Google's Neural Machine Translation System

Álvaro Peris
lvapeab@prhlt.upv.es

Pattern Recognition and Human Language Technologies
Research Center
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

DSIC UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Presentation Outline

Introduction: Neural Machine Translation 1

Google's Neural Machine Translation System 12

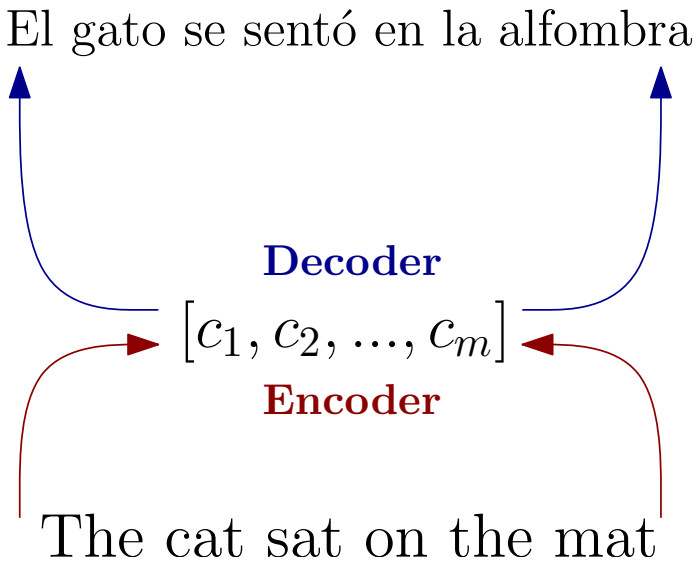
Practical issues 18

Experiments and Results 28

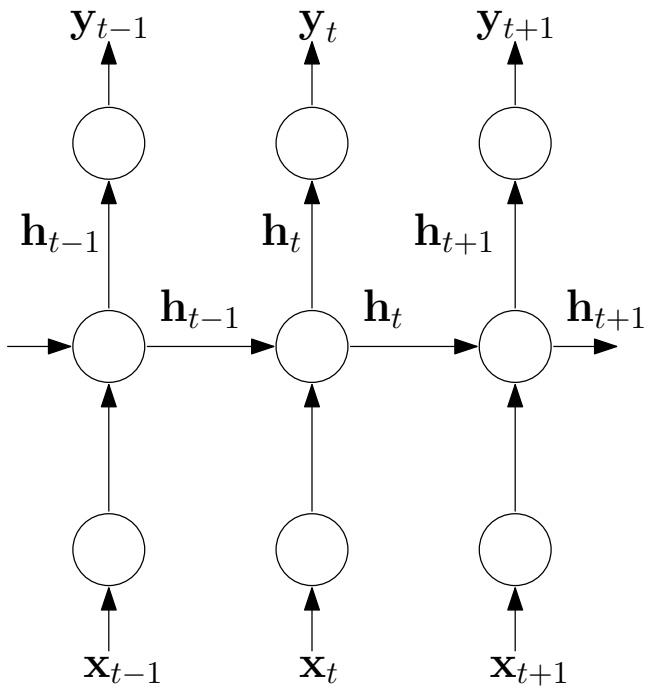
Conclusions 40

Introduction

- Distributed representation of words and sentences.
- Input and output sequences → Recurrent Neural Networks.
- Encoder – Decoder approach.

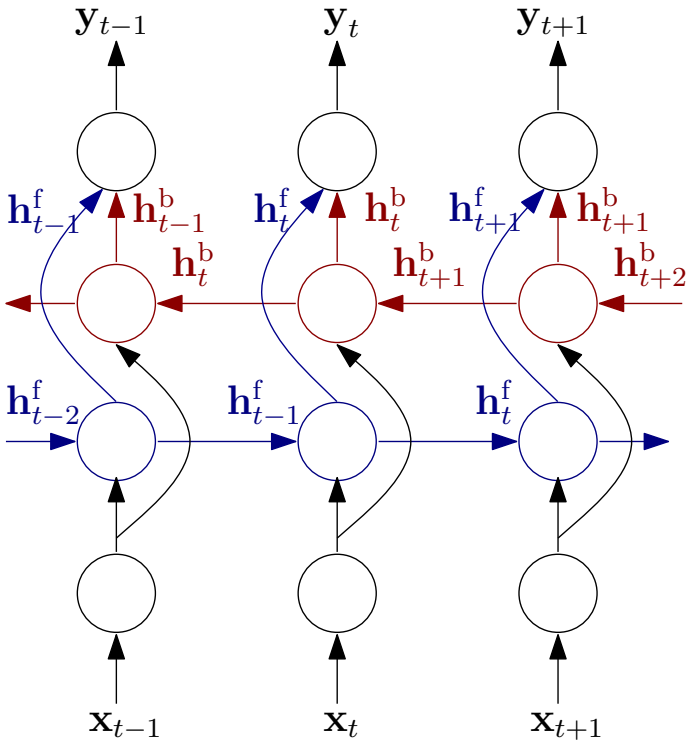


Recurrent Neural Networks



$$\mathbf{h}_t = f_h(\mathbf{x}_t, \mathbf{h}_{t-1})$$
$$\mathbf{y}_t = f_o(\mathbf{h}_t)$$

Bidirectional Recurrent Neural Networks



Output layer

Backward layer

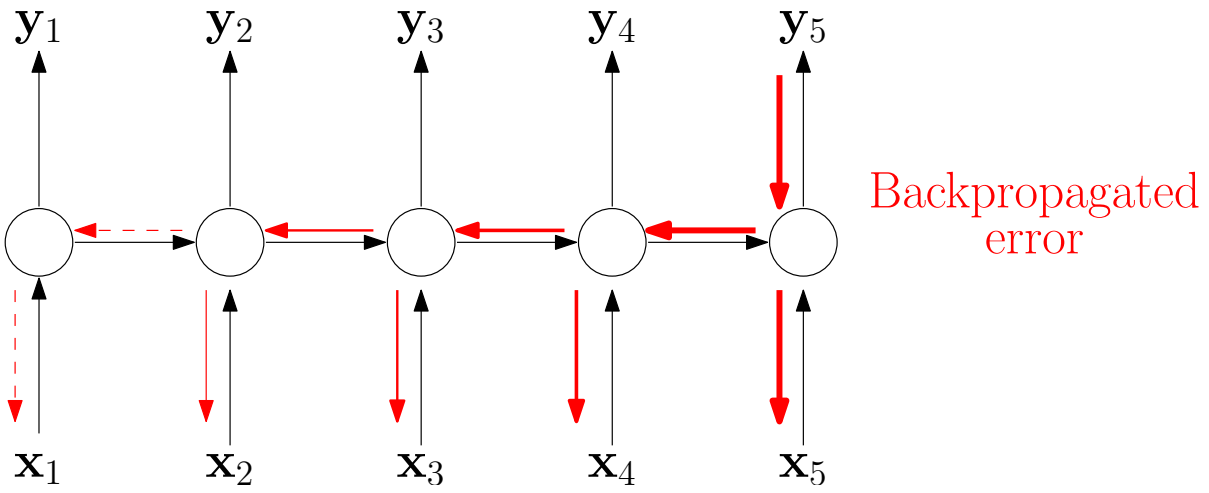
Forward layer

Input layer

$$\mathbf{h}^{\mathrm{f}}_t = f_h(\mathbf{x}_t, \mathbf{h}^{\mathrm{f}}_{t-1})$$
$$\mathbf{h}^{\mathrm{b}}_t = f_h(\mathbf{x}_t, \mathbf{h}^{\mathrm{b}}_{t+1})$$
$$\mathbf{y}_t = f_o(\mathbf{h}^{\mathrm{f}}_t, \mathbf{h}^{\mathrm{b}}_t)$$

RNNs training: Vanishing gradients problem

- Backpropagation through time (BPTT) algorithm.
- Problem: Vanishing gradients.
- If gradients are close to zero and are backpropagated many steps backs, its influence also tends to zero.
- Long-term relationships are lost.



Vanishing gradients problem

- (partial) Solution: Gated units.
- Control how much information...
  1. ...flows from the previous time-steps to the current one .
  2. ...is taken into account from the input in the current time-step.
- Great results in tasks with need of contextual information.
- *Long short-term memory* (LSTM) and *gated recurrent units* (GRU).

Long Short-Term Memory (LSTM) units

$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t$$

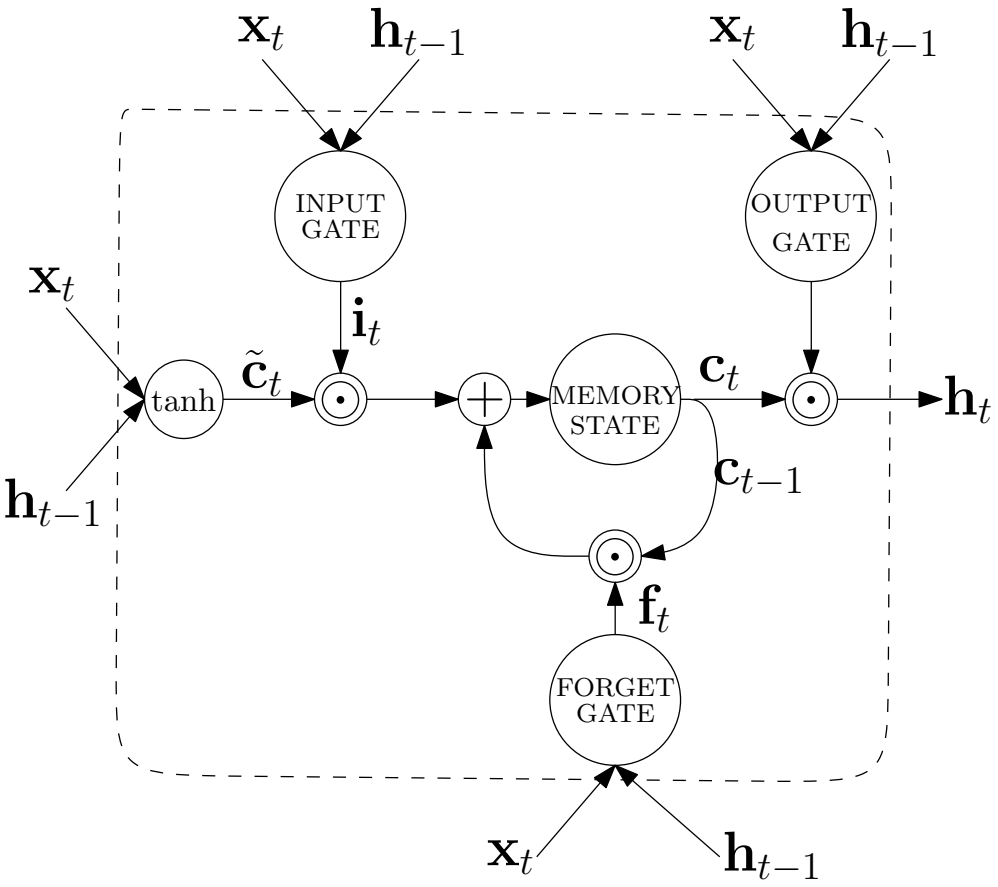
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1})$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1})$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1})$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1})$$



Sequence-to-Sequence Machine Translation

- Source sentence: $\mathbf{x} = x_1, \ldots, x_J, x_j \in V_x$.
- Target sentence: $\mathbf{y} = y_1, \ldots, y_I, y_i \in V_y$.
- Statistical Machine Translation goal:

$$\hat{\mathbf{y}} = \arg\max_{I,\mathbf{y}} Pr(\mathbf{y} \mid \mathbf{x})$$

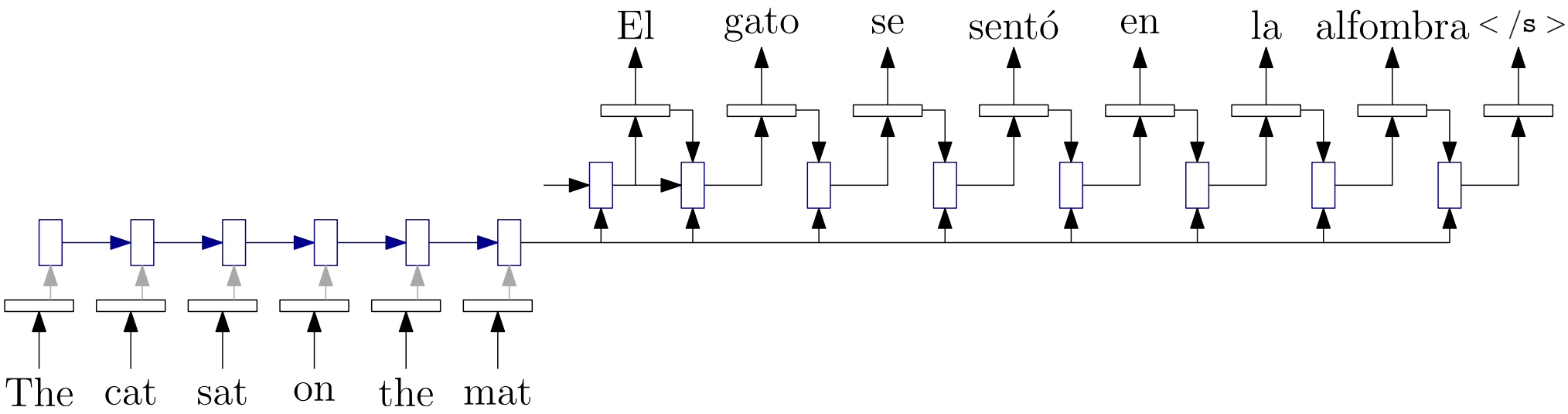
- NMT directly models the conditional translation probability:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{I} p(y_i \mid \mathbf{y}_{<i}, \mathbf{x})$$

Sequence-to-Sequence Machine Translation

1. Project source words to a continuous vector → *Source word embedding*.
2. Generate a compact representation of the source sentence → *Encoder*.
3. Obtain a continuous representation of the target sentence → *Decoder*.
4. Decode target representation to target words → *Target word embedding*.

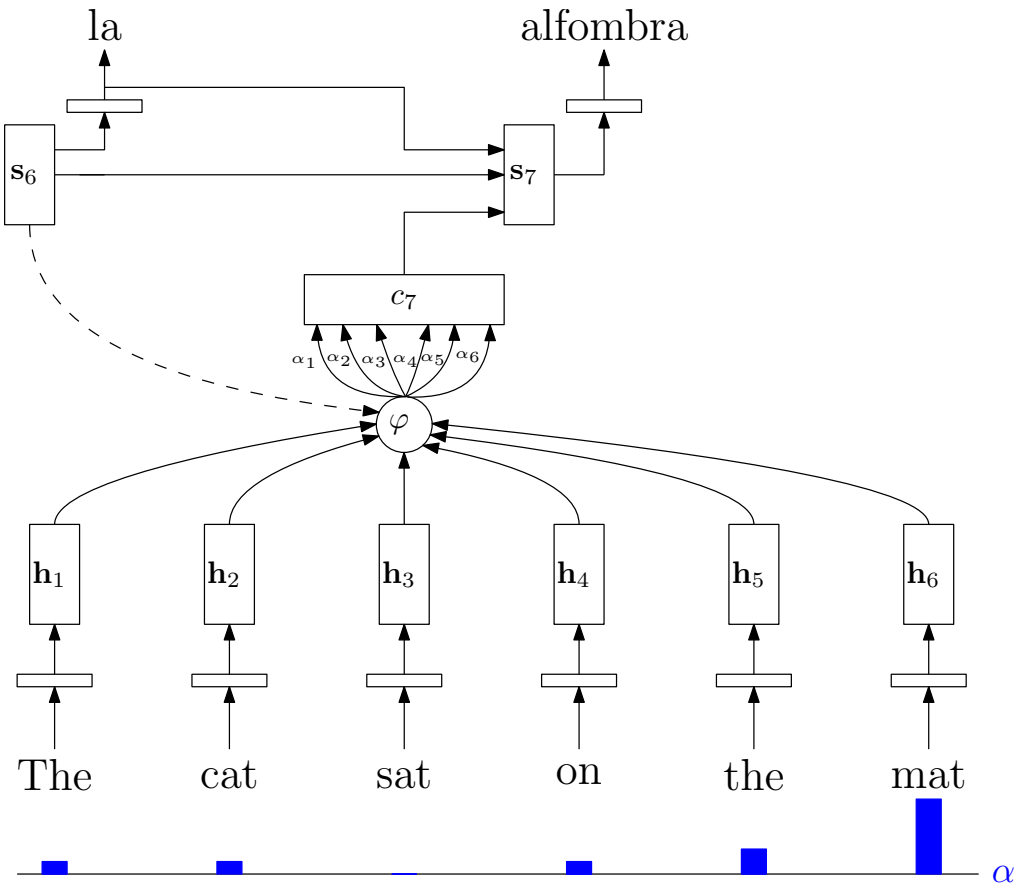
Encoder-Decoder



Attention-based Encoder-Decoder

- Problem! All the information cannot be stored in a compact vector for long sentences.
- Performance dropped in long sentences.
- (partial) Solution: Attention mechanism over the input sequence.
  - Dynamic representation of the relevant part of the sentence at the decoding time-step time i.
  - Weighted sum of the elements from input sequence.
  - Weights provided by an alignment model.

Attention-based Encoder-Decoder: Example



Google's Neural Machine Translation System

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes and Jeffrey Dean.

arXiv:1609.08144. 2016.

Architecture

- Deep-LSTM networks: Residual connections!
- Soft-Attention model: Single-layer feed-forward network (φ).

$$e_{ij} = \varphi(\mathbf{y}_{i-1}, \mathbf{h}_j) \quad \forall 1 \leq j \leq J$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{J} \exp(e_{ik})}$$

$$\mathbf{a}_i = \sum_{j=1}^{J} \alpha_{ij} \mathbf{h}_j$$

Residual Connections

- Very deep RNNs: Hard to train due to vanishing gradients problem.
- Residual connections.

Deep LSTM (layer l at time j):

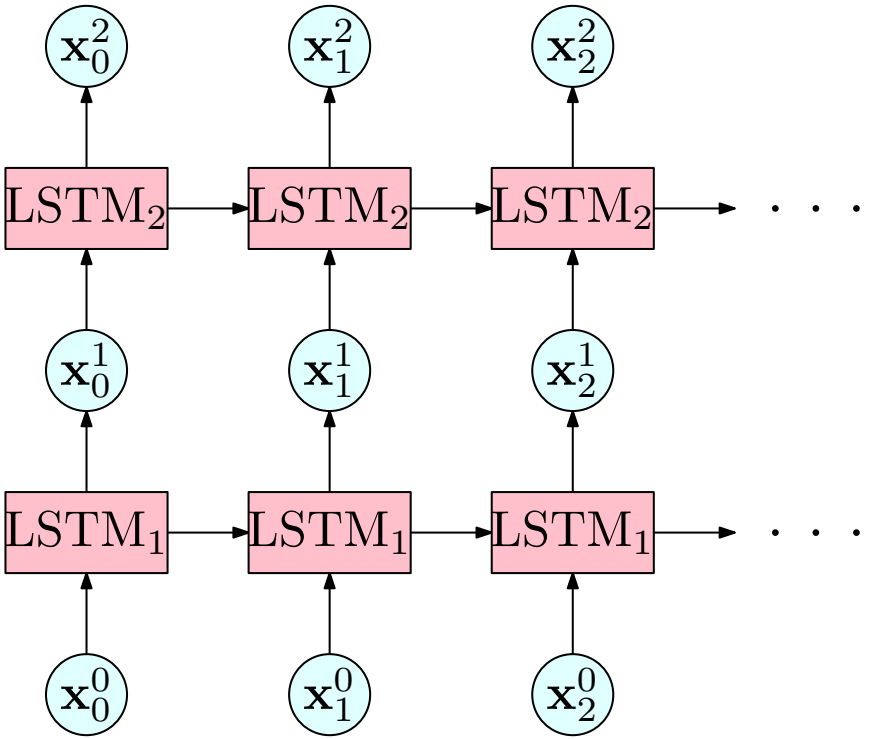
$$\mathbf{h}_j^l, \mathbf{c}_j^l = LSTM_l(\mathbf{h}_{j-1}^l, \mathbf{c}_{j-1}^l, \mathbf{x}_j^{l-1})$$
$$\mathbf{x}_j^l = \mathbf{h}_j^l$$
$$\mathbf{h}_j^{l+1}, \mathbf{c}_j^{l+1} = LSTM_{l+1}(\mathbf{h}_{j-1}^{l+1}, \mathbf{c}_{j-1}^{l+1}, \mathbf{x}_j^l)$$

Deep Residual LSTM (layer l at time j):

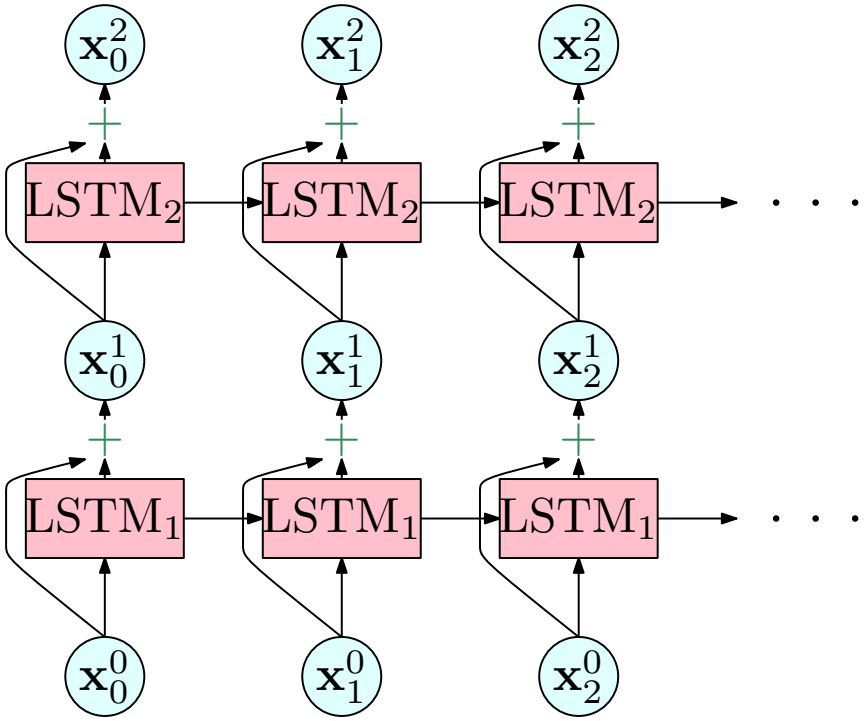
$$\mathbf{h}_j^l, \mathbf{c}_j^l = LSTM_l(\mathbf{h}_{j-1}^l, \mathbf{c}_{j-1}^l, \mathbf{x}_j^{l-1})$$
$$\mathbf{x}_j^l = \mathbf{h}_j^l + \mathbf{x}_j^{l-1}$$
$$\mathbf{h}_j^{l+1}, \mathbf{c}_j^{l+1} = LSTM_{l+1}(\mathbf{h}_{j-1}^{l+1}, \mathbf{c}_{j-1}^{l+1}, \mathbf{x}_j^l)$$

Residual Connections

Deep LSTM



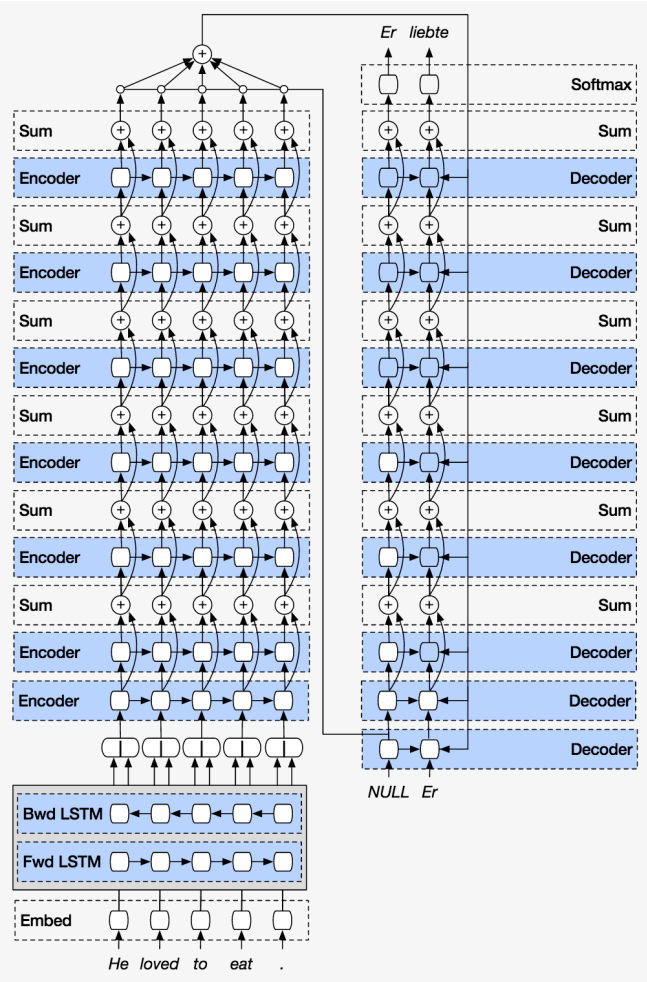
Deep residual LSTM



GNMT's Architecture & Parallelism

- 8 encoder LSTM layers.
- 8 decoder LSTM layers.
- Data parallelism:
  - n model replicas are trained concurrently with Downpour SGD.
  - Each replica asynchronously updates the parameters.
- Model parallelism:
  - Run each layer on a different GPU.
  - Partitioned softmax.
  - Constrains on model architecture:
    - Bidirectional encoder only for the first layer.
    - Attention model alignment: Bottom decoder output to the top encoder output.

The model


Er liebte
Softmax
Sum
Encoder
Decoder
Sum
Encoder
Decoder
Sum
Encoder
Decoder
Sum
Encoder
Decoder
Sum
Encoder
Decoder
Sum
Encoder
Decoder
Sum
Encoder
Decoder
Sum
Encoder
Decoder
Encoder
Decoder
Decoder
NULL Er
Bwd LSTM
Fwd LSTM
Embed
He loved to eat .

Practical issues

Segmentation approaches

- Limited size of vocabulary and out-of-vocabulary (OOV) words.
- Sub-word units: word-piece model (WPM). Similar to Byte Pair Encoding:

  Word: *Jet makers feud over seat width with big orders at stake*

  Sub-word: *_J et _makers _fe ud _over _seat _width _with _big _orders _at _stake*
  - Vocabulary size: $8k - 32k$
- Mixed word/character model: OOV words collapsed into a sequence characters.
  - Special prefixes indicate the position of the character (<B>, <M> <E>).
  - `Miki` $\rightarrow$ `<B>M <M>i <M>k <E>i`.
  - Reverse the tokenization to the original words as a post-processing step.

Training

- Training objective: Maximum likelihood. Maximize:

$$\mathcal{O}_{ML}(\theta) = \sum_{n=1}^{N} \log p(\hat{\mathbf{y}}^n \mid \mathbf{x}^n; \theta)$$

  - Parallel corpus: $\mathcal{S} = \{(\mathbf{x}^n, \hat{\mathbf{y}}^n)\}_{n=1}^{N}$.
  - Model parameters: θ.
- Stochastic Gradient Descent (SGD).
- Problem: Training objective differs from the evaluation metric (BLEU).
- Refinement of a model pre-trained on the maximum likelihood objective to directly optimize the evaluation metric $\rightarrow$ Reinforcement learning.

Training

- Expected reward objective:

$$\mathcal{O}_{RL}(\theta) = \sum_{n=1}^{N} \sum_{\mathbf{y} \in \mathcal{Y}} \log p(\mathbf{y} \mid \mathbf{x}^n; \theta) \;\; r(\mathbf{y}, \hat{\mathbf{y}})$$

  - $r(\mathbf{y}, \hat{\mathbf{y}})$: Per-sentence score.
  - $\mathcal{Y}$: m ($m = 15$) sequences drawn independently from the distribution $p(\mathbf{y} \mid \mathbf{x}^n; \theta)$.
- $r(\mathbf{y}, \hat{\mathbf{y}})$: $GLEU$ score.
- GLEU: Minimum of recall and precision of matching n-grams up to 4 in hypothesis and reference.
- $r(\mathbf{y}, \hat{\mathbf{y}})$: mean reward in $\mathcal{Y}$.

Training

- Training stabilization: linear combination of ML and RL objectives:

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

  - $\alpha = 0.017$ (???).
- Training procedure:
  1. Train with ML until convergence.
  2. Refine the model using $\mathcal{O}_{Mixed}(\theta)$ on a development set.

Decoding

- Decoding:

$$\hat{\mathbf{y}} = \arg\max_{I,\mathbf{y}} \log p(\mathbf{y} \mid \mathbf{x})$$

- Beam search.
  1. Original heuristic: **Length normalization**:

$$\log p(\mathbf{y} \mid \mathbf{x}) = \frac{\log p(\mathbf{y} \mid \mathbf{x})}{\mid \mathbf{y} \mid^{\gamma}}$$

     Good values for $\gamma \in [0.6, 0.7]$.
  2. **Length normalization + Coverage penalty**: Fully cover the source sentence according to the attention mechanism. Definition of a scoring function $s(\mathbf{y}, \mathbf{x})$:

$$s(\mathbf{y}, \mathbf{x}) = \frac{\log(p(\mathbf{y} \mid \mathbf{x}))}{lp(\mathbf{y})} + cp(\mathbf{x}; \mathbf{y})$$

     $lp(\mathbf{y})$: Length penalty function.
     $cp(\mathbf{x}; \mathbf{y})$: Coverage penalty function.

Decoding

$lp(\mathbf{y})$: Length penalty function:

$$lp(\mathbf{y}) = \frac{(5+ \mid \mathbf{y} \mid)^{\gamma}}{(5+1)^{\gamma}}$$

$cp(\mathbf{x};\mathbf{y})$: Coverage penalty function:

$$cp(\mathbf{x};\mathbf{y}) = \beta \sum_{j=1}^{J} \log(\min \sum_{i=1}^{|\mathbf{y}|} \alpha_{ij}, 1)$$

(This α_{ij} is the attention model weight between the source word j and the target word i).

Decoding

- $\gamma = 0.2$, $\beta = 0.2 \rightarrow \uparrow 1$ BLEU points in ML-trained models. Less effective after RL-refinement.
- Additional pruning. Let k be the beam size:
  - At each step, only consider hypotheses that have local scores that are not more than k times below the best token for this step.
  - Once a normalized best score has been found, prune all hypotheses that are more than k below the best normalized score so far.
- $k = 3$
- Batched decoding (up to 35 parallel sentences).

Deployment

- Challenging task: low latency translation required.
- Model computationally intensive at inference $\rightarrow$ Inference quantization.
- Weights can be quantized to only three states: -1, 0, and $+1$.
- Fixed-point integer operations with 8-bit or 16-bit resolution.
- Training at full precision. Constraints for the quantization:
  - Clip the values of $\mathbf{c}_t^l$ and $\mathbf{x}_t^l$ in $[-\delta, \delta]$
  - Output clipped to $[-\eta, \eta]$, then normalized.
- Tensor Processing Units (TPU):
  - Circuits developed specifically for machine learning.
  - High volume of reduced precision computation.

Deployment

Table 1: Model inference on CPU, GPU and TPU. Model trained with the ML objective with quantization.

	BLEU	Log Perplexity	Decoding time (s)
CPU[1]	31.2	1.4553	1322
GPU[2]	31.2	1.4553	3028
TPU	31.2	1.4626	384

- Training on GPU with quantization.
- Decoding on TPUs.

1. In 2 Intel Haswell CPUs, 88 CPU cores.
2. In a Nvidia Tesla K80.

Experiments and results

Experiments and results

- WMT'14 datasets (+internal Google datasets).
- Languages: En–Fr (36M training sentences), En–De (5M)

- Training setup:
  - Implemented using TensorFlow.
  - 12 replicas running concurrently on separate machines.
  - Every replica updates the shared parameters asynchronously.
- Training details:
  - All trainable parameters uniformly initialized between $[-0.04, 0.04]$.
  - Gradient norm clipping (5.0).
  - Dropout $(0.2 - 0.3)$.

Training procedure

1. Maximum likelihood training:
   - Adam (learning rate: 0.0002) during $60k$ updates.
   - Vanilla SGD (learning rate: 0.5). Learning rate annealing:
     - Start annealing after $1.2M$ updates.
     - Halve the learning rate each $200k$ updates during $800k$ updates.
   - Takes around 6 days to train a basic model using 96 NVIDIA K80 GPUs (!!!).
2. Reinforcement learning refinement:
   - Refine until the BLEU score does not change much on the development set.
   - SGD optimization.
   - Takes around 3 days.

Results: English – French

Table 2: English – French WMT'14. After maximum likelihood training.

Model	Test BLEU score
Phrase-based SMT	37.0
LSTM (6 layers) (Luong *et al.* , 2015)	31.5
LSTM (6 layers) + PosUnk (Luong *et al.* , 2015)	33.1
Deep Att (Zhou *et al.* , 2016)	37.7
Deep Att + PosUnk (Zhou *et al.* , 2016)	39.2
Word	37.9
Character	38.0
WPM-8K	38.3
WPM-16K	37.6
WPM-32K	38.9
Mixed Word/Character	38.4

Results: English – German

Table 3: English – German WMT'14. After maximum likelihood training.

Model	Test BLEU score
Phrase-based SMT	20.7
Att. GRU (Jean *et al.* , 2015)	16.0
Att. GRU Large Vocabulary (Jean *et al.* , 2015)	16.9
Deep Att (Zhou *et al.* , 2016)	20.6
Word	23.1
Character (512 nodes)	22.6
WPM-8K	23.5
WPM-16K	24.4
WPM-32K	24.6
Mixed Word/Character	24.2

Results: RL refinement

Table 4: BLEU scores of models prior and after RL refinement.

Language	ML	RL
En–Fr	38.9	39.9
En–De	24.7	24.6

Results: Ensembles

Table 5: English – French WMT'14.

Model	Test BLEU score
Phrase-based SMT	37.0
WPM-32K (single model)	38.9
WPM-32K (8 models)	40.3
RL-refined WPM-32K (8 models)	41.2

Results: Ensembles

Table 6: English – German WMT'14.

Model	Test BLEU score
Phrase-based SMT	20.7
WPM-32K	24.6
WPM-32K (8 models)	26.2
RL-refined WPM-32K (8 models)	26.3

Results: Human Evaluation

- Four-way side-by-side human evaluation.
- Humans rate four translations given a source sentence.
  1. Phrase based translations[1].
  2. Ensemble of 8 ML-trained models.
  3. Ensemble of 8 ML-trained and RL-refined models.
  4. Reference (human) translations.
- Scores from 0 (minimum) to 6 (maximum).

[1] From the Evaluation Matrix (EuroMatrix).

Results: Human Evaluation

Table 7: English – French WMT'14.

Model	Test BLEU score	Average side-by-side score
1. Phrase-based SMT	37.0	3.87
2. WPM-32K	40.3	4.46
3. RL-refined WPM-32K	41.2	4.44
4. Human		4.82

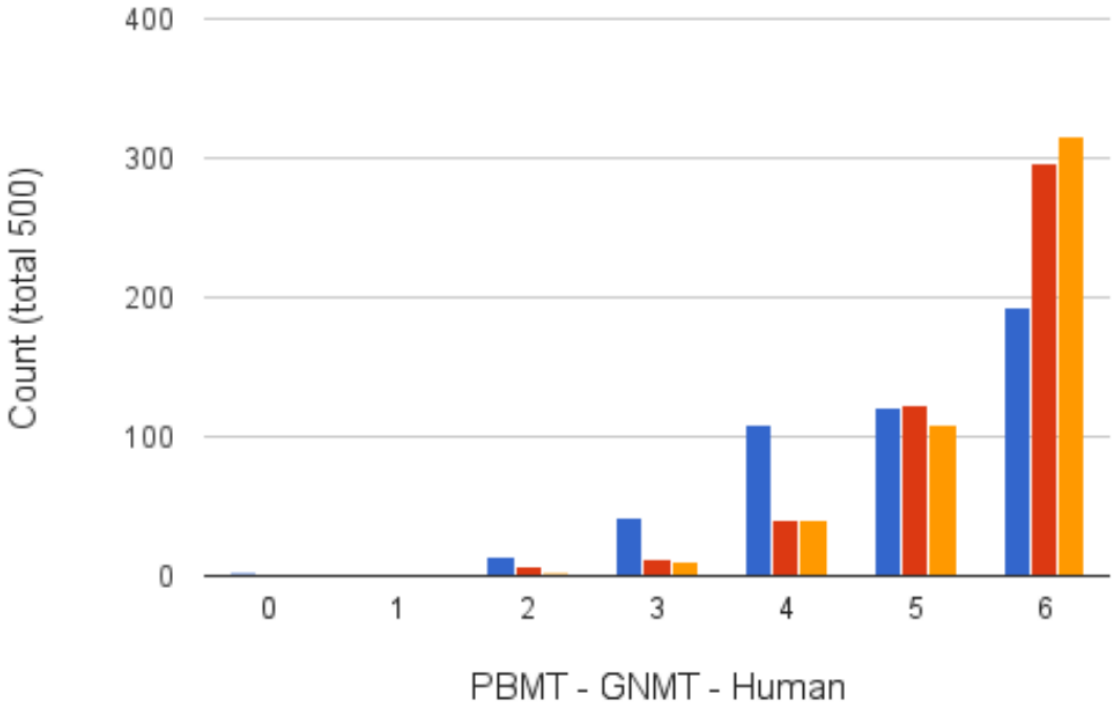
Results: Human Evaluation

Table 8: Results on production data.

Language	PB	GNMT	Human	Rel. Improvement
En→Es	4.88	5.43	5.50	87%
En→Fr	4.93	5.29	5.50	64%
En→Zh	4.03	4.59	4.99	58%
En→En	4.87	5.19	5.37	63%
Fr→En	5.05	5.34	5.40	83%
Zh→En	3.69	4.26	4.64	60%

Results: Human Evaluation

Histogram of side-by-side scores on 500 sampled sentences (En → Es) from Wikipedia and news websites. PBMT blue, GNMT red, Human orange.



Conclusions

- Wordpiece modeling effectively handles open vocabularies and morphologically rich languages.
- Model and data parallelism can be used to efficiently train NMT models.
  - Resource-demanding systems.
- Model quantization drastically accelerates translation inference, allowing the use of these large models in a deployed production environment.
- Although promising, RL refinement lacks of consistent results.
- Additional details like length-normalization, coverage penalties, and similar are essential to making NMT systems work well on real data.

References

Sébastien Jean, Kyunghyun Cho, Roland Memisevic and Yoshua Bengio.
On Using Very Large Target Vocabulary for Neural Machine Translation.
In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1–10*. 2015.

Minh-Thang Luong, Ilya Sutskever, Hieu Pham and Christopher D Manning.
Effective Approaches to Attention-based Neural Machine Translation.
In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1412–1321*. 2015

Zhou, J., Cao, Y., Wang, X., Li, P., and Xu, W.
Deep recurrent models with fast-forward connections for neural machine translation.
In *CoRR abs/1606.04199*. 2016.