Google's Neural Machine Translation System:

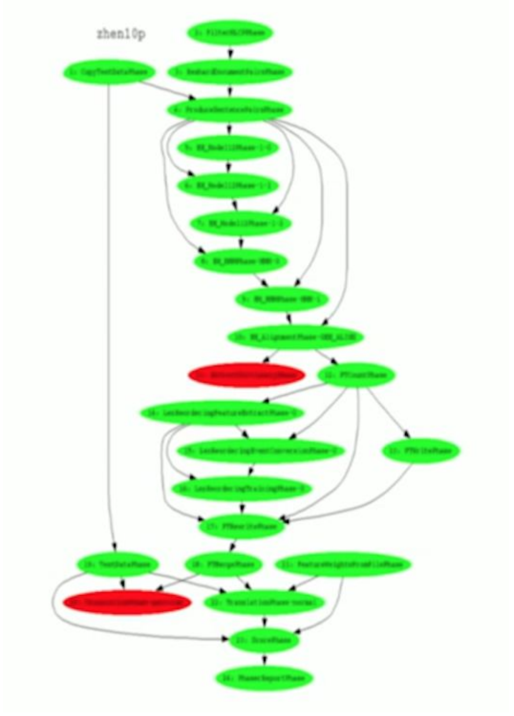
Bridging the Gap between Human and Machine Translation

October 2016

Daria Walter
November 13, 2017

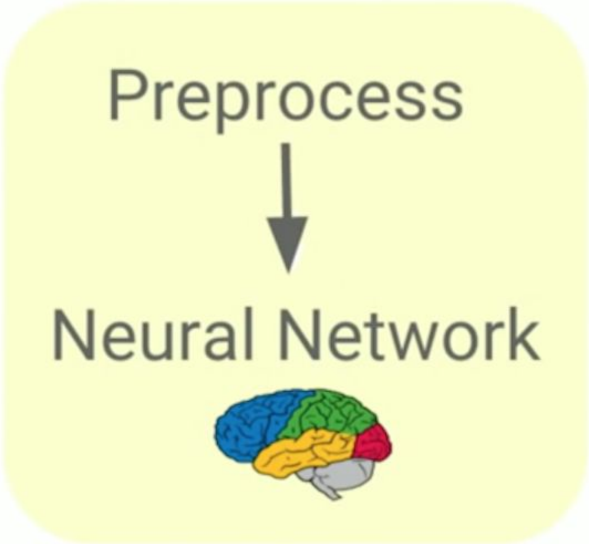
Old: phrase-based translation

- Lots of individual pieces
- Optimized somewhat independently

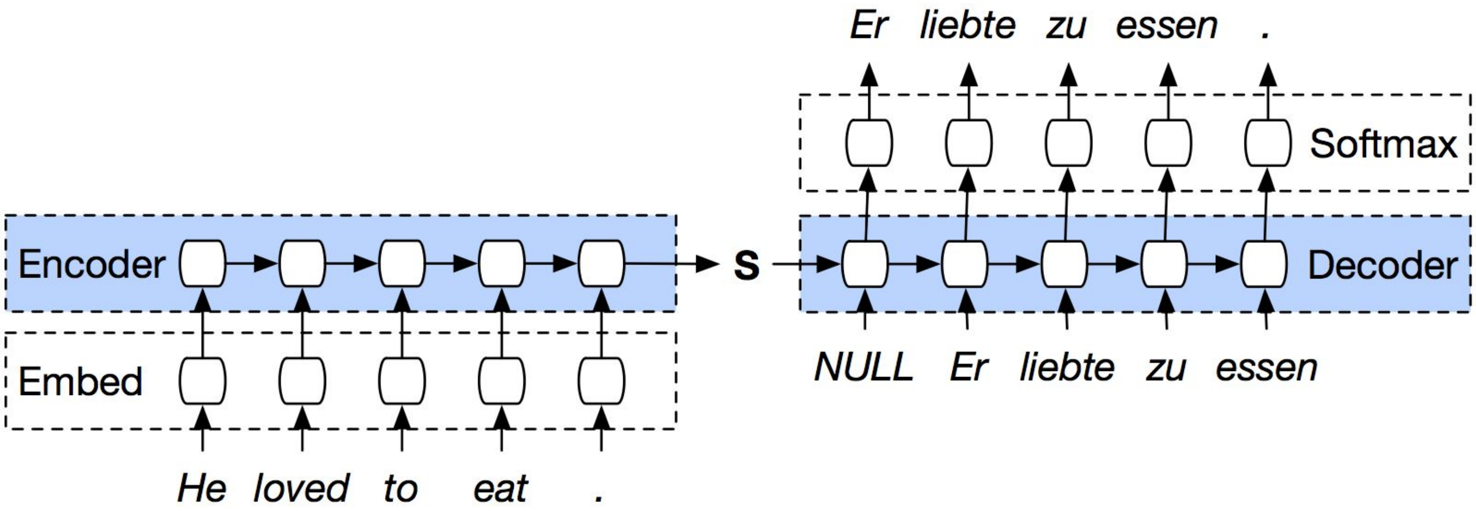


New: Neural Machine translation

- End-to-end learning
- Simpler architecture
- Results are much better (-60% translation errors on production data)



Neural Machine translation: general approach



- Encoder + decoder RNNs (LSTMs)
- Encoder maps the source sequence to the hidden vector **S.**
- Given **S**, decoder predicts output sequence, using Softmax activation.

Neural Machine translation: closer look

$$p(y_1, \dots, y_J | x_1, \dots, x_I) = \prod_{j=1}^{J} p(y_j | S, y_1, \dots, y_{j-1}) \to \max$$

$$\text{Encoder LSTM}: \quad s_i = f(x_i, s_{i-1})$$

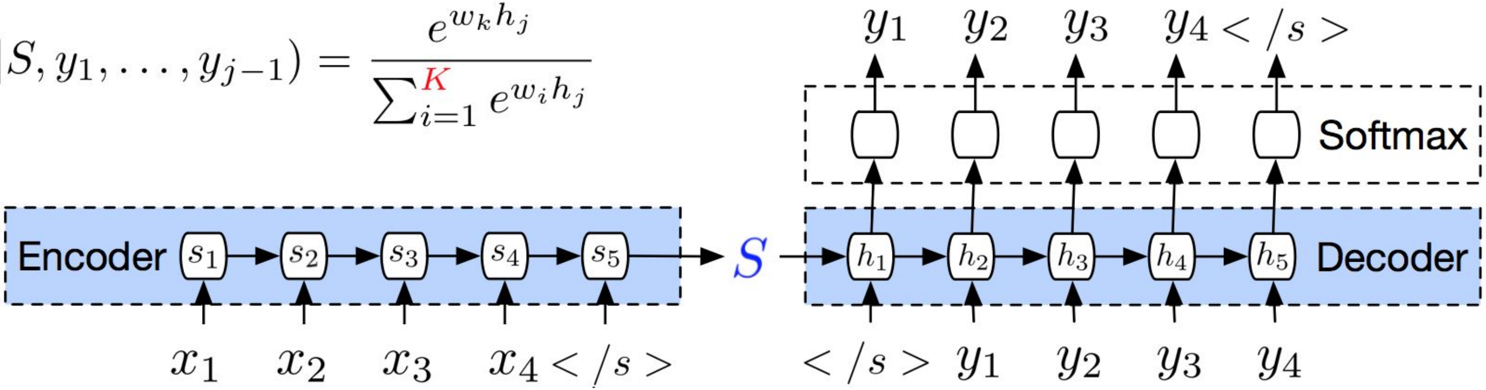
$$S = s_I - \text{context vector}$$

$$\text{Decoder LSTM}: \quad h_j = g(h_{j-1}, S, y_{j-1}])$$

$$K - \text{vocabulary size}$$

$$\text{Softmax prediction}:$$

$$p(y_j = k | S, y_1, \dots, y_{j-1}) = \frac{e^{w_k h_j}}{\sum_{i=1}^{K} e^{w_i h_j}}$$



Standart Neural Machine translation

+ End-to-end approach
+ Universal for different language pairs
+ Robust for languages with different base word order

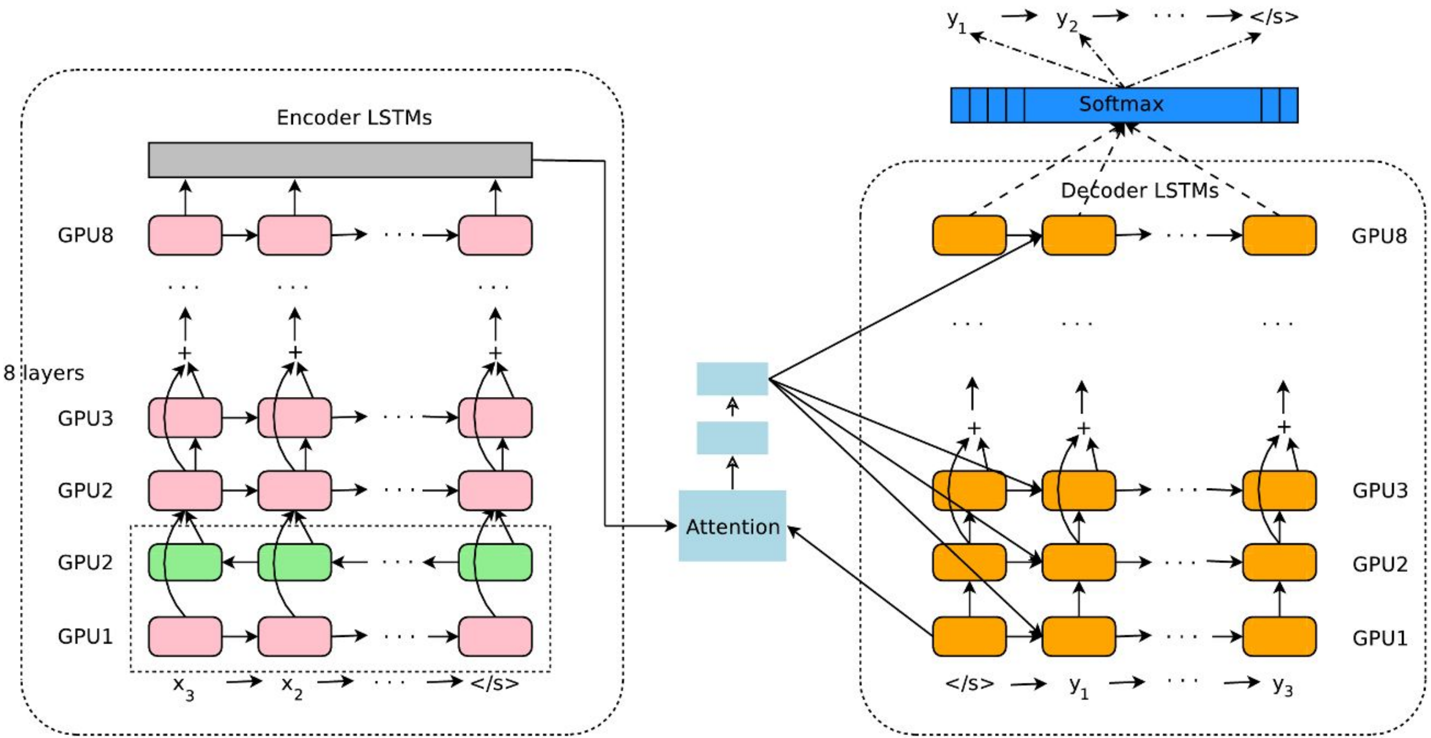
- Inefficiency of standart RNN architectures
- Computationally expensive during training
- Slow inference speed
- Fixed-size vocabulary: how to deal with rare words?
- Bad coverage

How GNMT addresses these problems:

Model Architecture

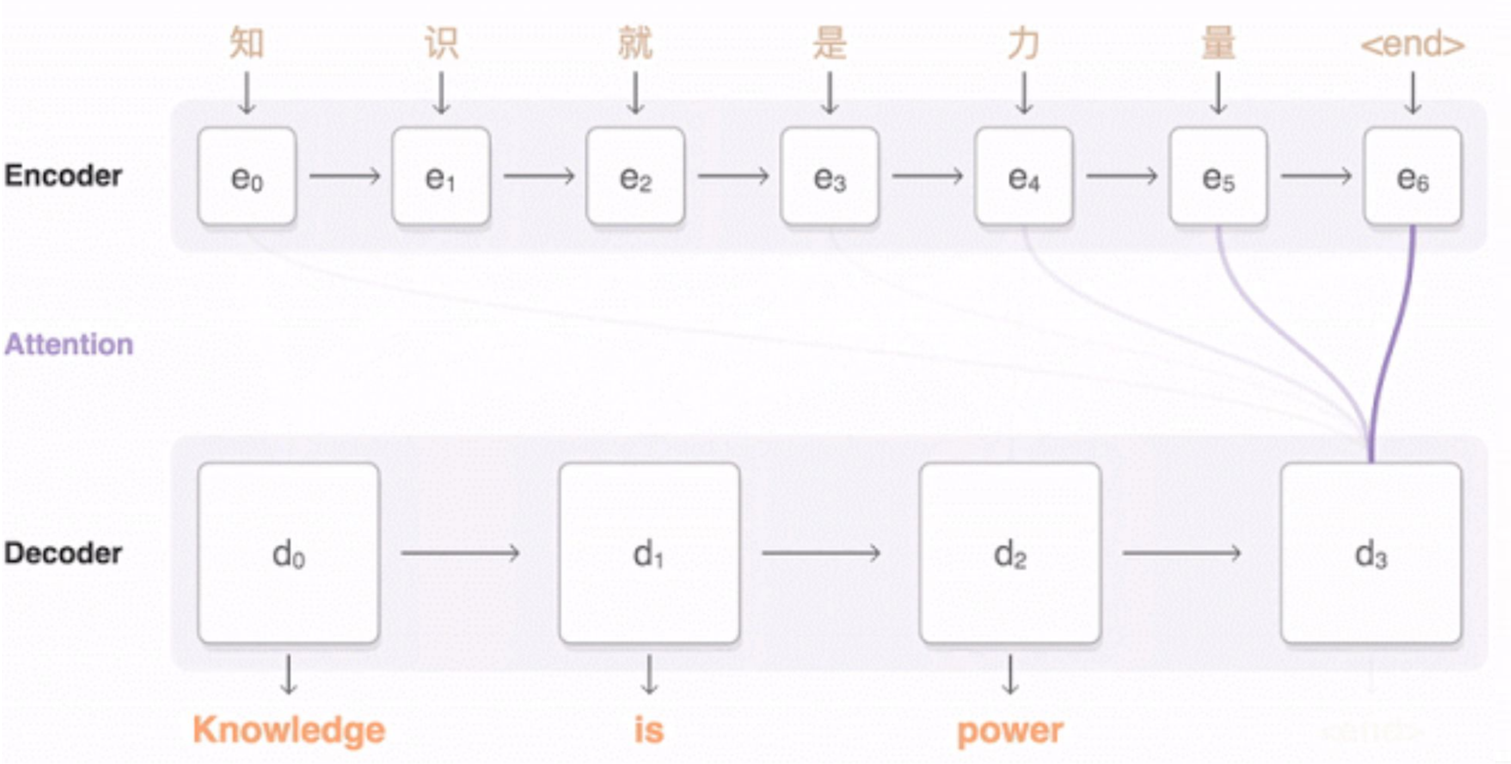
Deep Sequence 2 Sequence model

- 8 layers
- Bi-directional bottom layer
- Attention mechanism
- Residual connections
- Parallelism: 8 GPU

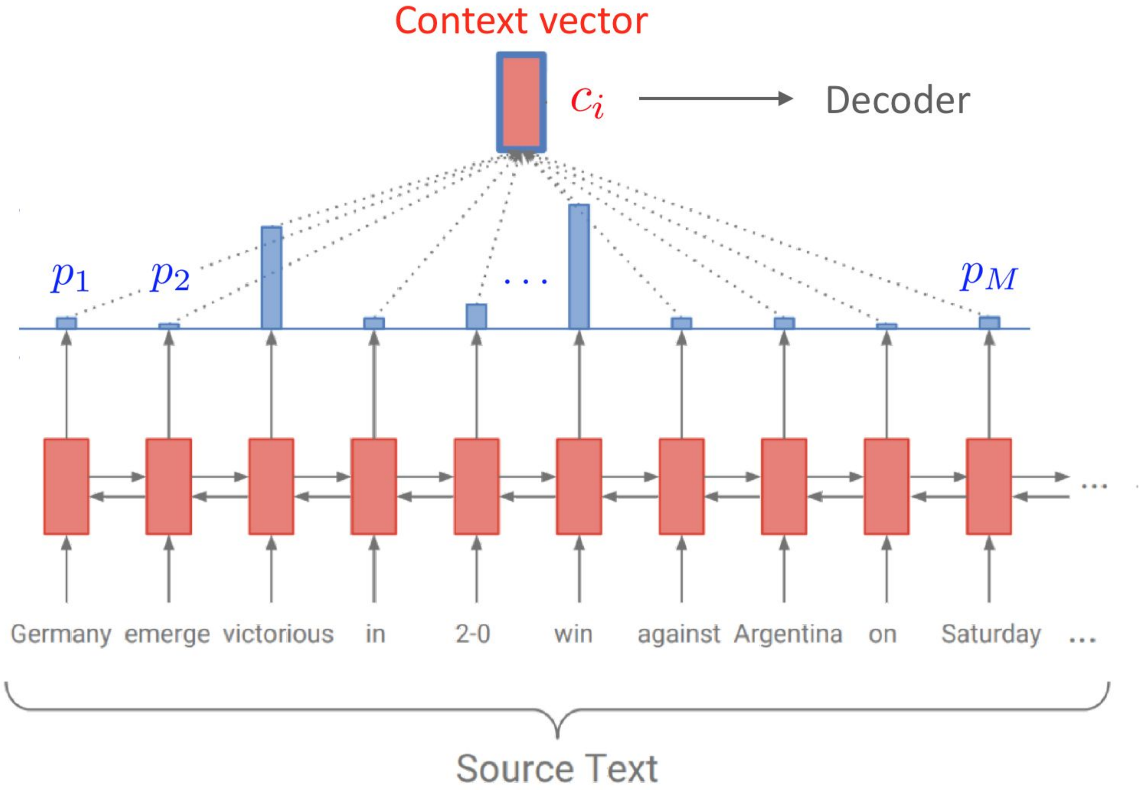


Attention mechanism

Idea: for a new element in decoder, assign “significance” weights to each encoded element



Attention mechanism: Decoder *i-th* step



$s_1, \dots, s_M$ – encoder output

h_{i-1} – decoder output from $i-1$ step

$a_t = AttentionFunction(h_{i-1}, s_t)$

$AttentionFunction$ - single-layer NN

$$p_t = \frac{exp(a_t)}{\sum_{t=1}^{M} exp(a_t)}$$

$$c_i = \sum_{t=1}^{M} p_t s_t$$

Decoder new hidden state:

$$h_i = g(h_{i-1}, [y_{i-1}, c_i])$$

Towards open vocabulary:

How GNMT deals with rare words

First approach: wordpiece model (WPM)

Example:

- **Word:** `Jet makers feud over seat width with big orders at stake`
- **Wordpiece:** **`_J et`** `_makers` **`_fe ud`** `_over _seat _width _with _big _orders _at _stake`

Given:

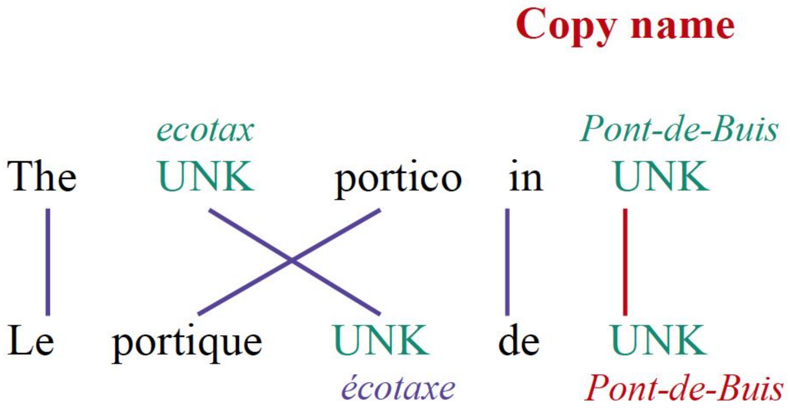
1) training corpus
2) desired number **D** of wordpieces

Select **D** unique wordpieces, so that the total number of wordpieces in resulting corpus, segmented according to this WPM, was minimal.

- **D is between 8k and 32k**

Translation of names, numbers, rare entities

- Sometimes it makes sense to copy rare entities from **source** to **target**:



- To support direct copy mechanism, use shared wordpiece model for source and target language.
- Same string in source and target language will be segmented in the same way -> model learns to copy.

Second approach: mixed word/character model

- Vocabulary: words + characters
- Out-of-vocabulary words are splitted into characters before encoding.
- Special prefixes are prepended to single characters:

    <B> - `beginning of the word`

    <M> - `middle of the word`

    <E> - `end of the word`

- Example:

    `Miki    -->    <B>M <M>i <M>k <E>i`

Training

Maximum-likelihood training criteria

Parallel training dataset: $\mathcal{D} = \{(X^{(i)}, Y^{*(i)})\}$

$$\mathcal{O}_{ML}(\Theta) = \sum_{i=1}^{N} \log P_{\Theta}(Y^{*(i)} | X^{(i)}) \rightarrow \max_{\Theta}$$

Drawbacks of this training criteria:

- Incorrect output sequences are never observed during training
- Does not incorporate BLEU score (main quality metrics in translation)

Idea: explicitly rank all possible output sequences by BLEU score

Translation quality: BLEU score

- N-gram overlap between machine translation output and reference translation
- Compute precision for n-grams of size 1 to 4
- Add brevity penalty (for too short translations)

$$\mathrm{BLEU} = \min(1, \frac{\text{output length}}{\text{reference length}})(\prod_{i=1}^{4} precision_i)^{1/4} \times 100$$

- Typically computed over the entire corpus, not single sentences

---

- The authors use corrected GLEU score, better on single sentences:

$$\mathrm{GLEU} = \min(1, \frac{\text{output length}}{\text{reference length}})(\prod_{i=1}^{4} min(precision_i, recall_i))^{1/4}$$

Model refinement using expected reward objective

$r(Y, Y^{*(i)})$ – per-sentence GLEU score

- Expected reward objective:

$$\mathcal{O}_{RL}(\Theta) = \sum_{i=1}^{N} \sum_{Y \in \mathcal{Y}} P_{\Theta}(Y|X^{(i)}) r(Y, Y^{*(i)})$$

- Optimize linear combination of **ML** and **RL** objectives:

$$\mathcal{O}_{Mixed}(\Theta) = \alpha \mathcal{O}_{ML}(\Theta) + \mathcal{O}_{RL}(\Theta)$$

$$\alpha = 0.017$$

Prediction

Generating predictions

- Maximize probability of the total output sequence:

$$P(Y|X) = P(Y|\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_M})$$
$$= \prod_{i=1}^{N} P(y_i|y_0, y_1, y_2, ..., y_{i-1}; \mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_M})$$

$$\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_M} = EncoderRNN(x_1, x_2, x_3, ..., x_M)$$

- **Need to use beam search to select best prediction**

Beam search refinements

Empirically designed penalty functions:

- **coverage penalty**

$$cp(X,Y) = \beta * \sum_{i=1}^{|X|} \log(\min(\sum_{j=1}^{|Y|} p_{i,j}, 1.0))$$

- **lenght normalization**

$$lp(Y) = \frac{(5+|Y|)^{\alpha}}{(5+1)^{\alpha}}$$

- Maximize scoring function with beam search:

$$s(Y,X) = \frac{\log(P(Y|X))}{lp(Y)} + cp(X,Y)$$

Results

Comparison against state-of-the-art NMT

Table 4: Single model results on WMT En→Fr (newstest2014)

Model	BLEU	CPU decoding time per sentence (s)
Word	37.90	0.2226
Character	38.01	1.0530
WPM-8K	38.27	0.1919
WPM-16K	37.60	0.1874
WPM-32K	38.95	0.2118
Mixed Word/Character	38.39	0.2774
PBMT [15]	37.0	
LSTM (6 layers) [31]	31.5	
LSTM (6 layers + PosUnk) [31]	33.1	
Deep-Att [45]	37.7	
* Deep-Att + PosUnk [45]	39.2	

Table 5: Single model results on WMT En→De (newstest2014)

Model	BLEU	CPU decoding time per sentence (s)
Word	23.12	0.2972
Character (512 nodes)	22.62	0.8011
WPM-8K	23.50	0.2079
WPM-16K	24.36	0.1931
WPM-32K	24.61	0.1882
Mixed Word/Character	24.17	0.3268
PBMT [6]	20.7	
RNNSearch [37]	16.5	
RNNSearch-LV [37]	16.9	
RNNSearch-LV [37]	16.9	
Deep-Att [45]	20.6	

Table 6: Single model test BLEU scores, averaged over 8 runs, on WMT En→Fr and En→De

Dataset	Trained with log-likelihood	Refined with RL
En→Fr	38.95	39.92
En→De	24.67	24.60

* Use external alignment models

Model ensemble

Model ensemble results on WMT En→Fr (newstest2014)

Model	BLEU
WPM-32K (8 models)	40.35
RL-refined WPM-32K (8 models)	41.16
LSTM (6 layers) [31]	35.6
LSTM (6 layers + PosUnk) [31]	37.5
Deep-Att + PosUnk (8 models) [45]	40.4

Model ensemble results on WMT En→De

Model	BLEU
WPM-32K (8 models)	26.20
RL-refined WPM-32K (8 models)	26.30

- Ensemble 8 RL-refined models
- **BLEU scores are better (+ 2), compared to single-model**

Results on production data: human evaluation

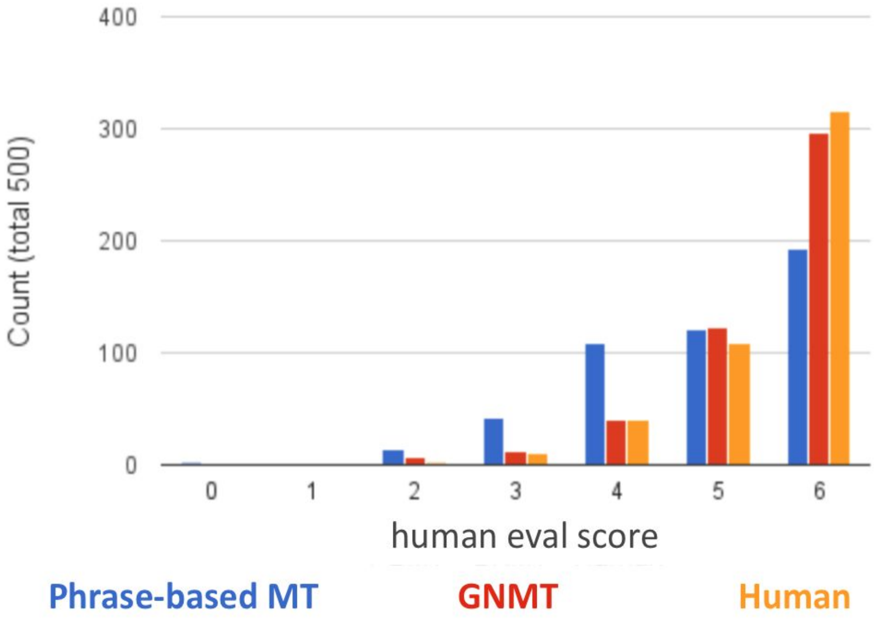


Table 10: Mean of side-by-side scores on production data

	PBMT	GNMT	Human	Relative Improvement
English → Spanish	4.885	5.428	5.504	87%
English → French	4.932	5.295	5.496	64%
English → Chinese	4.035	4.594	4.987	58%
Spanish → English	4.872	5.187	5.372	63%
French → English	5.046	5.343	5.404	83%
Chinese → English	3.694	4.263	4.636	60%

- **Relative improvement > 60% on major pairs of languages**

References

1) Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi. “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”, 2016 [link]
2) Stanford Seminar: Google's Multilingual Neural Machine Translation System [link]

Questions?