Neural Machine Translation

Seq2Seq, Attention and Beam Search

Ramaseshan Ramachandran

①

②

③

④

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.

Translation as Probabilistic Decoding:

$$\hat{E} = \arg\max_{E} p_\theta(E \mid F) \tag{1}$$

Neural Training Objective:

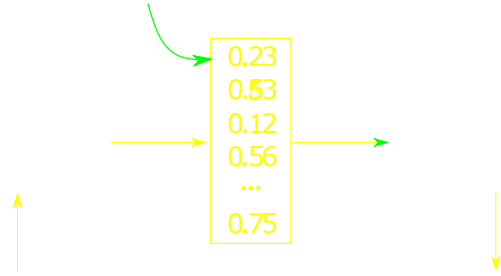
$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(E^{(i)}|F^{(i)}) \tag{2}$$

- Given a source sentence F, the goal is to find the most probable target sentence $\hat{E}$.
- The neural model is trained to maximize the conditional likelihood of target sentences given source sentences over the training corpus.

Once the conditional distribution is learned by a translation model, given a source sentence a corresponding translation can be generated by searching for the sentence that maximizes the conditional probability.

Unlike the phrase-based SMT models, NMT attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation[1]

ENCODER-DECODER MODEL



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

[1] 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, namely $P(E|F)$
- This model learns a continuous space representation of a phrase that preserves both the semantic and syntactic structure of the phrase[2].

RECURRENT NEURAL NETWORK

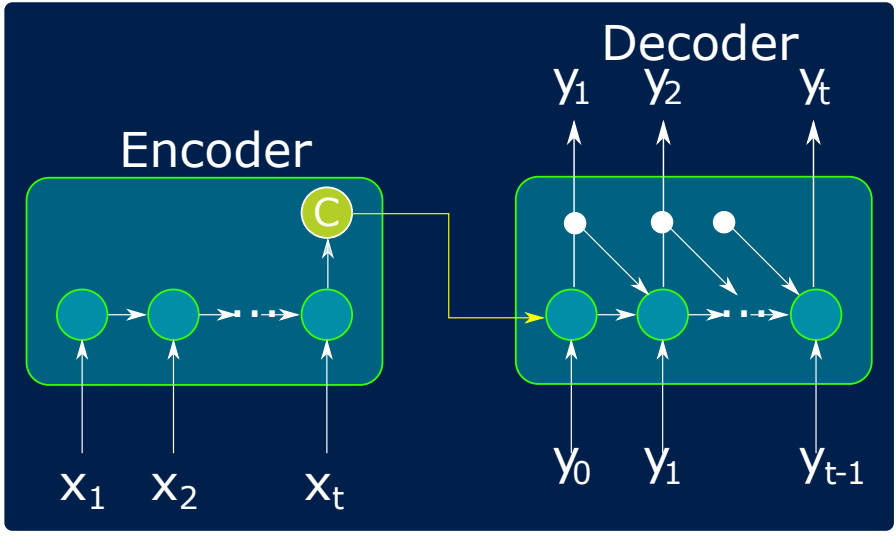
- Input units - variable length source sequence $x = (x_1, x_2, \ldots, x_T)$
- Output units - variable length target sequence $y = (y_1, y_2, \ldots, y'_T)$
- Hidden units for each input state,

$$h_t = f(h_{(t-1)}, x) \tag{3}$$

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 $t = p(x_t|x_{t-1}, \ldots x_1)$
- The output distribution (Softmax layer) size is equal to the size of the vocabulary V at every unit
- Then, $p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t|x_{t-1}, \ldots x_1)$

RNN-BASED ENCODER-DECODER



RNN-BASED ENCODER

- 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 $p(y_1, y_2, \ldots, y'_T | x_1, x_2, \ldots, x_T)$. T and T′ may differ
- Encoder reads every symbol in **x**, sequentially
- Hidden state changes according to Eq.(3)
- C is the summary of the hidden states at time T and has encoded all the symbols in the sequence

RNN-BASED DECODER

- This is trained to predict the next symbol y_t and generate the output sequence, given the previous state h_t
- y_t and $\mathbf{h}_t$ 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

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, y_{t-1}, \mathbf{C}) \tag{4}$$

$$P(y_t | y_{t-1}, y_{t-2} \ldots, y_1, \mathbf{C}) = g(h_{t-1}, y_{t-1}, \mathbf{C}) \tag{5}$$

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

$$J(\theta) = \max_{\theta} \frac{1}{N} \log p_\theta(\mathbf{y}_n|\mathbf{x}_m) \tag{6}$$

where θ 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

$$h_t = (Wx_t + Uh_{t-1})$$
$$s_t = \tanh(h_t)$$
$$z_t = Vs_t$$
$$\hat{y}_t = softmax(z_t)$$
$$E_t = -y_t \log(\hat{y}_t)$$
$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial z_t}\frac{\partial z_t}{\partial V} \tag{7}$$
$$\text{Let } \delta^t_{out} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}}{\partial z_t}$$
$$\frac{\partial E_t}{\partial V} = \delta^t_{out} s_t \tag{8}$$

Here δ^t_{out} is the loss for each of the units in the output layer

BPTT - DERIVATIVE FOR W

$$\frac{\partial E_t}{\partial W} = \underbrace{\frac{\partial E_t}{\partial \hat{y_t}} \frac{\partial \hat{y}}{\partial z_t}} \frac{\partial z_t}{\partial s_t} \frac{\partial s_t}{\partial h_t} \frac{\partial h_t}{\partial W} \tag{9}$$

$$= \delta^t_{out} V \sigma^{'}(h_t) x_t \tag{10}$$

Since the hidden layer activation depends on the previous time state, we have another similar term δ_{t-1} that get added to (10)

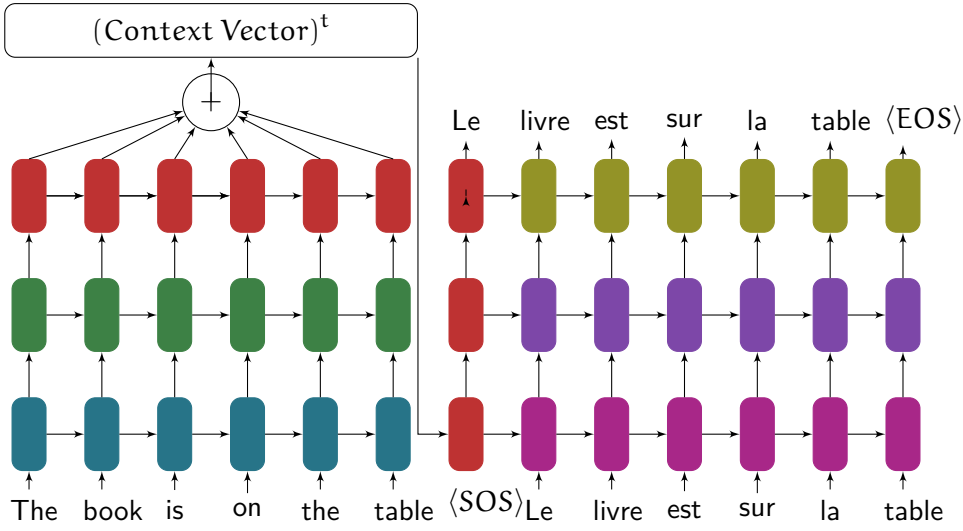
BPTT - DERIVATIVE FOR U

$$\frac{\partial E_t}{\partial U} = \underbrace{\frac{\partial E_t}{\partial \hat{y_t}} \frac{\partial \hat{y}}{\partial z_t} \frac{\partial z_t}{\partial s_t} \frac{\partial s_t}{\partial h_t}} \frac{\partial h_t}{\partial U} \tag{11}$$

$$= \delta^t_{out} V \sigma^{'}(h_t) h_{t-1} \tag{12}$$

Since we are back propagating the error from the current state to the previous state, $\delta_{next} = \sigma(h_t) U \delta^t_{out} V \sigma^{'}(h_t)$ needs to be added

SEQ2SEQ TRANSLATOR

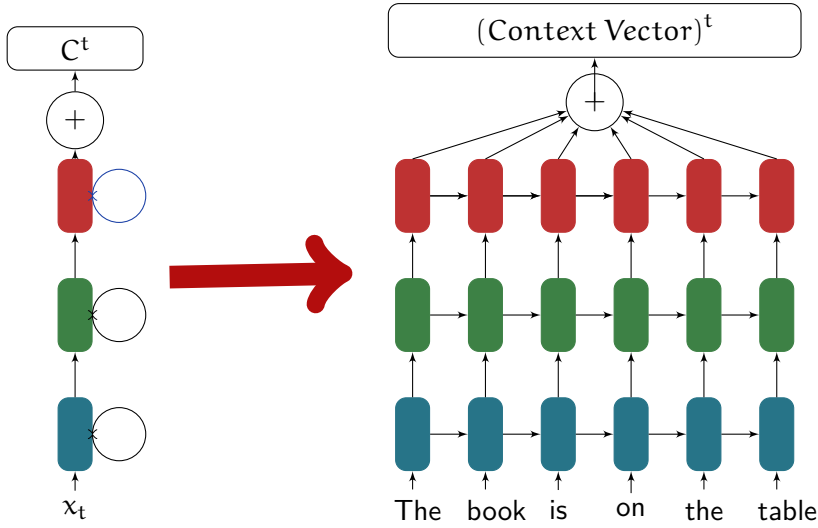


VARIATIONS IN RNN MODELS

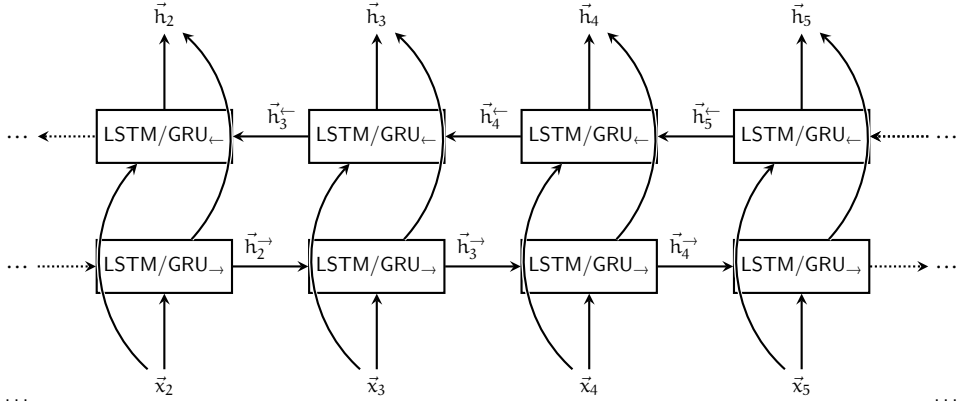
Choices vary in picking the Translation Architecture

- 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

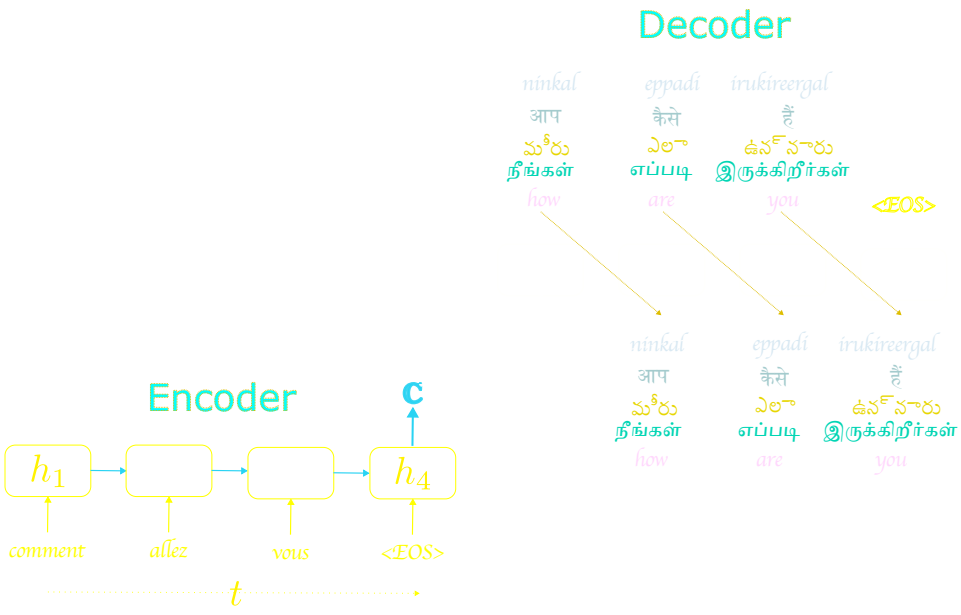
SEQ2SEQ ENCODER



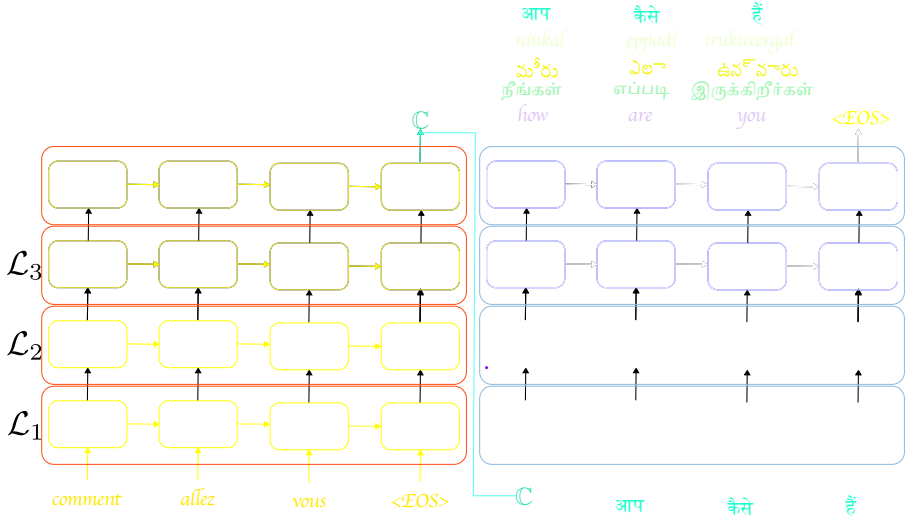
BIDIRECTIONAL RNN



SEQUENCE TO SEQUENCE TRANSLATION - NMT



SEQUENCE TO SEQUENCE TRANSLATION - DEEP RNN



APPLICATIONS

- Translation
- Dialog
- Code generation!

RNN WITH ALIGNMENT

- 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 [1] argue that the attention scores α_{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.

ENCODER

Let $x(x_1, x_2, \ldots, x_n)$ and $y(y_1, y_2, \ldots, y_m)$ be the source and target sentences
The encoder reads the input sentence x and converts into a context vector c

$$h_t = f(x_t, h_{t-1}) - \text{hidden values calculated at time } t \tag{13}$$

$$c = g(h_1, h_2, \ldots, h_n) - \text{context vectors computed using all } h_t \text{ values} \tag{14}$$

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

$$p(y) = p(y_t|c, \{y_1, y_2, \ldots, y_{t-1}\}) \quad (15)$$

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

$$p(y_t) = g(y_{t-1}, s_t, c) \quad (16)$$

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.

Figure

NMT WITH ATTENTION

Conditional probability for each output neuron

$$p(y_i|y_1, y_2, \ldots, x) = g(y_{i-1}, s_i, c_i) \tag{17}$$

where s_i if the RNN hidden neuron at time i and

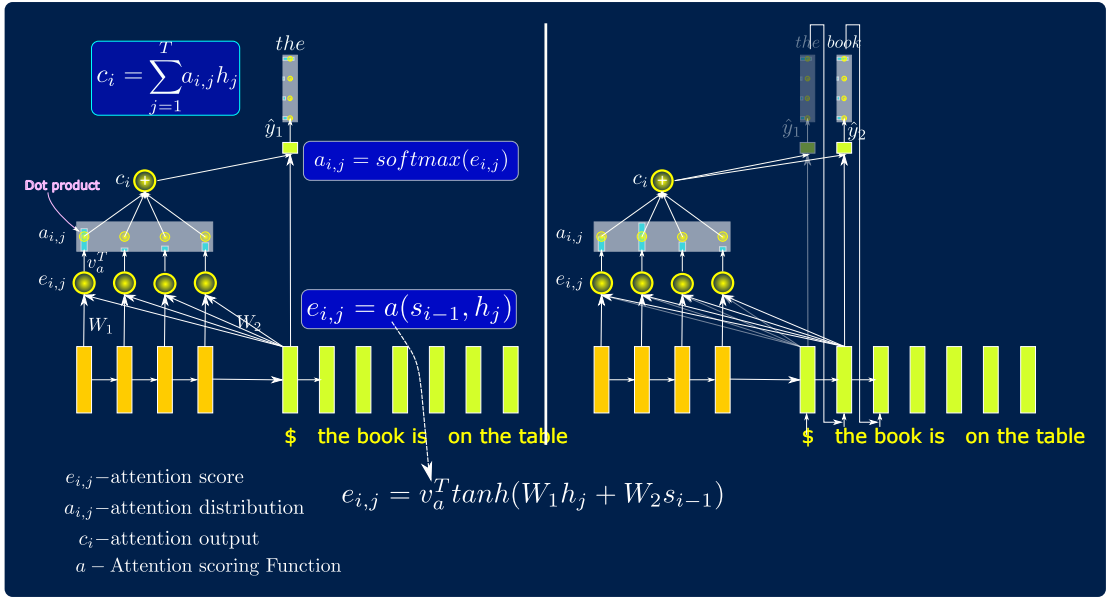
$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{18}$$

The context vector c_i depends on the sequence of annotations $(h_1, h_2, \ldots . h_{T_x})$[1].
Each h_i contains information about every word with a strong focus on context words surrounding the i^{th} word of the input sequence.
The context vector c_i is computed as the weighted sum of these annotations h_i

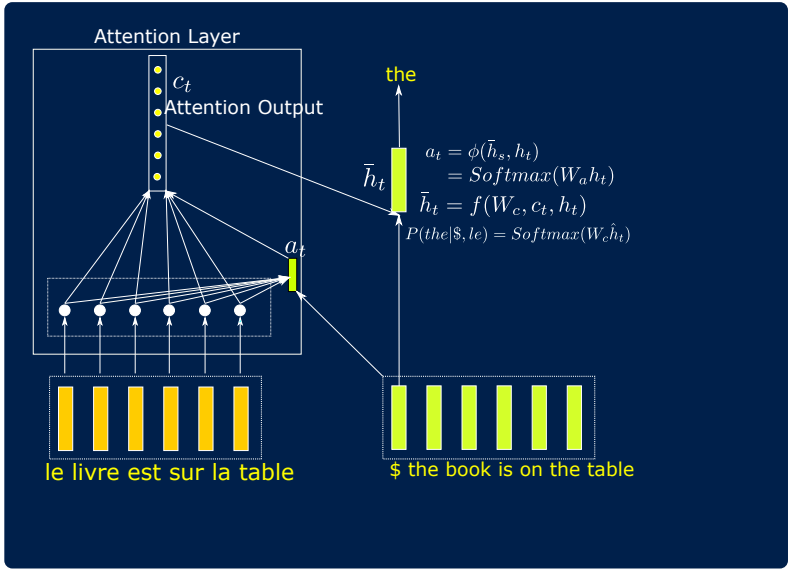
$$c_i = \sum_{j=1}^{T_x} \alpha_{i,j} h_j \qquad \alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{t_x} \exp(e_{i,k})} \qquad e_{i,j} = a(s_{i-1,h_j})$$

α_{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

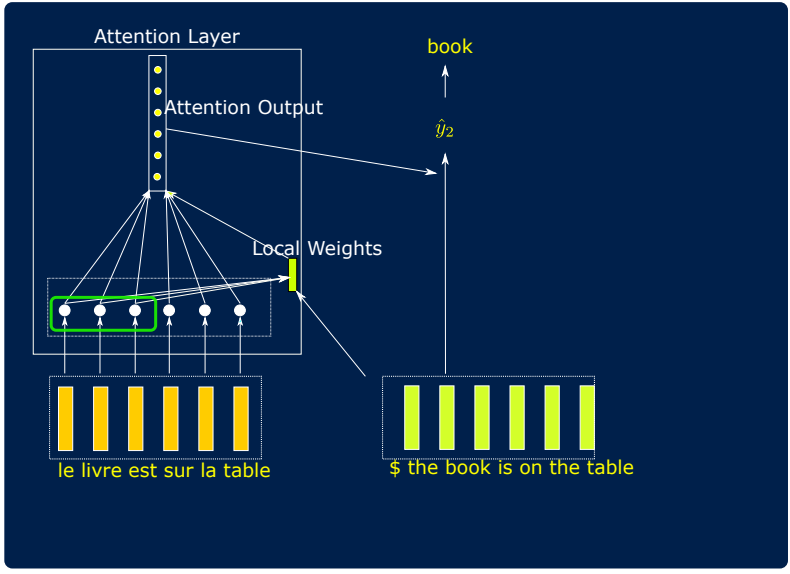
- The alignment is explicitly computed and not latent
- This alignment model is also trained along with the translation model
- α_{ij} is the probability that the target word y_i is aligned to the source x_j
- c_i is the expected annotation over all possible annotations α_{ij}
- α_{ij} or e_{ij} reflects 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

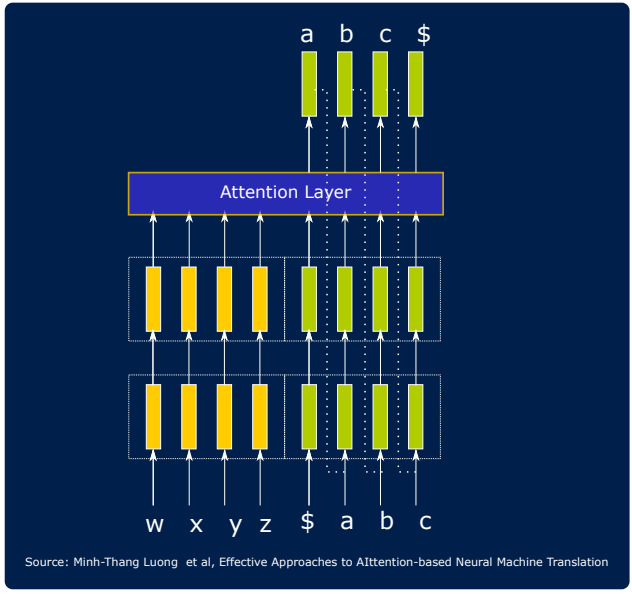

$c_i = \sum_{j=1}^{T} a_{i,j} h_j$
the
book
$\hat{y}_1$
$\hat{y}_2$
$a_{i,j} = softmax(e_{i,j})$
Dot product
c_i
$a_{i,j}$
v_a^T
$e_{i,j}$
W_1
W_2
$e_{i,j} = a(s_{i-1}, h_j)$
$ the book is on the table
$e_{i,j}$−attention score
$a_{i,j}$−attention distribution
c_i−attention output
a − Attention scoring Function
$e_{i,j} = v_a^T tanh(W_1 h_j + W_2 s_{i-1})$

TRANSLATION WITH GLOBAL ATTENTION



TRANSLATION WITH LOCAL ATTENTION




a b c $
Attention Layer
w x y z $ a b c
Source: Minh-Thang Luong et al, Effective Approaches to AIttention-based Neural Machine Translation

INFERENCE: INPUTTING A NEW SENTENCE

- The goal is to generate the most probable target sequence $\mathtt{y}^*$ (French) for a new source sentence $\mathtt{x}$ (English).
- The decoding strategy, typically **Beam Search**, determines the sequence quality.

PHASE 1: ENCODING THE SOURCE SENTENCE

1. **Input Tokenization & Embedding:** The English source sentence, e.g., "The cat sat on the mat," is tokenized and each word is converted into its embedding vector.
2. **Bidirectional RNN Run:** The trained Bidirectional Encoder (Bi-GRU/LSTM) processes the sequence.
3. **Output Annotations (H):** The encoder produces a sequence of rich hidden states (annotations) for every input word.

$$\mathrm{H} = (\mathrm{h}_1, \mathrm{h}_2, \ldots, \mathrm{h}_m)$$

   - H serves as the **Encoder Memory** that the decoder will reference at every subsequent step.
4. **Initialization:** The Decoder's initial hidden state (s_0) is set, usually derived from the final encoder states.

PHASE 2: ITERATIVE DECODING WITH ATTENTION I

The process starts with $\hat{y}_0 = \langle \text{SOS} \rangle$ and previous state s_{t-1}.

1. **Decoder Input:** The decoder takes the embedding of the previous predicted word ($\hat{y}_{t-1}$) as input.
2. **Attention Scoring (Alignment):** The decoder's previous state (s_{t-1}) is compared to **every** encoder annotation (h_i) using the additive attention function to get the relevance score $e_{t,i}$.

$$\text{Score } e_{t,i} = v_a^T \tanh(W_a s_{t-1} + U_a h_i)$$

3. **Attention Weights (α):** Scores are normalized using Softmax to produce the attention weights.

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{k=1}^{m} \exp(e_{t,k})}$$

4. **Context Vector Generation (c_t):** The context vector is computed as a weighted sum of the Encoder Annotations H.

$$c_t = \sum_{i=1}^{m} \alpha_{t,i} h_i$$

- c_t is the dynamic context, summarizing the most relevant English words for the current French word.

PHASE 2: ITERATIVE DECODING WITH ATTENTION III

5. **Decoder RNN Update:** The new hidden state s_t is computed using s_{t-1}, $E(\hat{y}_{t-1})$, and the fresh context c_t.

$$s_t = \text{RNN}(s_{t-1}, E(\hat{y}_{t-1}), c_t)$$

6. **Prediction and Selection:** The model predicts the probability distribution P_t over the French vocabulary. The decoding strategy (e.g., Beam Search) selects the next word(s) $\hat{y}_t$ to continue the loop.

PHASE 3: TERMINATION

- The iterative process repeats until the **End-of-Sequence ($\langle \mathbb{EOS} \rangle$)** token is predicted and selected.
- **Final Output:** If using Beam Search, the partial sequence with the highest total cumulative log-probability is output as the final French translation $\mathbb{y}^*$.

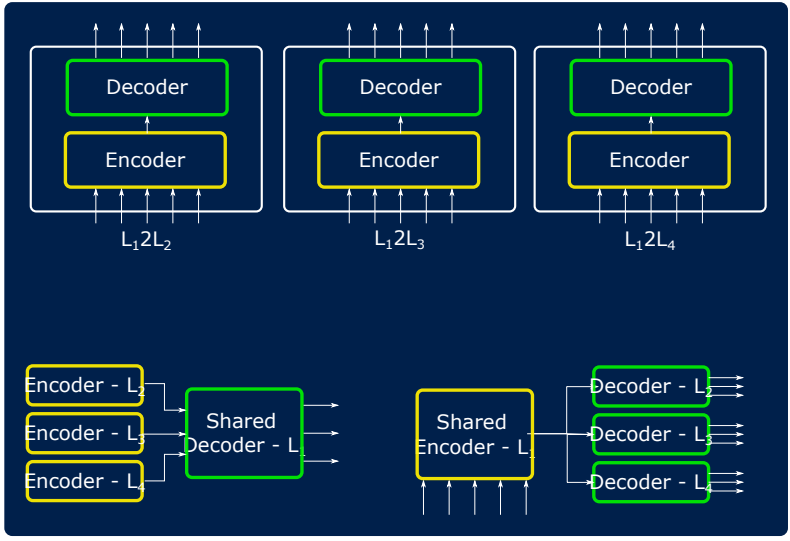
A TYPICAL SETUP

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

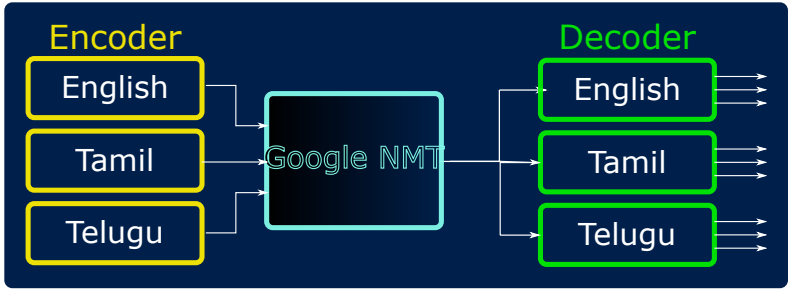


NMT FROM GOOGLE - ZERO-SHOT TRANSLATION

- Moved away from maintaining Seq2Seq model for every pair of languagess
- 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 hat the multilingual NMT system trained on this data could actually generate reasonable Japanese-Korean translations.
  - Is it trained create the Interlingua?
  - Is the system learning a common representation or a translational knowledge?

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

ZERO-SHOT TRANSLATION



FROM RNN ATTENTION TO THE TRANSFORMER

- RNN encoder–decoders with attention (Bahdanau, 2014) established attention as the key ingredient of NMT
- The Transformer (Vaswani et al., 2017, "Attention Is All You Need") removed recurrence entirely. It uses *only* attention plus feed-forward layers
- Benefits: full parallelism over positions, better handling of long-range dependencies, and no vanishing-gradient bottleneck
- Consequences for MT:
  - Transformer-based systems are now the standard for machine translation
  - Massively multilingual models (e.g., NLLB, M2M-100) translate between hundreds of language pairs
  - General-purpose LLMs perform competitive translation via prompting, without a dedicated MT model

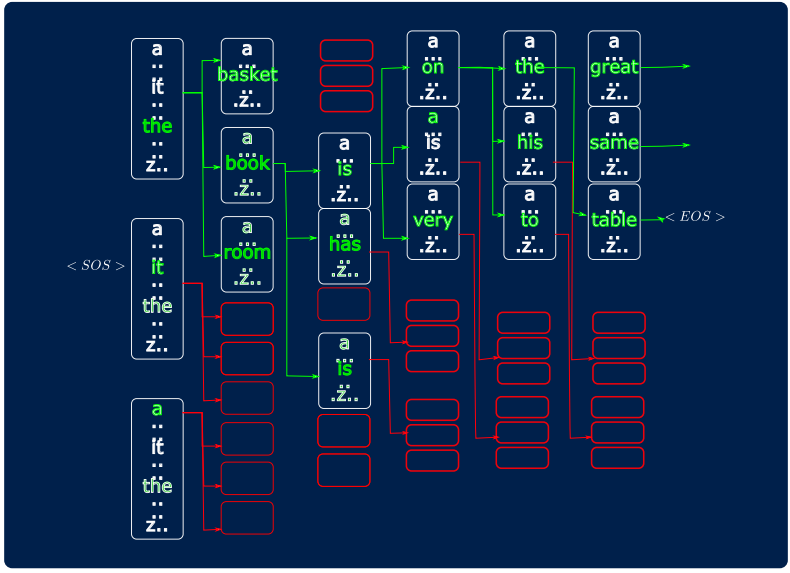
BEAM SEARCH

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
- $\tilde{C} = \bigcup_{i=1}^{M} C_{t-1}^i$

A typical beam width of 5–10 is used in NMT; BLEU scores are comparable across beam widths in this range. (Modern LLM decoding usually prefers sampling methods over beam search for open-ended generation.)

BEAM SEARCH - BEAM WIDTH = 3



Figure

BEAM SEARCH EXAMPLE

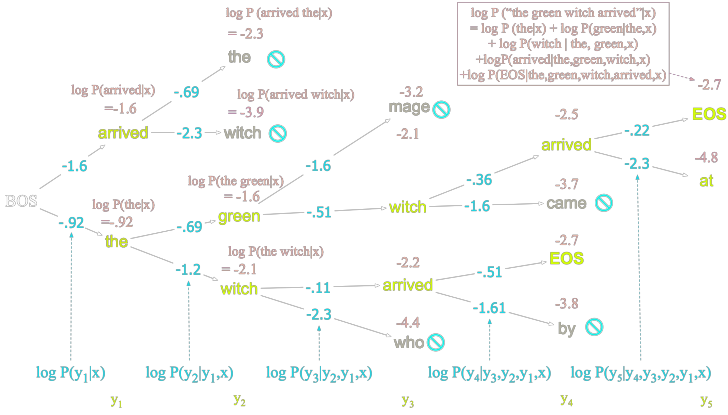


Figure: Scoring for beam search decoding with a beam width of k= 2. We maintain the log probability of each hypothesis in the beam by incrementally adding the logprob of generating each next token. Only the top k paths are extended to the next step

BEAM SEARCH SUMMARY

1. Use all possible partial translations - exhaustive search
2. Beam size, $b = 1$ - greedy search - Words are predicted until the $< EOS >$ is found
3. $b > 1$ - more than one hypotheses
4. Each hypothesis will be produced until the $< EOS >$ is found
5. Each hypothesis will be the translation of the source sentence
6. The length of all hypothesis may not be the same
7. We could use different ***terminate*** conditions
   - Fixed time steps
   - Compute until $< EOS >$ is reached for each hypothesis
8. Use either log probability or product of conditional probability to find the scores for each hypothesis that maximizes
   - $P(y_1, y_2, ...y_m|\mathbf{X}) = \prod_{t=1}^{T} P(y_t| < SOS >, \ldots, y_{t-1}, \mathbf{X})$
   - $\log P(y_1, y_2, ...y_m|\mathbf{X}) = \sum_{t=1}^{T} \log P(y_t| < SOS >, \ldots, y_{t-1}, \mathbf{X})$

DIFFICULTIES WITH HUMAN EVALUATION

- 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 LLM
- Two important aspects required for automatic evaluation
  1. A good metric
  2. A good/gold standards as references

THE IDEA

- Many possible sentences for a given sentence
- A good evaluator 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[3]

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 is a good candidate

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.

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.

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

BLEU EVALUATION

```
from nltk.translate.bleu_score import sentence_bleu
references = [
['It', 'is', 'a', 'guide', 'to', 'action', 'that',
  'ensures', 'that', 'the', 'military', 'will',
  'forever', 'heed', 'Party', 'commands'],
['It', 'is', 'the', 'guiding', 'principle',
 'which', 'guarantees', 'the', 'military',
  'forces', 'always', 'being', 'under',
  'the', 'command', 'of', 'the', 'Party.'],
['It', 'is', 'the', 'practical', 'guide', 'for',
 'the', 'army', 'to', 'heed', 'the',
  'directions', 'of', 'the', 'party.']
]
hypothesis = ['It', 'is', 'a', 'guide', 'to',
 'action', 'which', 'ensures', 'that', 'the',
 'military', 'always', 'obeys', 'the',
 'commands', 'of', 'the', 'party']
score = sentence_bleu(references, hypothesis, weights=(1, 0, 0, 0))
print(score)
```

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

$$\text{Modified unigram precision for C1} = \frac{17}{18} \qquad \text{Modified unigram precision for C2} = \frac{8}{14}$$

C3: the the the the the the the
R4: the cat is on the mat

$$\text{Unigram precision} = \frac{7}{7}$$
$$\text{Modified unigram precision} = \frac{2}{7}$$
$$\text{Modified bigram precision} = 0$$

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 $= \frac{10}{17}$

```
(It,is),(is,a),(a,guide),(guide,to),
(to,action),(action,that),(that,ensures),
(ensures,that),(that,the),(the,military),
(military,will),(will,forever),(forever,heed),
(heed,Party),(Party,commands)

(It,is),(is,the),(the,guiding),
(guiding,principle),(principle,which),
(which,guarantees),(guarantees,the),
(the,military),(military,forces),(forces,always),
(always,being),(being,under),(under,the),
(the,command),(command,of),
(of,the),(the,Party)

(It,is),(is,the),(the,practical),(practical,guide),
(guide,for),(for,the),(the,army),
(army,always),(always,to),(to,heed),
(heed,the),(the,directions),
(directions,of),(of,the),(the,party)
^^I^^I
```

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 $= \frac{1}{13}$

```
(It,is),(is,a),(a,guide),(guide,to),
(to,action),(action,that),(that,ensures),
(ensures,that),(that,the),(the,military),
(military,will),(will,forever),(forever,heed),
(heed,Party),(Party,commands)

(It,is),(is,the),(the,guiding),
(guiding,principle),(principle,which),
(which,guarantees),(guarantees,the),
(the,military),(military,forces),(forces,always),
(always,being),(being,under),(under,the),
(the,command),(command,of),
(of,the),(the,Party)

(It,is),(is,the),(the,practical),(practical,guide),
(guide,for),(for,the),(the,army),
(army,always),(always,to),(to,heed),
(heed,the),(the,directions),
(directions,of),(of,the),(the,party)
^^I^^I
```

COMBINING N-GRAM PRECISIONS

- Modified n-gram precisions decay exponentially as n increases[3]
- BLEU averages the log-precisions with uniform weights (the geometric mean of the modified n-gram precisions) to handle this decay
- A short candidate ($c < r$) can inflate precision, so a brevity penalty (BP) is applied 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

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{19}$$

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 $w_n = \frac{1}{N}$

$$p_n = \frac{\sum_{c \in C} \sum_{ngrams \in C} \text{Count}_{\text{clip}}(ngrams)}{\sum_{c \in C} \sum_{ngrams \in C} \text{Count}(ngrams)} \tag{20}$$

BLEU - DEMO

BLEU Demo

APPLICATIONS OF BLEU

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 at the sentence level
- Embedding / model-based metrics (increasingly preferred for fluent LLM output)
  - BERTScore measures token similarity using contextual embeddings
  - BLEURT and COMET are learned metrics trained to match human judgments
  - LLM-as-a-judge prompts a strong LLM to rate or compare outputs

REFERENCES I

[1] 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.

[2] KyungHyun Cho et al. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *CoRR* abs/1409.1259 (2014). arXiv: 1409.1259. URL: http://arxiv.org/abs/1409.1259.

[3] 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 Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://www.aclweb.org/anthology/P02-1040.