslation</td>
<td>1593</td>
<td>1552</td>
<td>-3%</td>
</tr>
<tr>
<td>&quot;no issue&quot;</td>
<td>449</td>
<td>788</td>
<td>+75%</td>
</tr>
</tbody>
</table>

Sennrich, Haddow

Practical Neural Machine Translation
Assessing MT Quality with Contrastive Translation Pairs

Questions
- how well does NMT perform for specific linguistic phenomena?
- example: is grammaticality affected by choice of subword unit?

Method [Sennrich, 2017]
- compare probability of human reference translation with contrastive translation that introduces a specific type of error
  → NMT model should prefer reference
- errors related to:
  - morphosyntactic agreement
  - discontiguous units of meaning
  - polarity
  - transliteration
Contrastive Translation Pairs: Example

English: [...] that the **plan will** be approved

German (correct): [...] dass der **Plan** verabschiedet **wird**

German (contrastive): * [...] dass der **Plan** verabschiedet **werden**

subject-verb agreement
Results

- WMT16 NMT system detects agreement errors with high accuracy – 96.6–98.7%.
- Character-level system [Lee et al., 2016] better than BPE-to-BPE system at transliteration, but worse at morphosyntactic agreement.
- Difference higher for agreement over long distances.
Experimental Setup

- Training and test drawn from UN corpus
  - Multi-parallel, 11M lines
  - Arabic, Chinese, English, French, Russian, Spanish
- Use only parallel data, evaluate with BLEU on 4000 sentences
Why is neural MT output more grammatical?

**phrase-based SMT**
- log-linear combination of many “weak” features
- data sparseness triggers back-off to smaller units
- strong independence assumptions

**neural MT**
- end-to-end trained model
- generalization via continuous space representation
- output conditioned on full source text and target history
Resource Usage

- We all want our experiments to finish faster . . .
- What influences training speed/memory usage?
  - Number of model parameters, especially vocabulary size
  - Size of training instance (max. length \( \times \) batch size)
  - Hardware and library versions

- Decoding speed
  - Less important for NMT researchers
  - Standard Nematus model \( \rightarrow \) Use **AmuNMT** (hand-crafted GPU code)
## Hardware/Library Choice

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Theano</th>
<th>CuDNN</th>
<th>gpuarray</th>
<th>Sentence/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (Xeon E5-2680)</td>
<td>0.8.2</td>
<td>No</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>GPU (Titan X Pascal)</td>
<td>0.8.2</td>
<td>No</td>
<td>No</td>
<td>83</td>
</tr>
<tr>
<td>GPU (Titan X Pascal)</td>
<td>0.8.2</td>
<td>5.10</td>
<td>No</td>
<td>138</td>
</tr>
<tr>
<td>GPU (Titan X Pascal)</td>
<td>0.9b</td>
<td>5.10</td>
<td>No</td>
<td>171</td>
</tr>
<tr>
<td>GPU (Titan Black)</td>
<td>0.9b</td>
<td>5.10</td>
<td>No</td>
<td>109</td>
</tr>
<tr>
<td>GPU (Titan X)</td>
<td>0.9b</td>
<td>5.10</td>
<td>No</td>
<td>110</td>
</tr>
<tr>
<td>GPU (GTX 1080)</td>
<td>0.9b</td>
<td>5.10</td>
<td>No</td>
<td>177</td>
</tr>
<tr>
<td>GPU (Tesla M60)</td>
<td>0.9b</td>
<td>5.10</td>
<td>No</td>
<td>110</td>
</tr>
<tr>
<td>GPU (Titan X Pascal)</td>
<td>0.9rc3</td>
<td>5.10</td>
<td>Yes</td>
<td>227</td>
</tr>
</tbody>
</table>
Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length \cdot batchsize)
- we show some profile output for guidance:
  - Nematus (`test_train.sh’)
  - NVIDIA GTX 1080 GPU

<table>
<thead>
<tr>
<th>layer size</th>
<th>vocabulary</th>
<th>batchsize</th>
<th>maxlen</th>
<th>GPU memory (peak)</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>embed</td>
<td>hidden</td>
<td>source</td>
<td>target</td>
<td></td>
<td>sents/s</td>
</tr>
<tr>
<td>256</td>
<td>1024</td>
<td>60000</td>
<td>80</td>
<td>6.6 GB</td>
<td>87</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>80</td>
<td>4.9 GB</td>
<td>110</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>3.6 GB</td>
<td>79</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>60000</td>
<td>40</td>
<td>2.7 GB</td>
<td>95</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>60000</td>
<td>40</td>
<td>2.3 GB</td>
<td>145</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>30000</td>
<td>40</td>
<td>2.1 GB</td>
<td>148</td>
</tr>
</tbody>
</table>

Sennrich, Haddow
Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length \times batchsize)
- we show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

<table>
<thead>
<tr>
<th>layer size</th>
<th>vocabulary</th>
<th>batchsize</th>
<th>maxlen</th>
<th>GPU memory (peak)</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>embed</td>
<td>hidden</td>
<td>source</td>
<td>target</td>
<td>sents/s</td>
</tr>
<tr>
<td>256</td>
<td>30000</td>
<td>512</td>
<td>40</td>
<td>50</td>
<td>1.2 GB</td>
</tr>
<tr>
<td>256</td>
<td>30000</td>
<td>512</td>
<td>60000</td>
<td>40</td>
<td>2.1 GB</td>
</tr>
<tr>
<td>256</td>
<td>60000</td>
<td>512</td>
<td>60000</td>
<td>40</td>
<td>2.3 GB</td>
</tr>
<tr>
<td>256</td>
<td>60000</td>
<td>1024</td>
<td>60000</td>
<td>40</td>
<td>2.7 GB</td>
</tr>
<tr>
<td>512</td>
<td>60000</td>
<td>1024</td>
<td>60000</td>
<td>40</td>
<td>3.6 GB</td>
</tr>
<tr>
<td>512</td>
<td>60000</td>
<td>1024</td>
<td>60000</td>
<td>80</td>
<td>4.9 GB</td>
</tr>
<tr>
<td>512</td>
<td>60000</td>
<td>1024</td>
<td>60000</td>
<td>80</td>
<td>6.6 GB</td>
</tr>
</tbody>
</table>
Hyperparameters: Efficiency

- Hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- Memory consumption affected by
  - Number of model parameters
  - Size of training instance (length \cdot batchsize)
- We show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

<table>
<thead>
<tr>
<th>layer size</th>
<th>vocabulary</th>
<th>batchsize</th>
<th>maxlen</th>
<th>GPU memory</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>embed</td>
<td>hidden</td>
<td>source</td>
<td>target</td>
<td>(peak)</td>
<td>sents/s</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>30000</td>
<td>30000</td>
<td>1.2 GB</td>
<td>174</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>30000</td>
<td>60000</td>
<td>2.1 GB</td>
<td>148</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>60000</td>
<td>60000</td>
<td>2.3 GB</td>
<td>145</td>
</tr>
<tr>
<td>256</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>2.7 GB</td>
<td>95</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>3.6 GB</td>
<td>79</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>4.9 GB</td>
<td>110</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>6.6 GB</td>
<td>87</td>
</tr>
</tbody>
</table>
Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length · batchsize)
- we show some profile output for guidance:
  - Nematus (’test_train.sh’)
  - NVIDIA GTX 1080 GPU

<table>
<thead>
<tr>
<th>layer size</th>
<th>vocabulary</th>
<th>batchsize</th>
<th>maxlen</th>
<th>GPU memory (peak)</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>embed</td>
<td>hidden</td>
<td>source</td>
<td>target</td>
<td></td>
<td>sents/s</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>30000</td>
<td>30000</td>
<td>1.2 GB</td>
<td>174</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>30000</td>
<td>60000</td>
<td>2.1 GB</td>
<td>148</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>60000</td>
<td>60000</td>
<td>2.3 GB</td>
<td>145</td>
</tr>
<tr>
<td>256</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>2.7 GB</td>
<td>95</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>3.6 GB</td>
<td>79</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>4.9 GB</td>
<td>110</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>6.6 GB</td>
<td>87</td>
</tr>
</tbody>
</table>
Hyperparameters: Efficiency

- Hyperparameters affect peak GPU memory and speed.
- GPU memory is often the bottleneck in NMT training.
- Memory consumption affected by:
  - Number of model parameters.
  - Size of training instance (length \cdot batchsize).
- We show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

<table>
<thead>
<tr>
<th>embed size</th>
<th>hidden size</th>
<th>source vocabulary</th>
<th>target vocabulary</th>
<th>batchsize</th>
<th>maxlen</th>
<th>GPU memory (peak)</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>sents/s</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>30000</td>
<td>30000</td>
<td>40</td>
<td>50</td>
<td>1.2 GB</td>
<td>174</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>30000</td>
<td>60000</td>
<td>40</td>
<td>50</td>
<td>2.1 GB</td>
<td>148</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>60000</td>
<td>60000</td>
<td>40</td>
<td>50</td>
<td>2.3 GB</td>
<td>145</td>
</tr>
<tr>
<td>256</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>40</td>
<td>50</td>
<td>2.7 GB</td>
<td>95</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>40</td>
<td>50</td>
<td>3.6 GB</td>
<td>79</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>80</td>
<td>50</td>
<td>4.9 GB</td>
<td>110</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>80</td>
<td>80</td>
<td>6.6 GB</td>
<td>87</td>
</tr>
</tbody>
</table>
Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length $\cdot$ batchsize)
- we show some profile output for guidance:
  - Nematus (‘test_train.sh’)
  - NVIDIA GTX 1080 GPU

<table>
<thead>
<tr>
<th>layer size</th>
<th>vocabulary</th>
<th>batchsize</th>
<th>maxlen</th>
<th>GPU memory (peak)</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>embed</td>
<td>hidden</td>
<td>source</td>
<td>target</td>
<td>sents/s</td>
<td>words/s</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>30000</td>
<td>30000</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>30000</td>
<td>60000</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>60000</td>
<td>60000</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>256</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>80</td>
<td>50</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>
Hyperparameters: Efficiency

- Hyperparameters affect peak GPU memory and speed.
- GPU memory is often the bottleneck in NMT training.
- Memory consumption affected by:
  - Number of model parameters
  - Size of training instance (length \times batchsize)
- We show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

<table>
<thead>
<tr>
<th>layer size</th>
<th>vocabulary</th>
<th>batchsize</th>
<th>maxlen</th>
<th>GPU memory (peak)</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>embed</td>
<td>hidden</td>
<td>source</td>
<td>target</td>
<td>sents/s</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>30000</td>
<td>30000</td>
<td>40</td>
<td>1.2 GB</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>60000</td>
<td>60000</td>
<td>40</td>
<td>2.1 GB</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>60000</td>
<td>60000</td>
<td>40</td>
<td>2.3 GB</td>
</tr>
<tr>
<td>256</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>40</td>
<td>2.7 GB</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>40</td>
<td>3.6 GB</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>80</td>
<td>4.9 GB</td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td>60000</td>
<td>60000</td>
<td>80</td>
<td>6.6 GB</td>
</tr>
</tbody>
</table>
Training: Minibatches

Why Minibatches?

- parallelization (GPUs!) is more efficiently with larger matrices
- easy way to increase matrix size: batch up training instances
- other advantage: stabilizes updates

- how do we deal with difference in sentence length in batch?
- standard solution: pad sentence with special tokens

### Table 1

<table>
<thead>
<tr>
<th>layer no.</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>input alphabet size (X)</td>
<td>300</td>
</tr>
<tr>
<td>embedding sizes</td>
<td>256</td>
</tr>
<tr>
<td>char RNN (forward)</td>
<td>400</td>
</tr>
<tr>
<td>char RNN (backward)</td>
<td>400</td>
</tr>
<tr>
<td>attention</td>
<td>300</td>
</tr>
<tr>
<td>char decoder</td>
<td>400</td>
</tr>
<tr>
<td>target alphabet size (T)</td>
<td>300</td>
</tr>
</tbody>
</table>

Table 1. Hyperparameter values used for training the char-to-char model. Where $\Sigma_{src}$ and $\Sigma_{trg}$ represent the number of classes in the source and target languages, respectively.

### Table 2

<table>
<thead>
<tr>
<th>layer no.</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>input alphabet size (X)</td>
<td>300</td>
</tr>
<tr>
<td>embedding sizes</td>
<td>256</td>
</tr>
<tr>
<td>char RNN (forward)</td>
<td>400</td>
</tr>
<tr>
<td>spaces RNN (forward)</td>
<td>400</td>
</tr>
<tr>
<td>spaces RNN (backward)</td>
<td>400</td>
</tr>
<tr>
<td>attention</td>
<td>300</td>
</tr>
<tr>
<td>char decoder</td>
<td>400</td>
</tr>
<tr>
<td>target alphabet size (T)</td>
<td>300</td>
</tr>
</tbody>
</table>

Table 2. Hyperparameter values used for training the char2word-to-char model. Where $\Sigma_{src}$ and $\Sigma_{trg}$ represent the number of classes in the source and target languages, respectively.

5.3. Results

5.3.1. Quantitative

The quantitative results of our models are illustrated in table 3. Notice that the char2word-to-char model outperforms the char-to-char model on all datasets (average 1.28 BLEU performance increase). This could be an indication that either having hierarchical, word-like, representations on the encoder or simply the fact that the encoder was significantly smaller, helps in NMT when using a character decoder with attention.
Training: Minibatches

Speed-ups

- sort sentences of same length together [Sutskever et al., 2014]
- adjust batch size depending on length [Johansen et al., 2016]

Table 1. Hyperparameter values used for training the char-to-char model. Where $\Sigma_{src}$ and $\Sigma_{trg}$ represent the number of classes in the source and target languages, respectively.

<table>
<thead>
<tr>
<th>Layer No.</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input alphabet size (X)</td>
<td>300</td>
</tr>
<tr>
<td>Embedding sizes</td>
<td>256</td>
</tr>
<tr>
<td>Char RNN (forward)</td>
<td>400</td>
</tr>
<tr>
<td>Char RNN (backward)</td>
<td>400</td>
</tr>
<tr>
<td>Attention</td>
<td>300</td>
</tr>
<tr>
<td>Char decoder</td>
<td>400</td>
</tr>
<tr>
<td>Target alphabet size (T)</td>
<td>300</td>
</tr>
</tbody>
</table>

Table 2. Hyperparameter values used for training the char2word-to-char model. Where $\Sigma_{src}$ and $\Sigma_{trg}$ represent the number of classes in the source and target languages, respectively.

<table>
<thead>
<tr>
<th>Layer No.</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input alphabet size (X)</td>
<td>300</td>
</tr>
<tr>
<td>Embedding sizes</td>
<td>256</td>
</tr>
<tr>
<td>Char RNN (forward)</td>
<td>400</td>
</tr>
<tr>
<td>Spaces RNN (forward)</td>
<td>400</td>
</tr>
<tr>
<td>Spaces RNN (backward)</td>
<td>400</td>
</tr>
<tr>
<td>Attention</td>
<td>300</td>
</tr>
<tr>
<td>Char decoder</td>
<td>400</td>
</tr>
<tr>
<td>Target alphabet size (T)</td>
<td>300</td>
</tr>
</tbody>
</table>

5.3. Results

5.3.1. Quantitative

The quantitative results of our models are illustrated in table 3. Notice that the char2word-to-char model outperforms the char-to-char model on all datasets (average 1.28 BLEU performance increase). This could be an indication that either having hierarchical, word-like, representations on the encoder or simply the fact that the encoder was significantly smaller, helps in NMT when using a character decoder with attention.
Out-of-memory: what to do

- little effect on quality:
  - reduce batch size
  - remove long sentences (also in validation!)
  - tie embedding layer and output layer in decoder [Press and Wolf, 2017] (’--tie_decoder_embeddings’ in Nematus)
  - model parallelism: different parts of model on different GPU

- unknown (or negative) effect on quality:
  - reduce layer size
  - reduce target vocabulary
Training and Convergence

- BLEU more unstable than cross-entropy
- useful convergence criteria: BLEU early stopping
Decoding Efficiency

How to make decoding fast?

- Small beam size is often sufficient
- Greedy decoding can be competitive in quality → especially with knowledge distillation [Kim and Rush, 2016]
- Filter output vocabulary [Jean et al., 2015, L’Hostis et al., 2016] based on which words commonly co-occur with source words
- process multiple sentences in batch [Wu et al., 2016]
- low-precision arithmetic [Wu et al., 2016] (requires suitable hardware)

NB: Amun supports batching, vocabulary filtering
Decoding Speed: Nematus vs. Amun

- **Single GPU, single model, Titan X (Pascal)**
Improving Translation Quality

There are many possible ways of improving the basic system:

1. Improve corpus preparation
2. Domain adaptation
3. Obtain appropriate synthetic data
4. Hybrid of NMT and traditional SMT
5. Add extra linguistic information
6. Minimum risk training
7. Deep models
8. Hyperparameter exploration
Corpus Preparation

Cleaning

Tokenisation

Case normalisation

Subword segmentation
Corpus Preparation

Cleaning

Tokenisation

Case normalisation

Subword segmentation

- Punctuation/encoding/spelling normalisation
- Language identification
-Removing non-parallel segments
Corpus Preparation

- Cleaning
  - Punctuation/encoding/spelling normalisation
  - Language identification
  - Removing non-parallel segments

- Tokenisation

- Case normalisation
  - Lowercasing
  - Truecasing (convert to most frequent)
  - Headlines etc.

- Subword segmentation
Corpus Preparation

Cleaning
- Punctuation/encoding/spelling normalisation
- Language identification
- Removing non-parallel segments

Tokenisation

Case normalisation
- Lowercasing
- Truecasing (convert to most frequent)
- Headlines etc.

Subword segmentation
- Statistical
- Linguistically motivated
Effect of Noise in Training Data

- [Chen et al., 2016] add noise to WMT EN-FR training data
- artificial noise: permute order of target sentences
- conclusion: NMT is more sensitive to (some types of) noise than SMT

<table>
<thead>
<tr>
<th></th>
<th>0%</th>
<th>10%</th>
<th>20%</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMT</td>
<td>32.74</td>
<td>32.69</td>
<td>32.61</td>
<td>31.96</td>
</tr>
<tr>
<td>NMT</td>
<td>35.44</td>
<td>34.83</td>
<td>32.05</td>
<td>30.11</td>
</tr>
</tbody>
</table>

results from presentation of [Chen et al., 2016] at AMTA 2016
Domain adaptation with continued training

- SGD is sensitive to order of training instances
- best practice:
  - first train on all available data
  - continue training on in-domain data
- Large BLEU improvements reported with minutes of training time
  
[Sennrich et al., 2016, Luong and Manning, 2015, Crego et al., 2016]

---

**Fine-tuning in IWSLT (en-de)**

<table>
<thead>
<tr>
<th></th>
<th>tst2013</th>
<th>tst2014</th>
<th>tst2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLEU</td>
<td>26.5</td>
<td>23.5</td>
<td>25.5</td>
</tr>
<tr>
<td></td>
<td>30.4</td>
<td>25.9</td>
<td>28.4</td>
</tr>
</tbody>
</table>

---

**Generic system (≈ 8M sentences), Fine-tune with TED (≈ 200k )**
Continued training with synthetic data

- what if we have monolingual in-domain training data?
- we compare fine-tuning with:
  - 200,000 sentence pairs in-domain
  - 200,000 target-language sentences in-domain, plus automatic back-translation

<table>
<thead>
<tr>
<th>system</th>
<th>BLEU (tst2015)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMT data</td>
<td>25.5</td>
</tr>
<tr>
<td>fine-tuned on in-domain (parallel)</td>
<td>28.4</td>
</tr>
<tr>
<td>fine-tuned on in-domain (synthetic)</td>
<td>26.7</td>
</tr>
</tbody>
</table>

English → German translation performance on IWSLT test set (TED talks).

→ parallel in-domain data is better, but domain adaptation with monolingual data is possible
→ WMT16 results (using large synthetic news corpora)

[Sennrich et al., 2016]
Continued training with synthetic data

Problem
How to create synthetic data from source-language in-domain data?

Solution
1. Gather source-language in-domain data.
2. Translate to target language.
3. Use this translated data to select from CommonCrawl corpus.
4. Back-translate selected data to create synthetic data.
Problem

How to create synthetic data from source-language in-domain data?

Solution

1. Gather source-language in-domain data.
2. Translate to target language
3. Use this translated data to select from CommonCrawl corpus
4. Back-translate selected data to create synthetic data
Continued training with synthetic data

Setup

- Language pairs: English → Czech, German, Polish and Romanian
- Domains: Two healthcare websites (NHS 24 and Cochrane)
- Baselines: Data drawn from WMT releases and OPUS
- Fine-tuning:
  - Use crawls of full websites as selection “seed”
  - Continue training with 50-50 synthetic/parallel mix
Continued Training with Synthetic Data:
Sample Learning Curve

- English → Polish, select using Cochrane
- Main training on general domain, finetune on 50-50 mix
Use domain interpolation to mix general and in-domain

--use_domain_interpolation
--domain_interpolation_indomain_datasets
--domain_interpolation_(min|max)
--domain_interpolation_inc
NMT Hybrid Models

Model combination (ensembling) is well established

Several ways to combine NMT with PBMT / Syntax-based MT:
- Re-ranking output of traditional SMT with NMT [Neubig et al., 2015]
- Incorporating NMT as feature function in PBMT [Junczys-Dowmunt et al., 2016b]
- Rescoring hiero lattices with NMT [Stahlberg et al., 2016]

Reduces chance of “bizarre” NMT outputs
NMT as feature function in PBMT [Junczys-Dowmunt et al., 2016b] → results depend on relative performance of PBMT and NMT

![Graph showing BLEU scores for English→Russian and Russian→English translations for phrase-based SMT, neural MT, and hybrid models.](image)
Why Linguistic Features?

disambiguate words by POS

<table>
<thead>
<tr>
<th>English</th>
<th>German</th>
</tr>
</thead>
<tbody>
<tr>
<td>close\textsubscript{verb}</td>
<td>schließen</td>
</tr>
<tr>
<td>close\textsubscript{adj}</td>
<td>nah</td>
</tr>
<tr>
<td>close\textsubscript{noun}</td>
<td>Ende</td>
</tr>
</tbody>
</table>

source: *We thought a win like this might be close\textsubscript{adj}.*  
reference: *Wir dachten, dass ein solcher Sieg nah sein könnte.*  
baseline NMT: *Wir dachten, ein Sieg wie dieser könnteschließen.*
Why Linguistic Features?

better generalization; combat data sparsity

<table>
<thead>
<tr>
<th>word form</th>
<th>German</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>liegen</td>
<td>lie</td>
<td></td>
</tr>
<tr>
<td>liegst</td>
<td>lie</td>
<td></td>
</tr>
<tr>
<td>lag</td>
<td>lay</td>
<td></td>
</tr>
<tr>
<td>läge</td>
<td>lay</td>
<td></td>
</tr>
</tbody>
</table>
Why Linguistic Features?

better generalization; combat data sparsity

<table>
<thead>
<tr>
<th>word form</th>
<th>lemma</th>
<th>morph. features</th>
</tr>
</thead>
<tbody>
<tr>
<td>liegen (lie)</td>
<td>liegen (lie)</td>
<td>(3.p.pl. present)</td>
</tr>
<tr>
<td>liegst (lie)</td>
<td>liegen (lie)</td>
<td>(2.p.sg. present)</td>
</tr>
<tr>
<td>lag (lay)</td>
<td>liegen (lie)</td>
<td>(3.p.sg. past)</td>
</tr>
<tr>
<td>läge (lay)</td>
<td>liegen (lie)</td>
<td>(3.p.sg. subjunctive II)</td>
</tr>
</tbody>
</table>
Neural Machine Translation: Multiple Input Features

Use separate embeddings for each feature, then concatenate

baseline: only word feature

\[
E(\text{close}) = \begin{bmatrix} 0.5 \\ 0.2 \\ 0.3 \\ 0.1 \end{bmatrix}
\]

\[ |F| \text{ input features} \]

\[
E_1(\text{close}) = \begin{bmatrix} 0.4 \\ 0.1 \\ 0.2 \end{bmatrix} \quad E_2(\text{adj}) = \begin{bmatrix} 0.1 \end{bmatrix} \quad E_1(\text{close}) \parallel E_2(\text{adj}) = \begin{bmatrix} 0.4 \\ 0.1 \\ 0.2 \\ 0.1 \end{bmatrix}
\]
Experiments

Features
- lemmas
- morphological features
- POS tags
- dependency labels
- BPE tags

Data
- WMT16 training/test data
- English↔German and English→Romanian
Results: **BLEU**

<table>
<thead>
<tr>
<th>Language Pair</th>
<th>Baseline</th>
<th>All Features</th>
<th>Baseline (+synthetic data)</th>
<th>All Features (+synthetic data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>English → German</td>
<td>27.8</td>
<td>33.1</td>
<td>33.2</td>
<td>33.2</td>
</tr>
<tr>
<td>German → English</td>
<td>31.4</td>
<td>32.9</td>
<td>37.5</td>
<td>38.5</td>
</tr>
<tr>
<td>English → Romanian</td>
<td>23.8</td>
<td>24.8</td>
<td>28.2</td>
<td>29.2</td>
</tr>
</tbody>
</table>
The standard NMT training objective is cross-entropy → maximise probability of training data

In traditional SMT, we usually tune for BLEU

Can train NMT to minimise Expected Loss

\[ \sum_{s=1}^{S} \mathbb{E}_{p(y|x^{(s)})} \left[ \Delta(y, y^{(s)}) \right] \]

(Loss function: \( \Delta \); Training pair: \((x^{(s)}, y^{(s)})\))

Run MRT after training with cross-entropy loss

Approximate expectation with sum over samples
Minimum Risk Training in Nematus

Recipe:
- Train initial model with standard cross-entropy training
- Continue training with ’--objective MRT’

Sensitive to hyperparameters
→ Use small learning rate with SGD

mixed results:
- Improvements over some baselines (EN→RO parallel)
- No improvement so far over others (EN→DE with synthetic data)
Deep Models

depth architecture by [Zhou et al., 2016]

- deep recurrent architectures [Zhou et al., 2016, Wu et al., 2016]
- [Zhou et al., 2016] report +4 BLEU from 16 RNN layers: (9 encoder; 7 decoder)
- important trick: residual connections
- challenges: efficiency; memory limitations
Massive Exploration of Neural Machine Translation Architectures [Britz et al., 2017]

- Spent 250,000 hours GPU time exploring hyperparameters
- Conclusions:
  - Small gain from increasing embedding size
  - LSTM better than GRU
  - 2-4 Layer Bidirectional encoder better
  - 4-layer decoder gives some advantage
  - Additive better than multiplicative attention
  - Large beams not helpful (best = 10)
- BLEU variance across runs small (± 0.2-0.3)
sample files and instructions for training NMT model
https://github.com/rsennrich/wmt16-scripts

pre-trained models to test decoding (and for further experiments)
http://statmt.org/rsennrich/wmt16_systems/

lab on installing/using Nematus:
(A small selection of) Resources

## NMT tools

- **Nematus (theano)** [https://github.com/rsennrich/nematus](https://github.com/rsennrich/nematus)
- **OpenNMT (torch)** [https://github.com/OpenNMT/OpenNMT](https://github.com/OpenNMT/OpenNMT)
- **nmt.matlab** [https://github.com/lmthang/nmt.matlab](https://github.com/lmthang/nmt.matlab)
- **neural monkey (tensorflow)** [https://github.com/ufal/neuralmonkey](https://github.com/ufal/neuralmonkey)
- **lamtram (DyNet)** [https://github.com/neubig/lamtram](https://github.com/neubig/lamtram)
- **...and many more** [https://github.com/jonsafari/nmt-list](https://github.com/jonsafari/nmt-list)
Further Reading

**secondary literature**

- chapter on *Neural Network Models* in “Statistical Machine Translation” by Philipp Koehn
- tutorial on sequence-to-sequence models by Graham Neubig

Acknowledgments

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreements 645452 (QT21) and 644402 (HimL).
Thank you!
Several Studies on Natural Language and Back-Propagation.
In IEEE First International Conference on Neural Networks, pages 335–341, San Diego, California, USA.

Neural Machine Translation by Jointly Learning to Align and Translate.

A Neural Probabilistic Language Model.

Neural versus Phrase-Based Machine Translation Quality: a Case Study.
In EMNLP 2016.

Massive Exploration of Neural Machine Translation Architectures.
ArXiv e-prints.

Chen, B., Kuhn, R., Foster, G., Cherry, C., and Huang, F. (2016).
In Proceedings of AMTA.

Natural Language Understanding with Distributed Representation.
CoRR, abs/1511.07916.
Describing Multimedia Content using Attention-based Encoder-Decoder Networks.

ArXiv e-prints.

SYSTRAN’s Pure Neural Machine Translation Systems.
ArXiv e-prints.


A New Algorithm for Data Compression.

A Theoretically Grounded Application of Dropout in Recurrent Neural Networks.
ArXiv e-prints.

The KIT translation systems for IWSLT 2015.

On Using Very Large Target Vocabulary for Neural Machine Translation.

Neural Machine Translation with Characters and Hierarchical Encoding.
CoRR, abs/1610.06550.

In Proceedings of IWSLT.

The AMU-UEDIN Submission to the WMT16 News Translation Task: Attention-based NMT Models as Feature Functions in Phrase-based SMT.

Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing.
Recurrent Continuous Translation Models.

A Convolutional Neural Network for Modelling Sentences.

Sequence-Level Knowledge Distillation.
CoRR, abs/1606.07947.

Fully Character-Level Neural Machine Translation without Explicit Segmentation.
ArXiv e-prints.

Vocabulary Selection Strategies for Neural Machine Translation.
ArXiv e-prints.


Addressing the Rare Word Problem in Neural Machine Translation.
CoRR, abs/1410.8206.
Recurrent neural network based language model.

Neural Reranking Improves Subjective Quality of Machine Translation: NAIST at WAT2015.

Using the Output Embedding to Improve Language Models.


How Grammatical is Character-level Neural Machine Translation? Assessing MT Quality with Contrastive Translation Pairs.

In Proceedings of EACL (Demo Session).

Improving Neural Machine Translation Models with Monolingual Data.


Syntactically Guided Neural Machine Translation.

Sequence to Sequence Learning with Neural Networks.

Decoding with Large-Scale Neural Language Models Improves Translation.
ArXiv e-prints.