Machine Translation Overview

From Austin Matthews (and others)
One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: ‘This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.’

Warren Weaver to Norbert Wiener, March, 1947
NEW ENGLISH TRANSLATION
NOVUM TESTAMENTUM GRAECUM

NEW
TESTAMENT
## CLASSIC SOUPS

<table>
<thead>
<tr>
<th>No.</th>
<th>Soup Name</th>
<th>Sm.</th>
<th>Lg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>57.</td>
<td>House Chicken Soup (Chicken, Celery, Potato, Onion, Carrot)</td>
<td>1.50</td>
<td>2.75</td>
</tr>
<tr>
<td>58.</td>
<td>Chicken Rice Soup</td>
<td>1.85</td>
<td>3.25</td>
</tr>
<tr>
<td>59.</td>
<td>Chicken Noodle Soup</td>
<td>1.85</td>
<td>3.25</td>
</tr>
<tr>
<td>60.</td>
<td>Cantonese Wonton Soup</td>
<td>1.50</td>
<td>2.75</td>
</tr>
<tr>
<td>61.</td>
<td>Tomato Clear Egg Drop Soup</td>
<td>1.65</td>
<td>2.95</td>
</tr>
<tr>
<td>62.</td>
<td>Regular Wonton Soup</td>
<td>1.10</td>
<td>2.10</td>
</tr>
<tr>
<td>63.</td>
<td>Hot &amp; Sour Soup</td>
<td>1.10</td>
<td>2.10</td>
</tr>
<tr>
<td>64.</td>
<td>Egg Drop Soup</td>
<td>1.10</td>
<td>2.10</td>
</tr>
<tr>
<td>65.</td>
<td>Egg Drop Wonton Mix</td>
<td>1.10</td>
<td>2.10</td>
</tr>
<tr>
<td>66.</td>
<td>Tofu Vegetable Soup</td>
<td>NA</td>
<td>3.50</td>
</tr>
<tr>
<td>67.</td>
<td>Chicken Corn Cream Soup</td>
<td>NA</td>
<td>3.50</td>
</tr>
<tr>
<td>68.</td>
<td>Crab Meat Corn Cream Soup</td>
<td>NA</td>
<td>3.50</td>
</tr>
<tr>
<td>69.</td>
<td>Seafood Soup</td>
<td>NA</td>
<td>3.50</td>
</tr>
</tbody>
</table>
Egyptian

Greek
Noisy Channel MT

We want a model of $p(e|f)$
Noisy Channel MT

We want a model of $p(e|f)$

Confusing foreign sentence
Noisy Channel MT

We want a model of $p(e|f)$

Possible English translation

Confusing foreign sentence

Possible English translation
Noisy Channel MT

$p(e)$

“English”

$p(f|e)$

“Foreign”

channel

decode
Noisy Channel MT

\[ \hat{e} = \arg \max_e p(e|f) \]

\[ = \arg \max_e \frac{p(e) \times p(f|e)}{p(f)} \]

\[ = \arg \max_e p(e) \times p(f|e) \]

"Language Model"  "Translation Model"
Noisy Channel Division of Labor

• Language model – $p(e)$
  • is the translation fluent, grammatical, and idiomatic?
  • use any model of $p(e)$ – typically an $n$-gram model

• Translation model – $p(f|e)$
  • “reverse” translation probability
  • ensures adequacy of translation
Translation Model

• $p(f | e)$ gives the channel probability – the probability of translating an English sentence into a foreign sentence

• $f = \text{je voudrais un peu de frommage}$

• $e_1 = \text{I would like some cheese}$

• $e_2 = \text{I would like a little of cheese}$

• $e_3 = \text{There is no train to Barcelona}$

$0.4$  $0.5$  $>0.00001$
Translation Model

• How do we parameterize $p(f|e)$?

$$p(f|e) = \frac{\text{count}(f, e)}{\text{count}(e)}$$

• There are a lot of sentences: this won’t generalize to new inputs
Lexical Translation

• How do we translate a word? Look it up in a dictionary!

  *Haus: house, home, shell, household*

• Multiple translations
  • Different word senses, different registers, different inflections
  • *house, home* are common
  • *shell* is specialized (the Haus of a snail is its shell)
How common is each?

<table>
<thead>
<tr>
<th>Translation</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>house</td>
<td>5000</td>
</tr>
<tr>
<td>home</td>
<td>2000</td>
</tr>
<tr>
<td>shell</td>
<td>100</td>
</tr>
<tr>
<td>household</td>
<td>80</td>
</tr>
</tbody>
</table>
MLE

\( \hat{p}_{\text{MLE}}(e \mid \text{Haus}) = \begin{cases} 
0.696 & \text{if } e = \text{house} \\
0.279 & \text{if } e = \text{home} \\
0.014 & \text{if } e = \text{shell} \\
0.011 & \text{if } e = \text{household} \\
0 & \text{otherwise}
\end{cases} \)
Lexical Translation

• Goal: a model \( p(e|f,m) \)
• where \( e \) and \( f \) are complete English and Foreign sentences

\[
e = \langle e_1, e_2, \ldots, e_m \rangle \quad f = \langle f_1, f_2, \ldots, f_n \rangle
\]
Lexical Translation

• Goal: a model $p(e|f,m)$
• where $e$ and $f$ are complete English and Foreign sentences
• Lexical translation makes the following **assumptions**:
  • Each word $e_i$ in $e$ is generated from exactly one word in $f$
  • Thus, we have a latent *alignment* $a_i$ that indicates which word $e_i$ “came from.” Specifically it came from $f_{a_i}$.
  • Given the alignments $a$, translation decisions are conditionally independent of each other and depend *only* on the aligned source word $f_{a_i}$. 
Lexical Translation

• Putting our assumptions together, we have:

\[ p(e | f, m) = \sum_{a \in [0, n]^m} p(a | f, m) \times \prod_{i=1}^{m} p(e_i | f_{a_i}) \]
Alignment

\[ p(a \mid f, m) \]

- Most of the action for the first 10 years of SMT was here. Words weren’t the problem. Word \textit{order} was hard.
Alignment

• Alignments can be visualized by drawing links between two sentences, and they are represented as vectors of positions:

\[ a = (1, 2, 3, 4)^\top \]
Reordering

• Words may be reordered during translation

\[ a = (3, 4, 2, 1)^\top \]
Word Dropping

• A source word may not be translated at all

\[
a = (2, 3, 4)^\top
\]
Word Insertion

• Words may be inserted during translation
• E.g. English just does not have an equivalent
• But these words must be explained – we typically assume every source sentence contains a NULL token

![Graph showing word insertion]

\[
a = (1, 2, 3, 0, 4)^T
\]
One-to-many Translation

- A source word may translate into more than one target word

\[ a = (1, 2, 3, 4, 4) \]
Many-to-one Translation

- More than one source word may **not** translate as a unit in lexical translation

\[ a = ??? \quad a = (1, 2, (3, 4)^\top)^\top \]
IBM Model 1

• Simplest possible lexical translation model
• Additional assumptions:
  • The alignment decisions are independent
  • The alignment distribution for each $a_i$ is uniform over all source words and NULL

for each $i \in [1, 2, \ldots, m]$

\[ a_i \sim \text{Uniform}(0, 1, 2, \ldots, n) \]
\[ e_i \sim \text{Categorical}(\theta_{f_{a_i}}) \]
Translating with Model 1

0 1 2 3 4
NULL das Haus ist klein
Translating with Model 1

Language model says: ☑
Translating with Model 1

Language model says: ☑
Learning Lexical Translation Models

• How do we learn the parameters $p(e|f)$?
• “Chicken and egg” problem
  • If we had the alignments, we could estimate the translation probabilities (MLE estimation)
  • If we had the translation probabilities we could find the most likely alignments (greedy)
EM Algorithm

• Pick some random (or uniform) starting parameters
• Repeat until bored (~5 iterations for lexical translation models):
  • Using the current parameters, compute “expected” alignments \( p(a_i|e, f) \) for every target word token in the training data
  • Keep track of the expected number of times \( f \) translates into \( e \) throughout the whole corpus
  • Keep track of the number of times \( f \) is used in the source of any translation
  • Use these estimates in the standard MLE equation to get a better set of parameters
EM for Model 1

... la maison ... la maison blue ... la fleur ...

... the house ... the blue house ... the flower ...

- Initial step: all alignments equally likely
- Model learns that, e.g., la is often aligned with the
EM for Model 1

... la maison ... la maison blue ... la fleur ...

... the house ... the blue house ... the flower ...

- After one iteration
- Alignments, e.g., between la and the are more likely
EM for Model 1

... la maison ... la maison bleu ... la fleur ...

... the house ... the blue house ... the flower ...

• After another iteration

• It becomes apparent that alignments, e.g., between fleur and flower are more likely (pigeon hole principle)
EM for Model 1

... la maison ... la maison bleu ... la fleur ...

... the house ... the blue house ... the flower ...

\[
p(\text{la} | \text{the}) = 0.453 \\
p(\text{le} | \text{the}) = 0.334 \\
p(\text{maison} | \text{house}) = 0.876 \\
p(\text{bleu} | \text{blue}) = 0.563 \\
\]

- Parameter estimation from the aligned corpus
Convergence

<table>
<thead>
<tr>
<th>e</th>
<th>f</th>
<th>initial</th>
<th>1st it.</th>
<th>2nd it.</th>
<th>3rd it.</th>
<th>...</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>das</td>
<td>0.25</td>
<td>0.5</td>
<td>0.6364</td>
<td>0.7479</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>book</td>
<td>das</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1208</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>house</td>
<td>das</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1313</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>the</td>
<td>buch</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1208</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>book</td>
<td>buch</td>
<td>0.25</td>
<td>0.5</td>
<td>0.6364</td>
<td>0.7479</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>buch</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1313</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>book</td>
<td>ein</td>
<td>0.25</td>
<td>0.5</td>
<td>0.4286</td>
<td>0.3466</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>ein</td>
<td>0.25</td>
<td>0.5</td>
<td>0.5714</td>
<td>0.6534</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>haus</td>
<td>0.25</td>
<td>0.5</td>
<td>0.4286</td>
<td>0.3466</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>house</td>
<td>haus</td>
<td>0.25</td>
<td>0.5</td>
<td>0.5714</td>
<td>0.6534</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>
Extensions

- Phrase-based MT:
  - Allow multiple words to translate as chunks (including many-to-one)
  - Introduce another latent variable, the source *segmentation*

```
Maria no dio una bofetada a la bruja verde
```

```
Mary not give a slap to the witch green
did not a slap by hag bawdy
no slap to the green witch
did not give the
the witch
```

Adapted from Koehn (2006)
Extensions

• Alignment Priors:
  • Instead of assuming the alignment decisions are uniform, impose (or learn) a prior over alignment grids:

Chahuneau et al. (2013)
Extensions

• Syntactic structure
• Rules of the form:
• \( X \text{ 之一} \rightarrow \text{one of the } X \)

Chang (2005), Galley et al. (2006)
Evaluation

• How do we evaluate translation systems’ output?
• Central idea: “The closer a machine translation is to a professional human translation, the better it is.”
• Most commonly used metric is called BLEU
Candidate 1: *It is a guide to action which ensures that the military always obey the commands of the party.*

Reference 1: *It is a guide to action that ensures that the military will forever heed Party commands.*

Reference 2: *It is the guiding principle which guarantees the military forces always being under the command of the Party.*

Reference 3: *It is the practical guide for the army always to heed directions of the party.*

Unigram Precision : 17/18
Issue of N-gram Precision

• What if some words are over-generated?
  • e.g. “the”
• An extreme example

Candidate: the the the the the the the.
Reference 1: The cat is on the mat.
Reference 2: There is a cat on the mat.

• N-gram Precision: 7/7
• Solution: reference word should be exhausted after it is matched.

Adapted from slides by Arthur Chan
Issue of N-gram Precision

• What if some words are just dropped?
• Another extreme example

Candidate: *the*.
Reference 1: *My mom likes the blue flowers.*
Reference 2: *My mother prefers the blue flowers.*

• N-gram Precision: 1/1
• Solution: add a penalty if the candidate is too short.
BLEU

\[
\text{BLEU} = \left( p_1 \cdot p_2 \cdot p_3 \cdot p_4 \right)^{\frac{1}{4}} \max(1, e^{1 - \frac{r}{c}})
\]

- Ranges from 0.0 to 1.0, but usually shown multiplied by 100
- An increase of +1.0 BLEU is usually a conference paper
- MT systems usually score in the 10s to 30s
- Human translators usually score in the 70s and 80s
A Short Segue

- Word- and phrase-based ("symbolic") models were cutting edge for decades (up until ~2014)
  - Such models are still the most widely used in commercial applications
- Since 2014 most research on MT has focused on neural models
“Neurons”
“Neurons”
“Neurons”

\[ y = x \cdot w \]
“Neurons”

\[ y = g(x \cdot w) \]
“Neurons”
“Neural” Networks
“Neural” Networks
“Neural” Networks

\[ y = x^T W \]
“Neural” Networks

\[ y = x^T W \]

\[ y = g(x^T W) \]
“Neural” Networks

\[ y = x^T W \]

\[ y = g(x^T W) \]

\[ g(u)_i = \frac{\exp u_i}{\sum_{i'} \exp u_{i'}} \]

“Soft max”
“Deep”
“Deep”
“Deep”
"Deep"

\[ z = g(y^T V) \]
"Deep"

\[ z = g(y^T V) \]

\[ z = g(h(x^T W)^T V) \]
"Deep"

\[ z = g(y^T V) \]
\[ z = g(h(x^T W)^T V) \]
\[ z = g(V h(W x)) \]
Deep

\[ z = g(y^T V) \]
\[ z = g(h(x^T W)^T V) \]
\[ z = g(V h(W x)) \]

Note:

\[ \text{if } g(x) = h(x) = x \]
\[ z = \underbrace{V W}_U x \]
“Recurrent”
Design Decisions

• How to represent inputs and outputs?
• Neural architecture?
  • How many layers? (Requires non-linearities to improve capacity!)
  • How many neurons?
  • Recurrent or not?
  • What kind of non-linearities?
Representing Language

- “One-hot” vectors
  - Each position in a vector corresponds to a word type

  
<table>
<thead>
<tr>
<th>Aardvark</th>
<th>Abandon</th>
<th>Abandon</th>
<th>Aban...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

  dog = \langle 0.79995, 0.67263, 0.73924, 0.77496, 0.09286, 0.802798, 0.35508, 0.44789 \rangle

- Distributed representations
  - Vectors encode “features” of input words (character n-grams, morphological features, etc.)
Training Neural Networks

• Neural networks are supervised models – you need a set of inputs paired with outputs

• Algorithm
  • Run until bored:
    • Give input to the network, see what it predicts
    • Compute loss($y, y^*$)
    • Use chain rule (aka “back propagation”) to compute gradient with respect to parameters
    • Update parameters (SGD, Adam, LBFGS, etc.)
Neural Language Models

\[ p(e) = \prod_{i=1}^{\left| e \right|} p(e_i \mid e_{i-n+1}, \ldots, e_{i-1}) \]

\[ p(e_i \mid e_{i-n+1}, \ldots, e_{i-1}) = \]

\[ \begin{align*}
& e_{i-1} & C & W & V & e_i \\
& e_{i-2} & C & \text{tanh} & \text{softmax} & \\
& e_{i-3} & C & \end{align*} \]
Bengio et al. (2003)

<table>
<thead>
<tr>
<th>Model</th>
<th>n</th>
<th>c</th>
<th>h</th>
<th>m</th>
<th>direct</th>
<th>mix</th>
<th>train.</th>
<th>valid.</th>
<th>test.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP1</td>
<td>5</td>
<td>50</td>
<td>60</td>
<td>yes</td>
<td>no</td>
<td></td>
<td>182</td>
<td>284</td>
<td>268</td>
</tr>
<tr>
<td>MLP2</td>
<td>5</td>
<td>50</td>
<td>60</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>275</td>
<td>257</td>
<td></td>
</tr>
<tr>
<td>MLP3</td>
<td>5</td>
<td>0</td>
<td>60</td>
<td>yes</td>
<td>no</td>
<td></td>
<td>201</td>
<td>327</td>
<td>310</td>
</tr>
<tr>
<td>MLP4</td>
<td>5</td>
<td>0</td>
<td>60</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>286</td>
<td>272</td>
<td></td>
</tr>
<tr>
<td>MLP5</td>
<td>5</td>
<td>50</td>
<td>30</td>
<td>yes</td>
<td>no</td>
<td></td>
<td>209</td>
<td>296</td>
<td>279</td>
</tr>
<tr>
<td>MLP6</td>
<td>5</td>
<td>50</td>
<td>30</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>273</td>
<td>259</td>
<td></td>
</tr>
<tr>
<td>MLP7</td>
<td>3</td>
<td>50</td>
<td>30</td>
<td>yes</td>
<td>no</td>
<td></td>
<td>210</td>
<td>309</td>
<td>293</td>
</tr>
<tr>
<td>MLP8</td>
<td>3</td>
<td>50</td>
<td>30</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>284</td>
<td>270</td>
<td></td>
</tr>
<tr>
<td>MLP9</td>
<td>5</td>
<td>100</td>
<td>30</td>
<td>no</td>
<td>no</td>
<td></td>
<td>175</td>
<td>280</td>
<td>276</td>
</tr>
<tr>
<td>MLP10</td>
<td>5</td>
<td>100</td>
<td>30</td>
<td>no</td>
<td>yes</td>
<td></td>
<td>265</td>
<td>252</td>
<td></td>
</tr>
<tr>
<td>Del. Int.</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>31</td>
<td>352</td>
<td>336</td>
</tr>
<tr>
<td>Kneser-Ney back-off</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>334</td>
<td>323</td>
<td></td>
</tr>
<tr>
<td>Kneser-Ney back-off</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>332</td>
<td>321</td>
<td></td>
</tr>
<tr>
<td>Kneser-Ney back-off</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>332</td>
<td>321</td>
<td></td>
</tr>
</tbody>
</table>
Neural Features for Translation

• Turn Bengio et al. (2003) into a translation model
• Conditional model, generate the next English word conditioned on
  • The previous $n$ English words you generated
  • The aligned source word and its $m$ neighbors
\[ p(e \mid f, a) = \prod_{i=1}^{\mid e \mid} p(e_i \mid e_{i-2}, e_{i-1}, f_{a_i-1}, f_{a_i}, f_{a_i+1}) \]

\[ p(e_i \mid e_{i-2}, e_{i-1}, f_{a_i-1}, f_{a_i}, f_{a_i+1}) = \]

Devlin et al. (2014)
Neural Features for Translation

<table>
<thead>
<tr>
<th>BOLT Test</th>
<th>Ar-En</th>
<th>BLEU</th>
<th>% Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Simple Hier.” Baseline</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2T/L2R NNJM (Dec)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Window=7</td>
<td>38.3</td>
<td>98%</td>
<td></td>
</tr>
<tr>
<td>Source Window=5</td>
<td>38.2</td>
<td>96%</td>
<td></td>
</tr>
<tr>
<td>Source Window=3</td>
<td>37.8</td>
<td>87%</td>
<td></td>
</tr>
<tr>
<td>Source Window=0</td>
<td>35.3</td>
<td>33%</td>
<td></td>
</tr>
<tr>
<td>Layers=384x768x768</td>
<td>38.5</td>
<td>102%</td>
<td></td>
</tr>
<tr>
<td>Layers=192x512</td>
<td>38.1</td>
<td>93%</td>
<td></td>
</tr>
<tr>
<td>Layers=128x128</td>
<td>37.1</td>
<td>72%</td>
<td></td>
</tr>
<tr>
<td>Vocab=64,000</td>
<td>38.5</td>
<td>102%</td>
<td></td>
</tr>
<tr>
<td>Vocab=16,000</td>
<td>38.1</td>
<td>93%</td>
<td></td>
</tr>
<tr>
<td>Vocab=8,000</td>
<td>37.3</td>
<td>83%</td>
<td></td>
</tr>
<tr>
<td>Activation=Rectified Lin.</td>
<td>38.5</td>
<td>102%</td>
<td></td>
</tr>
<tr>
<td>Activation=Linear</td>
<td>37.3</td>
<td>76%</td>
<td></td>
</tr>
</tbody>
</table>

Devlin et al. (2014)
Notation Simplification
RNNs Revisited
Fully Neural Translation

• Fully end-to-end RNN-based translation model
• Encode the source sentence using one RNN
• Generate the target sentence one word at a time using another RNN

Sutskever et al. (2014)
Attentional Model

• The encoder-decoder model struggles with long sentences
• An RNN is trying to compress an arbitrarily long sentence into a finite-length worth vector
• What if we only look at one (or a few) source words when we generate each output word?

Bahdanau et al. (2014)
Our large black dog bit the poor mailman.

うちの大きな黒い犬が可哀想な郵便屋に噛みついた。
The Attention Model

Encoder

Decoder

Bahdanau et al. (2014)
The Attention Model

Encoder

I am a student</s>

Attention Model

Decoder

Bahdanau et al. (2014)
The Attention Model

Encoder

Decoder

Bahdanau et al. (2014)
The Attention Model

Encoder

Attention Model

Decoder

Context Vector

Bahdanau et al. (2014)

I am a student
The Attention Model

Attention Model

Encoder

Decoder

Context Vector

Bahdanau et al. (2014)
The Attention Model

Bahdanau et al. (2014)
The Attention Model

Bahdanau et al. (2014)
The Attention Model

Encoder

I am a student

Context Vector

Attention Model

Decoder

je suis

je

Bahdanau et al. (2014)
The Attention Model

Encoder

Decoder

Attention Model

Context Vector

Bahdanau et al. (2014)
The Attention Model

Encoder

Decoder

Bahdanau et al. (2014)
The Attention Model

Encoder

I am a student

Attention Model

Context Vector

Decoder

je suis étudiant

Bahdanau et al. (2014)
The Transformer

• Idea: Instead of using an RNN to encode the source sentence and the partial target sentence, use self-attention!

Vaswani et al. (2017)
The Transformer

Encoder

I am a student

Attention Model

Decoder

je suis étudiant

Context Vector

Vaswani et al. (2017)
The Transformer

- Computation is easily parallelizable
- Shorter path from each target word to each source word → stronger gradient signals
- Empirically stronger translation performance
- Empirically trains substantially faster than more serial models

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Training Cost (FLOPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EN-DE</td>
<td>EN-FR</td>
</tr>
<tr>
<td>ByteNet [17]</td>
<td>23.75</td>
<td></td>
</tr>
<tr>
<td>Deep-Att + PosUnk [37]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNMT + RL [36]</td>
<td>24.6</td>
<td>2.3 \cdot 10^{19}</td>
</tr>
<tr>
<td>ConvS2S [9]</td>
<td>25.16</td>
<td>9.6 \cdot 10^{18}</td>
</tr>
<tr>
<td>MoE [31]</td>
<td>26.03</td>
<td>2.0 \cdot 10^{19}</td>
</tr>
<tr>
<td>Deep-Att + PosUnk Ensemble [37]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNMT + RL Ensemble [36]</td>
<td>26.30</td>
<td>1.8 \cdot 10^{20}</td>
</tr>
<tr>
<td>ConvS2S Ensemble [9]</td>
<td>26.36</td>
<td>7.7 \cdot 10^{19}</td>
</tr>
<tr>
<td>Transformer (base model)</td>
<td>27.3</td>
<td>3.3 \cdot 10^{18}</td>
</tr>
<tr>
<td>Transformer (big)</td>
<td>28.4</td>
<td>2.3 \cdot 10^{19}</td>
</tr>
</tbody>
</table>
Current Research Directions on Neural MT

• Incorporation syntax into Neural MT
• Handling of morphologically rich languages
• Optimizing translation quality (instead of corpus probability)
• Multilingual models
• Document-level translation