

# Neural Machine Translation

Ramaseshan Ramachandran

- ① Neural Machine Translation
  - Encoder-Decoder Model
  - Recurrent Neural Network
  - Encoder
  - Decoder
  - Estimating Model Parameters

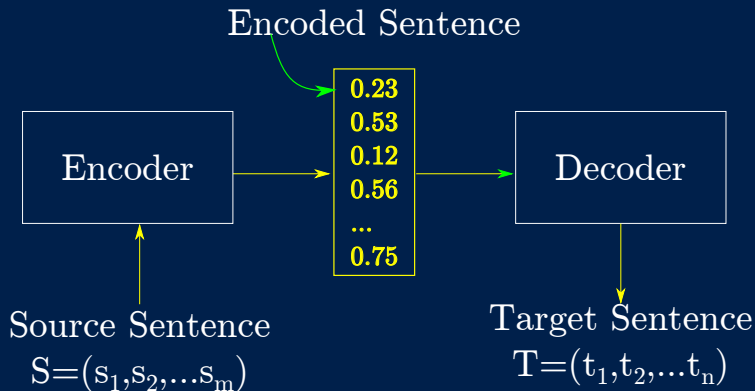
- The -BLEU idea
  - unigram precision
  - Modified- n-gram precision
  - Combining n-gram precisions
  - Demo
  - Other Metrics

# NEURAL MACHINE TRANSLATION

---

Neural Machine Translation (NMT) is the mechanism of modeling the Machine translation process using artificial neural network. Let  $F$  and  $E$  be the source and the target sentences in a parallel corpora, respectively.

# ENCODER-DECODER MODEL



- ▶ All sentences (of varying length) are encoded into fixed sized vector
- ▶ Uses fraction of the memory needed by traditional SMT models<sup>1</sup>
- ▶ Performance of this model decreases as the length of a source sentence increase

<sup>1</sup>Cho et al, On the Properties of Neural Machine Translation: Encoder-Decoder Approaches, 2014

# RNN-BASED TRANSLATION MODEL

---

- ▶ Uses RNN for both encoding and decoding
- ▶ Encoder maps the variable length sentence into a fixed-length vector
- ▶ Decoder translates the vector representation back to a variable-length target sequence
- ▶ Two networks are trained jointly to maximize the conditional probability of the target sentence, given the source sentence -  $P(f|e)$
- ▶ This model learns a continuous space representation of a phrase that preserves both the semantic and syntactic structure of the phrase[1].

# RECURRENT NEURAL NETWORK

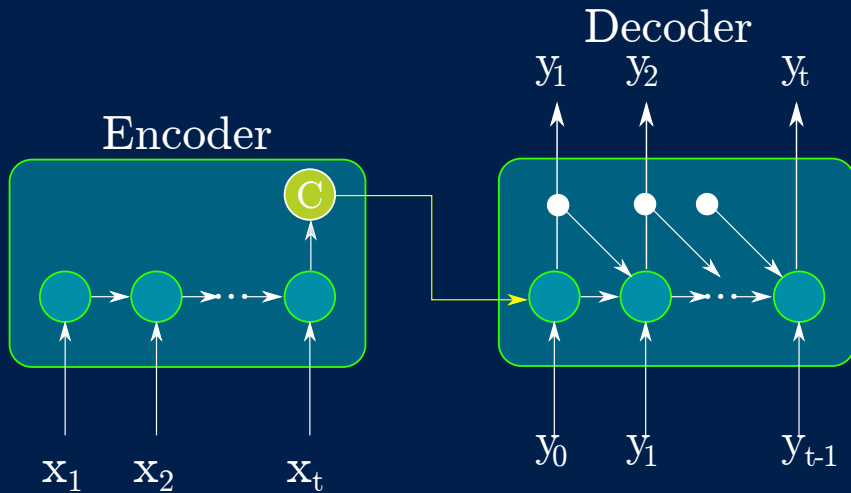
- ▶ Input units - variable length source sequence  $\mathbf{x} = (x_1, x_2, \dots, x_T)$
- ▶ Output units - variable length target sequence  $\mathbf{y} = (y_1, y_2, \dots, y'_T)$
- ▶ Hidden units for each input state,

$$h_t = f(h_{(t-1)}, x) \quad (1)$$

where  $f$  is a simple non-linear activation function (sigmoid or tanh) or a complex LSTM/GRU cell

- ▶ RNN is trained to predict the next word in the sequence or RNN learns a probability distribution over a sequence
- ▶ The output at each time step  $t = p(x_t | x_{t-1}, \dots, x_1)$
- ▶ The output distribution (Softmax layer) size is equal to the size of the vocabulary  $V$  at every unit
- ▶ Then,  $p(\mathbf{x}) = \prod_{t=1}^T p(x_t | x_{t-1}, \dots, x_1)$

# RNN-BASED ENCODER-DECODER



- ▶ RNN learns to map an input sentence of variable length into a fixed-dimensional vector representation.
- ▶ It learns to decode a fixed length vector representation back into a variable length sequence
- ▶ This model learns to predict a sequence given a sequence  $p(y_1, y_2, \dots, y_{T'} | x_1, x_2, \dots, x_T)$ .  $T$  and  $T'$  may differ
- ▶ Encoder reads every symbol in  $\mathbf{x}$ , sequentially
- ▶ Hidden state changes according to Eq.(1)
- ▶  $\mathbf{C}$  is the summary of the hidden states at time  $T$  and has encoded all the symbols in the sequence



- ▶ This is trained to predict the next symbol  $y_t$  and generate the output sequence, given the previous state  $h_t$
- ▶  $y_t$  and  $h_t$  are conditioned on the summary from the encoder,  $C$  and its previous hidden state
- ▶ Decoder's hidden state is given by
- ▶ Conditional distribution for the next symbol is

$$h_t = f(h_{t-1}, y_{t-1}, C) \quad (2)$$

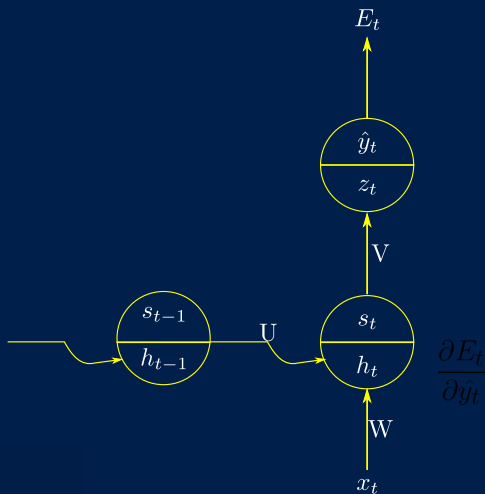
$$P(y_t | y_{t-1}, y_{t-2} \dots, y_1, C) = g(h_{t-1}, y_{t-1}, C) \quad (3)$$

Both encoder and decoder are jointly trained to maximize the conditional likelihood

$$J(\theta) = \max_{\theta} \frac{1}{N} \log p_{\theta}(\mathbf{y}_n | \mathbf{x}_m) \quad (4)$$

where  $\theta$  is the set of model parameters that will be learned during the BPTT and  $(\mathbf{x}_m, \mathbf{y}_n)$  is the source sentence sequence and target sequence pair

## BPTT - DERIVATIVE FOR V



$$h_t = (Wx_t + Uh_{t-1})$$

$$s_t = \tanh(h_t)$$

$$z_t = Vs_t$$

$$\hat{y}_t = \text{softmax}(z_t)$$

$$E_t = -y_t \log(\hat{y}_t)$$

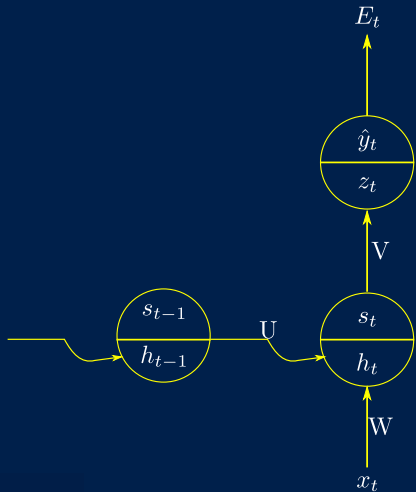
$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t} \frac{\partial z_t}{\partial V} \quad (5)$$

$$\text{Let } \delta_{out}^t = \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t}$$

$$\frac{\partial E_t}{\partial V} = \delta_{out}^t s_t \quad (6)$$

Here  $\delta_{out}^t$  is the loss for each of the units in the output layer

## BPTT - DERIVATIVE FOR W

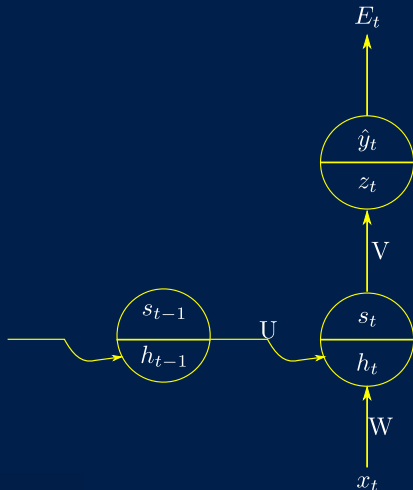


$$\frac{\partial E_t}{\partial W} = \underbrace{\frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t}}_{\delta_{\text{out}}^t} \frac{\partial z_t}{\partial s_t} \frac{\partial s_t}{\partial h_t} \frac{\partial h_t}{\partial W} \quad (7)$$

$$= \delta_{\text{out}}^t V \sigma'(h_t) x_t \quad (8)$$

Since the hidden layer activation depends on the previous time state, we have another similar term  $\delta_{t-1}$  that get added to (8)

## BPTT - DERIVATIVE FOR U

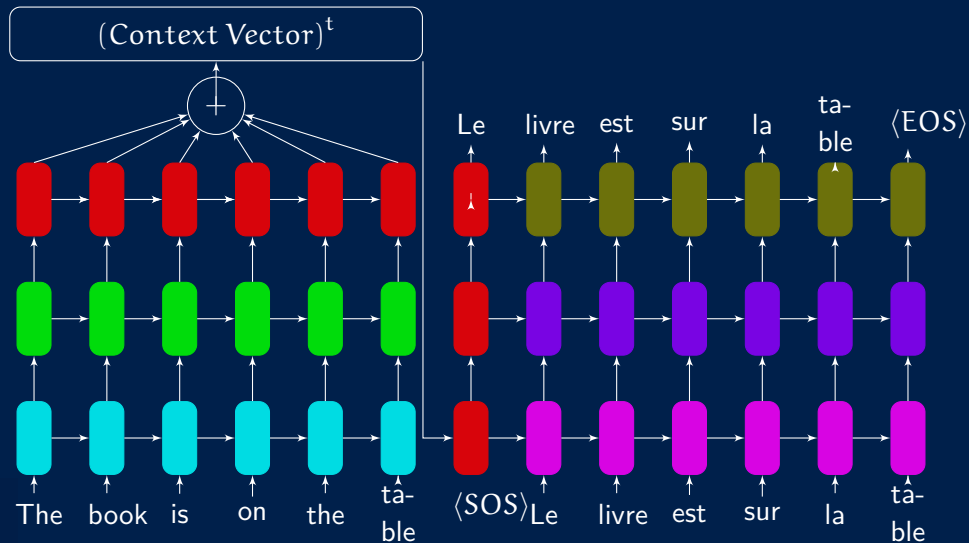


$$\frac{\partial E_t}{\partial \mathbf{U}} = \frac{\partial E_t}{\partial \hat{\mathbf{y}}_t} \underbrace{\frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{z}_t} \frac{\partial \mathbf{z}_t}{\partial \mathbf{s}_t} \frac{\partial \mathbf{s}_t}{\partial \mathbf{h}_t}}_{\delta_{\text{out}}^t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{U}} \quad (9)$$

$$= \delta_{\text{out}}^t \mathbf{V} \sigma'(\mathbf{h}_t) \mathbf{h}_{t-1} \quad (10)$$

Since we are back propagating the error from the current state to the previous state,  $\delta_{\text{next}} = \sigma(\mathbf{h}_t) \mathbf{U} \delta_{\text{out}}^t \mathbf{V} \sigma'(\mathbf{h}_t)$  needs to be added

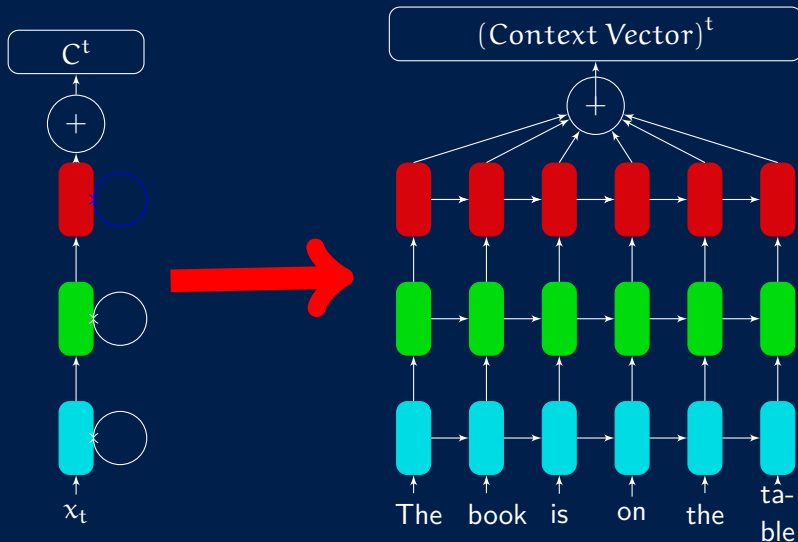
# SEQ2SEQ TRANSLATOR



Choices vary in picking the Translation Architecture

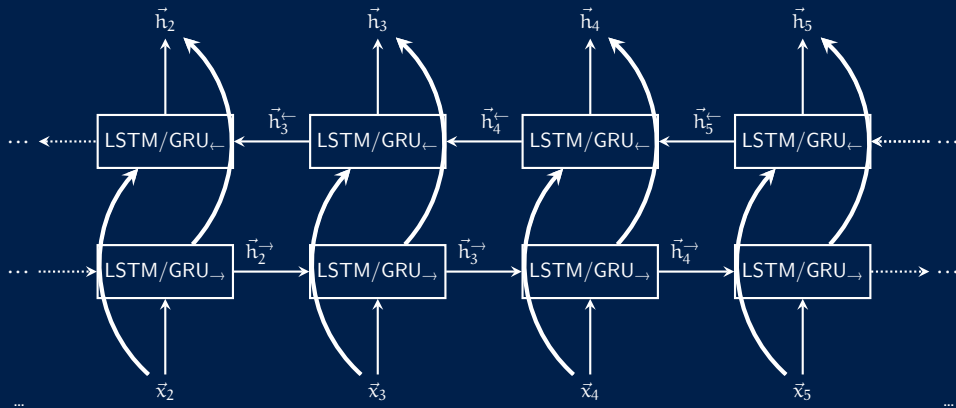
- ▶ Directionality - Unidirectional or bidirectional
- ▶ number of hidden layers and units
- ▶ Plain vanilla RNN
- ▶ Long Short-term Memory units
- ▶ Gated Recurrent Unit
- ▶ Choice of Learning Algorithm

# SEQ2SEQ ENCODER

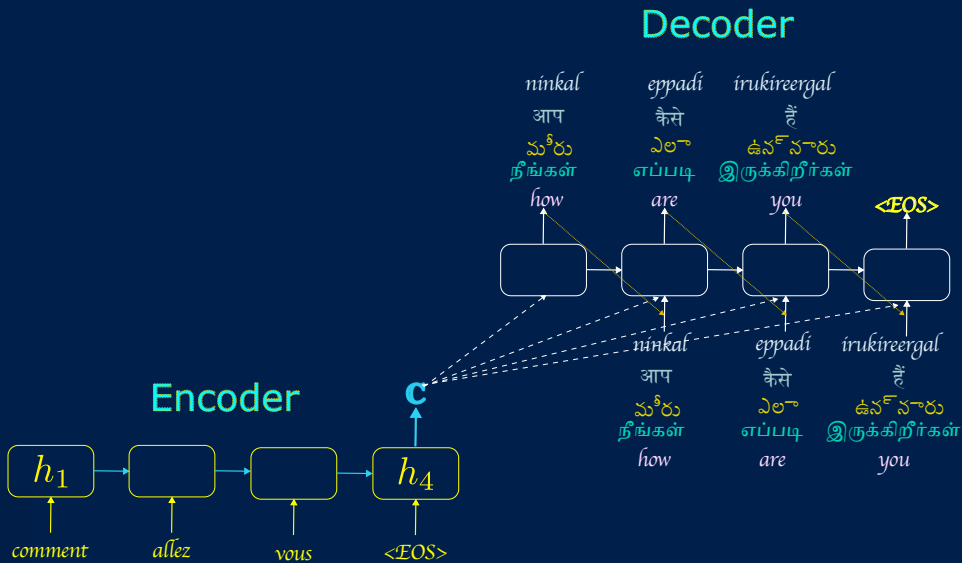




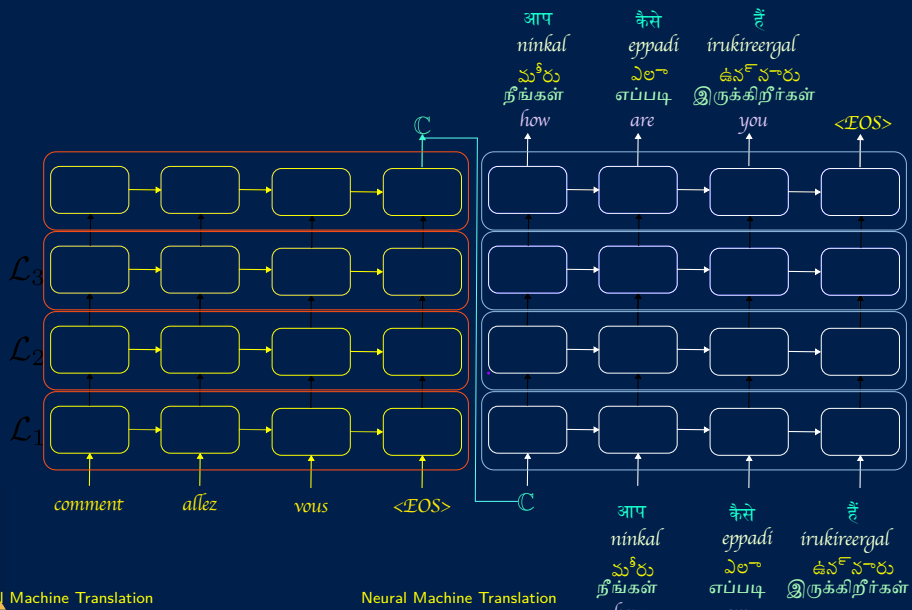
# BIDIRECTIONAL RNN



# SEQUENCE TO SEQUENCE TRANSLATION - NMT



# SEQUENCE TO SEQUENCE TRANSLATION - DEEP RNN



# APPLICATIONS

---

- ▶ Translation
- ▶ Dialog
- ▶ Code generation!

- ▶ The objective of attention is to capture the information from the passage tokens that is relevant to the contents of the translation
- ▶ Different parts of an input have different levels of significance
- ▶ Different parts of the output may even consider different parts of the input as "important"
- ▶ The purpose of the attention mechanism is to let the decoder *peek* at the relevant information encapsulating the source sentence as it generates the answer
- ▶ Attention mechanisms provide the decoder network with the entire input sequence at every decoding step; the decoder can then decide what input words are important at any point in time

- ▶ The attention-based model learns to assign significance to different parts of the input for each step of the output.
- ▶ In the context of translation, attention can be thought of as "alignment."
- ▶ Bahdanau et al [2] argue that the attention scores  $\alpha_{ij}$ , at decoding step  $i$ , signify the words in the source sentence that align with word  $j$  in the target.
- ▶ We can use attention scores to build an alignment table. It is a table mapping of words in the source to corresponding words in the target sentence - based on the learned encoder and decoder from our Seq2Seq NMT system.

Let  $x(x_1, x_2, \dots, x_n)$  and  $y(y_1, y_2, \dots, y_m)$  be the source and target sentences

The encoder reads the input sentence  $x$  and converts into a context vector  $c$

$$h_t = f(x_t, h_{t-1}) - \text{hidden values calculated at time } t \quad (11)$$

$$c = g(h_1, h_2, \dots, h_n) - \text{context vectors computed using all } h_t \text{ values} \quad (12)$$

where functions  $f$  and  $g$  are non-linear functions.

How does  $c$  differ from the *context* of an  $n$ -gram language model?

# DECODER

Decoder is trained to predict the next word using the  $c$  computed by the encoder

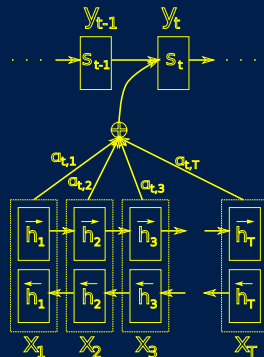
$$p(y) = p(y_t | c, \{y_1, y_2, \dots, y_{t-1}\}) \quad (13)$$

For the RNN, the probability of the next word  $p(y_t)$  is computed using

$$p(y_t) = g(y_{t-1}, s_t, c) \quad (14)$$

The hidden states  $s$  of the decoder are computed using a recursive formula of the form  $s_i = f(s_{i-1}, y_{i-1}, c_i)$ , where  $s_{i-1}$  is the previous hidden vector,  $y_{i-1}$  is the generated word at the previous step, and

$c_i$  is a context vector that capture the context from the original sentence that is relevant to the time step  $i$  of the decoder.





Conditional probability for each output neuron

$$p(y_i | y_1, y_2, \dots, x) = g(y_{i-1}, s_i, c_i) \quad (15)$$

where  $s_i$  is the RNN hidden neuron at time  $i$  and

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \quad (16)$$

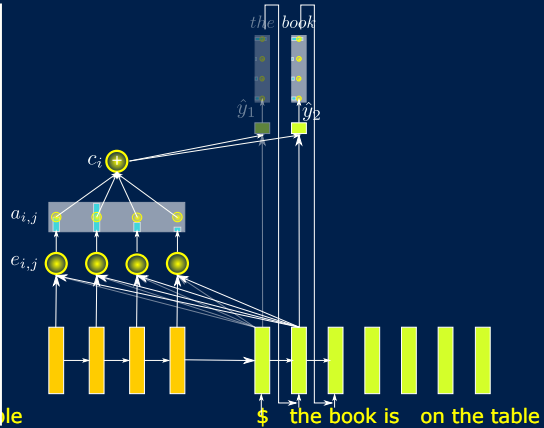
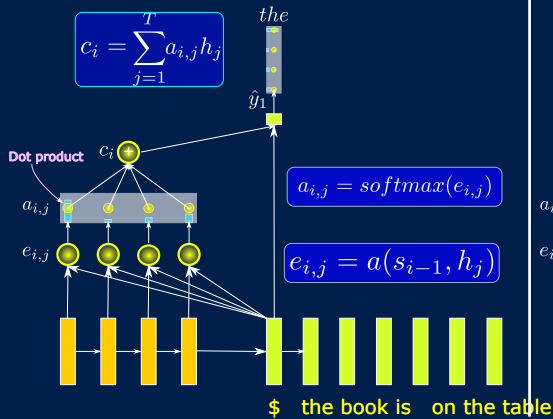
The context vector  $c_i$  depends on the sequence of annotations  $(h_1, h_2, \dots, h_{T_x})$ [3]. Each  $h_i$  contains information about every word with a strong focus on context words surrounding the  $i^{\text{th}}$  word of the input sequence.

The context vector  $c_i$  is computed as the weighted sum of these annotations  $h_i$

$$c_i = \sum_{j=1}^{T_x} \alpha_{i,j} h_j \quad \left| \quad \alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{t_x} \exp(e_{i,k})} \quad \left| \quad e_{i,j} = a(s_{i-1}, h_j) \right.$$

$\alpha_{ij}$  of each annotation  $h_j$  is computed by is the alignment model. This learns how well the inputs surrounding position  $j$  and the output at position  $i$  match

- ▶ The alignment is explicitly computed and not latent
- ▶ This alignment model is also trained along with the translation model
- ▶  $\alpha_{ij}$  is the probability that the target word  $y_i$  is aligned to the source  $x_j$
- ▶  $c_i$  is the expected annotation over all possible annotations  $\alpha_{ij}$
- ▶  $\alpha_{ij}$  or  $e_{ij}$  reflects the importance of the annotation  $h_j$  wrt to the previous hidden state  $s_{i-1}$  of the target. This enables the next state  $s_i$  to generate  $y_i$
- ▶ The decoder decides which part of the input is important to generate a respective translation rather than depending on the encoded vector of the entire sentence
- ▶ Decoder has control over the input sequence and selectively learns to align words/phrases automatically

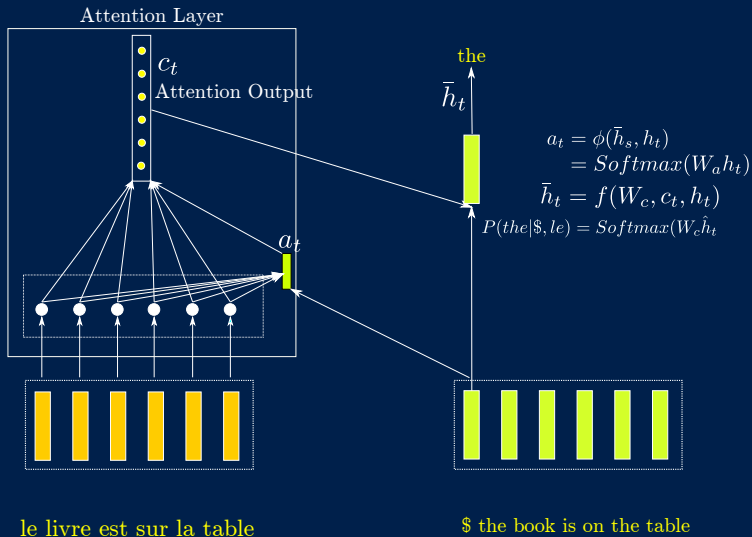


$e_{i,j}$ —attention score

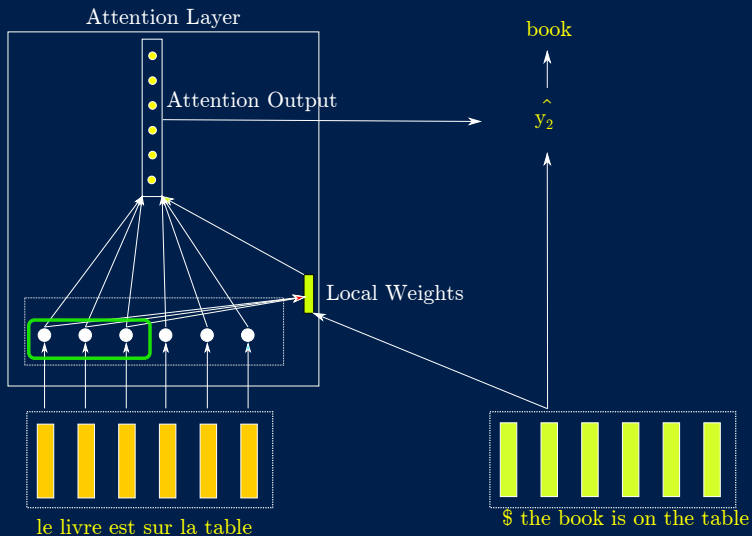
$a_{i,j}$ —attention distribution

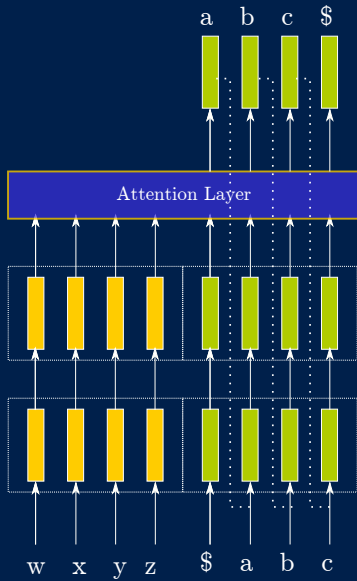
$c_i$ —attention output

# TRANSLATION WITH GLOBAL ATTENTION



# TRANSLATION WITH LOCAL ATTENTION





Source: Minh-Thang Luong et al, Effective Approaches to Attention-based Neural Machine Translation

## A TYPICAL SETUP

---

Sentence pairs	3-5M
English words	110M
French words	116M
Vocabulary	≈50K (Source and Target)
Word Embedding size	1000
Hidden layer	1000 LSTM cells
Stacked Hidden Layer	4-8
Learning Rate	Initially as high as 1 and exponential reduction
Training	
Mini batch Gradient Descend size	128
Training Time	1 GPU - about 7-10 days
Evaluation	Bleu - scores ranging from 27-32

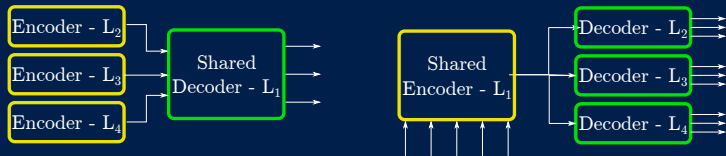
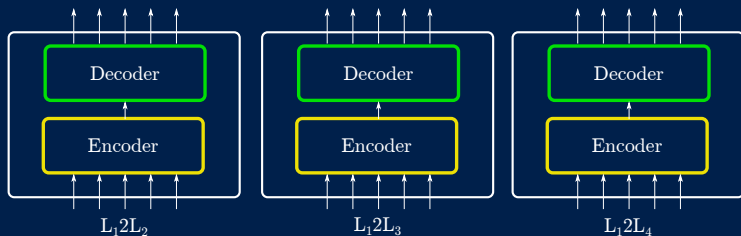
# ADVANTAGES OF ATTENTION

---

- ▶ Ability to focus on significant part of the sentence
- ▶ Ability to peek into source sentence
- ▶ Reduces the problem of vanishing gradient
- ▶ Alignments are found automatically during the training process
- ▶ Improves NMT performance for alignment



# DIFFERENT TYPE OF NMT



# NMT FROM GOOGLE - ZERO-SHOT TRANSLATION

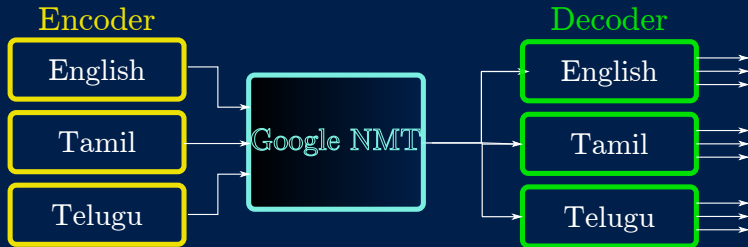
---

- ▶ Moved away from maintaining Seq2Seq model for every pair of languages
- ▶ A single system that translates between any two languages even in the absence of the training corpus for these two languages
  - ▶ Assume that only examples of Japanese-English and Korean-English translations are available, Google found that the multilingual NMT system trained on this data could actually generate reasonable Japanese-Korean translations.
  - ▶ Is it trained to create the Interlingua?
  - ▶ Is the system learning a common representation or a translational knowledge?

**Ref:** Johnson et al. 2016, "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation"

# ZERO-SHOT TRANSLATION

---

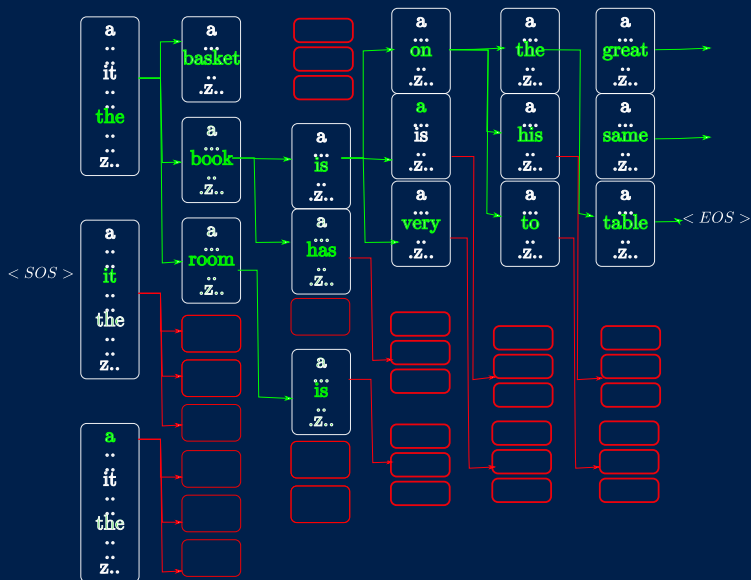


Beam search is a heuristic search algorithm that selects a few candidate hypothesis from  $|V|$ . It reduces memory requirement by using only a  $M < |V|$  candidates using a score.

- ▶ Maintain  $M$  candidates/hypothesis at each time step -  $C_t = (x_1^1, ..x_t^1) ... (x_1^M ... x_t^M)$
- ▶ Compute  $C_{t+1}$  by expanding  $C_t$  and keeping the best  $M$  candidates
- ▶  $\tilde{C} = \bigcup_{i=1}^M C_{t-1}^i$

Typical Beam width of size 5-10 used in NMT. The BLEU scores computed using Beam search using  $B=5-10$  are comparable

# BEAM SEARCH - BEAM WIDTH = 3



1. Use all possible partial translations - exhaustive search
2. Beam size,  $b = 1$  - greedy search - Words are predicted until the  $\langle \text{EOS} \rangle$  is found
3.  $b > 1$  - several hypotheses
4. Each hypothesis will be produced until the  $\langle \text{EOS} \rangle$  is found
5. Each hypothesis will have a translation
6. The length of all hypothesis may not be the same
7. We could use different **terminate** conditions
  - ▶ Fixed time steps
  - ▶ Compute until  $\langle \text{EOS} \rangle$  is reached for each hypothesis
8. Use either log probability or product of conditional probability to find the scores for each hypothesis that maximizes

○  $P(y_1, y_2, \dots, y_m | \mathbf{X}) = \prod_{t=1}^T P(y_t | \langle \text{SOS} \rangle, \dots, y_{t-1}, \mathbf{X})$

○  $P(y_1, y_2, \dots, y_m | \mathbf{X}) = \sum_{t=1}^T \log P(y_t | \langle \text{SOS} \rangle, \dots, y_{t-1}, \mathbf{X})$

# DIFFICULTIES WITH HUMAN EVALUATION OF MT

---

- ▶ Human evaluations are extensive but expensive
- ▶ A need for quick, reusable, inexpensive method that correlates highly with human evaluation
- ▶ Many aspects of translation, including adequacy and fluency should be considered during the automatic evaluation
- ▶ Automatic evaluation is a boon to developers of MT
- ▶ Two important aspects required for automatic evaluation
  1. A good metric
  2. A good/gold standards as references

# THE IDEA

---

- ▶ Many translations possible for a given sentence
- ▶ A good translator identifies a good candidate using adequacy and fluency

The main idea is to use a weighted average of variable length phrase matches against the reference translations[4]

Candidate 1: **It is a guide to action which ensures that the military always obeys the commands of the party**

Candidate 2: **It is to insure the troops forever hearing the activity guidebook that party direct**

Reference: **It is a guide to action that ensures that the military will for ever heed Party commands**

If many words and phrases are shared between the candidate and the reference translations, then it a good choice

Can n-grams help in matching the words and phrases?



# UNIGRAM PRECISION

C1: It is a guide to action which ensures that the military always **obeys** the commands of the party

R1: [It] [is] [a] [guide] [to] [action] that [ensures] [that] [the] [military] will forever heed [Party] [commands]

R2: It is the guiding principle which guarantees the military forces always being under the command [of] [the] Party.

R3: It is the practical guide for the army to heed the directions of the party.

$$\text{Unigram precision} = \frac{17}{18}$$

C2: It is to **insure** the **troops** forever **hearing** the **activity guidebook** that party **direct**

R1: [It] [is] a guide to action [that] ensures that [the] military will [forever] heed Party commands

R2: It is the guiding principle which guarantees the military forces always being under the command of the [Party].

R3: It is the practical guide for the army always to heed the directions of the party.

$$\text{Unigram precision} = \frac{8}{14}$$

## MODIFIED- N-GRAM PRECISION

---

Compare the number of n-grams in the candidate and in the reference translation

Penalize models that produces many words of the same type

- ▶ Count the number of times a word occurs in any single reference translation
- ▶  $\text{Count}_{\text{clip}} = \min(\text{Candidate Count}, \text{Maximum Reference Count})$

---

Refer the previous slide for the examples

Modified unigram precision for C1 =  $\frac{17}{18}$

• Modified unigram precision for C2 =  $\frac{8}{14}$

---

C3: the the the the the the the

R4: the cat is on the mat

Unigram precision =  $\frac{7}{7}$

Modified unigram precision =  $\frac{2}{7}$

Modified bigram precision = 0

---

Modified Unigram precision defines the adequacy of the translation, while modified bigram precision matches the fluency of the translation

# MODIFIED BIGRAM PRECISION

(It,is),(is,a),(a,guide),  
(guide,to),(to,action),  
(action,which),(which,ensures),  
(ensures,that),(that,the),  
(the,military),(military,always),  
(always,obeys),(obeys,the),  
(the,commands),(commands,of),  
(of,the),(the,party)

Modified bigram precision for C1 =  $\frac{10}{17}$

```
uuu(It,is),(is,a),(a,guide),(guide,to),
uuu(to,action),(action,that),(that,ensures),
uuu(ensures,that),(that,the),(the,military),
uuu(military,will),(will,forever),(forever,heed),
uuu(heed,Party),(Party,commands)
uuu
uuu(It,is),(is,the),(the,guiding),
uuu(guiding,principle),(principle,which),
uuu(which,guarantees),(guarantees,the),
uuu(the,military),(military,forces),(forces,always),
uuu(always,being),(being,under),(under,the),
uuu(the,command),(command,of),
uuu(of,the),(the,Party)
uuu
uuu(It,is),(is,the),(the,practical),(practical,guide),
uuu(guide,for),(for,the),(the,army),
uuu(army,always),(always,to),(to,heed),
uuu(heed,the),(the,directions),
uuu(directions,of),(of,the),(the,party)
uuu
```

## MODIFIED BIGRAM PRECISION - CANDIDATE 2

(It,is),(is,to),(to,insure),  
(insure,the),(the,troops),  
(troops,forever),(forever,hearing),  
(hearing,the),(the,activity),  
(activity,guidebook),  
(guidebook,that),(that,party),  
(party,direct)

Modified bigram precision for C2 =  $\frac{1}{13}$

```
uuu(It,is),(is,a),(a,guide),(guide,to),
uuu(to,action),(action,that),(that,ensures),
uuu(ensures,that),(that,the),(the,military),
uuu(military,will),(will,forever),(forever,heed),
uuu(heed,Party),(Party,commands)
uuu
uuu(It,is),(is,the),(the,guiding),
uuu(guiding,principle),(principle,which),
uuu(which,guarantees),(guarantees,the),
uuu(the,military),(military,forces),(forces,always),
uuu(always,being),(being,under),(under,the),
uuu(the,command),(command,of),
uuu(of,the),(the,Party)
uuu
uuu(It,is),(is,the),(the,practical),(practical,guide),
uuu(guide,for),(for,the),(the,army),
uuu(army,always),(always,to),(to,heed),
uuu(heed,the),(the,directions),
uuu(directions,of),(of,the),(the,party)
uu
```

## COMBINING N-GRAM PRECISIONS

---

- ▶ Modified n-gram precisions decay exponentially as n increases[4]
- ▶ BLEU uses a average log with a uniform weights to tackle the decay problem to get a score equivalent to the geometric mean of modified n-gram precisions
- ▶  $c < r$  inflates the precision
- ▶ A brevity penalty (BP) is introduced when  $c \leq r$

$$BP = \begin{cases} 1, & \text{if } c > r \\ \exp(1 - \frac{r}{c}), & \text{if } c \leq r \end{cases}$$

where  $r$  is the effective length of the reference corpus and  $c$  is the length of the candidate sentence

BLEU score is obtained by

$$\text{BLEU} = \text{BP} \cdot \exp \sum_{n=1}^N w_n \log p_n \quad (17)$$

where  $N$  is the  $n$ -gram size (BLEU uses 4-gram by default),  $w_n$  is the weights associated with unigram, bigram, trigram and 4-grams, and  $p_n$  is the modified precision score of the test corpus. Here,  $\sum_{n=1}^N w_n = 1$ . One option for  $w_n = \frac{1}{N}$

$$p_n = \frac{\sum_{c \in C} \sum_{\text{ngrams} \in C} \text{Count}_{\text{clip}}(\text{ngrams})}{\sum_{c \in C} \sum_{\text{ngrams} \in C} \text{Count}(\text{ngrams})} \quad (18)$$

## BLEU Demo

BLEU is designed as a corpus measure

- ▶ Machine translation
- ▶ Image labeling
- ▶ Text summarization
- ▶ Speech recognition



## OTHER METRICS

---

- ▶ NIST - National Institute of Standards and Technology - based on BLEU
- ▶ METEOR - Metric for Evaluation of Translation with Explicit ORdering
  - Uses stemming and synonymy matching
- ▶ WER - Word Error Rate
  - ▶ Uses edit distance (Levenshtein distance)
  - ▶ Finds minimum number of edit operations such as insertion, deletions or substitutions, needed to change the candidate sentence into the reference sentence
- ▶ GLEU - Google BLEU
  - ▶ Correlates well with BLEU, and works with sentence level translation

## REFERENCES

---

- [1] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: [10.3115/v1/D14-1179](https://doi.org/10.3115/v1/D14-1179). URL: <https://aclanthology.org/D14-1179>.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. English (US). In: *arXiv* (2014).
- [3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015.
- [4] Kishore Papineni et al. “Bleu: a Method for Automatic Evaluation of Machine Translation”. In: *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for