CSE440: Natural Language Processing II

Dr. Farig Sadeque
Associate Professor
Department of Computer Science and Engineering
BRAC University

Lecture 7: Translation

Outline

- Probabilistic Translation (Lecture)
- Seq2seq model (Book chapter 13)
- Attention mechanism (Book chapter 13)
- Translation issues (Book chapter 13)

Probabilistic Translation

Goal:

- Get the most probable English sentence given a French sentence
  - argmax $P(e|f)$

Using Bayes Theorem:

$$\arg\max P(e|f) = \arg\max (P(e)*P(f|e))$$

Notation:

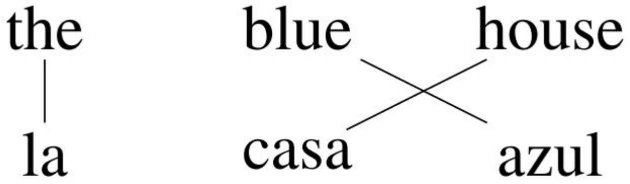
- $P(e)$: probability of English sentence e (Do the words follow English order?)
- $P(f|e)$: probability that, given an English sentence e, a translator produces French sentence f (Are the words good translations?)

How to get P(e): Language Modeling

- A language model estimates P(e), the probability that a sentence e is an English sentence.
- Many language modeling techniques exist:
    - n-gram language models (e.g., SRILM, KenLM)
        - Assumption: a sentence is a bag of overlapping n-grams
    - Neural language models (RNNs, etc.)
        - Assumption: a sentence is a sequence of words
- All such models are trained on huge, unlabeled data

How to get P(f|e): Translation Modeling

- Many translation models incorporate some form of alignment indicating which words were translated as which. We will follow IBM Model 1.
- An example alignment:



- IBM Model 1 calculates translation probability as:

$$P(f|e) = \sum_{a} P(a, f|e) = \sum_{a} \prod_{(e_i, f_j) \in a} P(f_j|e_i)$$

IBM Model 1

$$P(f|e) = \sum_a P(a, f|e) = \sum_a \prod_{(e_i, f_j) \in a} P(f_j|e_i)$$

Intuition:

- Consider all possible word alignments
- Combine the word-translation probabilities

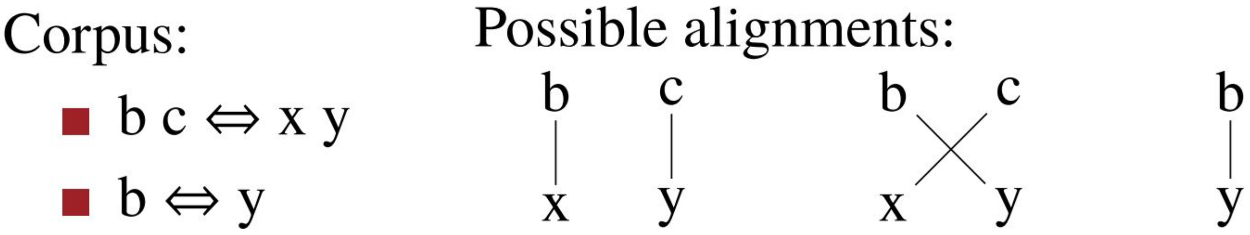
We need to estimate $P(f_j|e_i)$: For each English word ei, the probability of it being translated as French word fj

- If we had a corpus of word-level alignments, we could just count and divide.
- We typically have only sentence translations. How do we get the word translations?
- The expectation maximization (EM) algorithm

Expectation-maximization algorithm

Expectation maximization:

1. Start with uniform estimates of word-word translations
2. Use word-word translation probabilities to estimate alignment probabilities
3. Use alignment probabilities to estimate word-word translation probabilities
4. Go to 2



EM Walkthrough

1. Start with uniform estimates of word-word translations

Words:

$$P(x|b) =$$

$$P(y|b) =$$

$$P(x|c) =$$

$$P(y|c) =$$

Alignments:

$$P\left(\begin{array}{cc} \mathrm{b} & \mathrm{c} \\ | & | \\ \mathrm{x} & \mathrm{y} \end{array}, f|e\right) =$$

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ & \times & \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) =$$

$$P\left(\begin{array}{c} \mathrm{b} \\ | \\ \mathrm{y} \end{array}, f|e\right) =$$

EM Walkthrough

1. Start with uniform estimates of word-word translations

Words:

$$P(x|b) = \frac{1}{2}$$

$$P(y|b) = \frac{1}{2}$$

$$P(x|c) = \frac{1}{2}$$

$$P(y|c) = \frac{1}{2}$$

Alignments:

$$P\begin{pmatrix} \mathrm{b} & & \mathrm{c} \\ | & & | \\ \mathrm{x} & & \mathrm{y} \end{pmatrix}, f|e \Bigg) =$$

$$P\begin{pmatrix} \mathrm{b} & & \mathrm{c} \\ & \times & \\ \mathrm{x} & & \mathrm{y} \end{pmatrix}, f|e \Bigg) =$$

$$P\begin{pmatrix} \mathrm{b} \\ | \\ \mathrm{y} \end{pmatrix}, f|e \Bigg) =$$

EM Walkthrough

2. For each alignment, compute $P(a, f|e) = \prod_{(e_i, f_j) \in a} P(f_j|e_i)$

Words:

$$P(x|b) = \frac{1}{2}$$

$$P(y|b) = \frac{1}{2}$$

$$P(x|c) = \frac{1}{2}$$

$$P(y|c) = \frac{1}{2}$$

Alignments:

$$P\left(\begin{array}{ccc} b & & c \\ | & & | \\ x & & y \end{array}, f|e\right) =$$

$$P\left(\begin{array}{ccc} b & & c \\ & \times & \\ x & & y \end{array}, f|e\right) =$$

$$P\left(\begin{array}{c} b \\ | \\ y \end{array}, f|e\right) =$$

EM Walkthrough

2. For each alignment, compute $P(a, f|e) = \prod_{(e_i, f_j) \in a} P(f_j|e_i)$

Words:

$$P(x|b) = \frac{1}{2}$$

$$P(y|b) = \frac{1}{2}$$

$$P(x|c) = \frac{1}{2}$$

$$P(y|c) = \frac{1}{2}$$

Alignments:

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ | & & | \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{4}$$

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ & \times & \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{4}$$

$$P\left(\begin{array}{c} \mathrm{b} \\ | \\ \mathrm{y} \end{array}, f|e\right) = \frac{1}{2}$$

EM Walkthrough

3. Normalize so that each sentence sums to 1.

Words:

$$P(x|b) = \frac{1}{2}$$

$$P(y|b) = \frac{1}{2}$$

$$P(x|c) = \frac{1}{2}$$

$$P(y|c) = \frac{1}{2}$$

Alignments:

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ | & & | \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{4}$$

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ & \times & \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{4}$$

$$P\left(\begin{array}{c} \mathrm{b} \\ | \\ \mathrm{y} \end{array}, f|e\right) = \frac{1}{2}$$

EM Walkthrough

2.1. Normalize so that each sentence sums to 1.

Words:

$$P(x|b) = \frac{1}{2}$$

$$P(y|b) = \frac{1}{2}$$

$$P(x|c) = \frac{1}{2}$$

$$P(y|c) = \frac{1}{2}$$

Alignments:

$$P\left(\begin{matrix} \mathrm{b} & & \mathrm{c} \\ | & & | \\ \mathrm{x} & & \mathrm{y} \end{matrix}, f|e\right) = \frac{1}{2}$$

$$P\left(\begin{matrix} \mathrm{b} & & \mathrm{c} \\ & \times & \\ \mathrm{x} & & \mathrm{y} \end{matrix}, f|e\right) = \frac{1}{2}$$

$$P\left(\begin{matrix} \mathrm{b} \\ | \\ \mathrm{y} \end{matrix}, f|e\right) = 1$$

EM Walkthrough

3. Collect fractional counts for word-word translations

Words:

$$P(x|b) = \frac{1}{2}$$

$$P(y|b) = \frac{1}{2}$$

$$P(x|c) = \frac{1}{2}$$

$$P(y|c) = \frac{1}{2}$$

Alignments:

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ | & & | \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{2}$$

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ & \times & \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{2}$$

$$P\left(\begin{array}{c} \mathrm{b} \\ | \\ \mathrm{y} \end{array}, f|e\right) = 1$$

EM Walkthrough

3. Collect fractional counts for word-word translations

Words:

$$P(x|b) = \frac{1}{2}$$

$$P(y|b) = \frac{3}{2}$$

$$P(x|c) = \frac{1}{2}$$

$$P(y|c) = \frac{1}{2}$$

Alignments:

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ | & & | \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{2}$$

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ & \times & \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{2}$$

$$P\left(\begin{array}{c} \mathrm{b} \\ | \\ \mathrm{y} \end{array}, f|e\right) = 1$$

EM Walkthrough

3.1. Normalize so that each word sums to 1

Words:

$$P(x|b) = \frac{1}{2}$$

$$P(y|b) = \frac{3}{2}$$

$$P(x|c) = \frac{1}{2}$$

$$P(y|c) = \frac{1}{2}$$

Alignments:

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ | & & | \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{2}$$

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ & \times & \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{2}$$

$$P\left(\begin{array}{c} \mathrm{b} \\ | \\ \mathrm{y} \end{array}, f|e\right) = 1$$

EM Walkthrough

3.1. Normalize so that each word sums to 1

Words:

$$P(x|b) = \frac{1}{4}$$

$$P(y|b) = \frac{3}{4}$$

$$P(x|c) = \frac{1}{2}$$

$$P(y|c) = \frac{1}{2}$$

Alignments:

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ | & & | \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{2}$$

$$P\left(\begin{array}{ccc} \mathrm{b} & & \mathrm{c} \\ & \times & \\ \mathrm{x} & & \mathrm{y} \end{array}, f|e\right) = \frac{1}{2}$$

$$P\left(\begin{array}{c} \mathrm{b} \\ | \\ \mathrm{y} \end{array}, f|e\right) = 1$$

EM Walkthrough

- Continue
- What will happen if we keep repeating this process?

Decoding/prediction

How do we generate e sentences for the argmax?

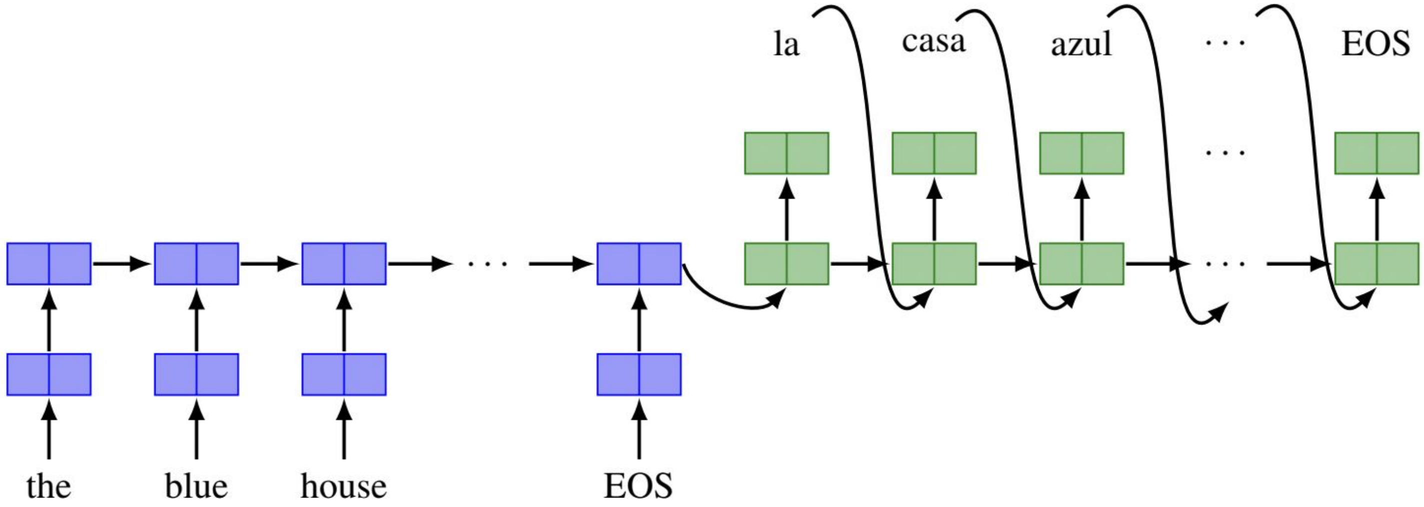
Build translation left to right:

- Randomly select foreign word to be translated
- Find possible English word translation
- Add English word to end of partial translation
- Mark foreign word as translated

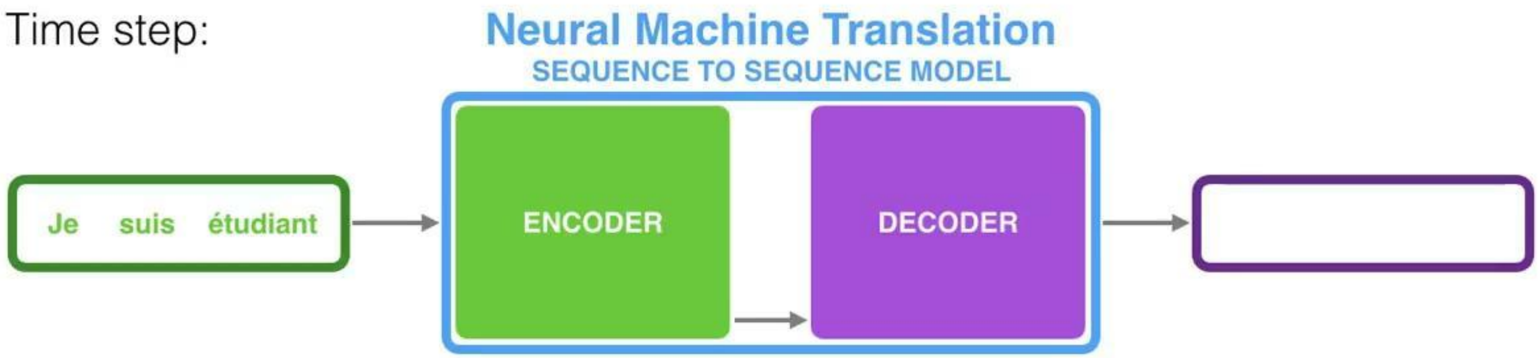
Both steps 1 and 2 have many possibilities: use AI search techniques to explore the space

Neural Machine Translation: Seq2Seq Model

- MT model using RNN

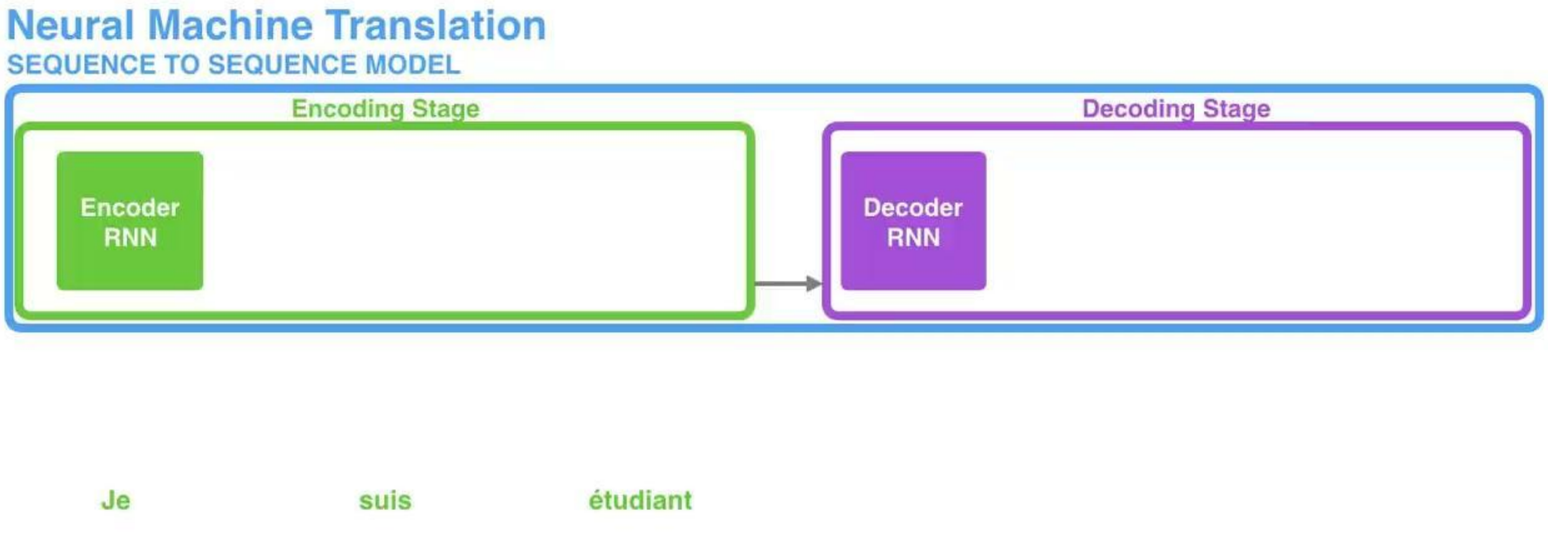


MT model using encoder-decoder



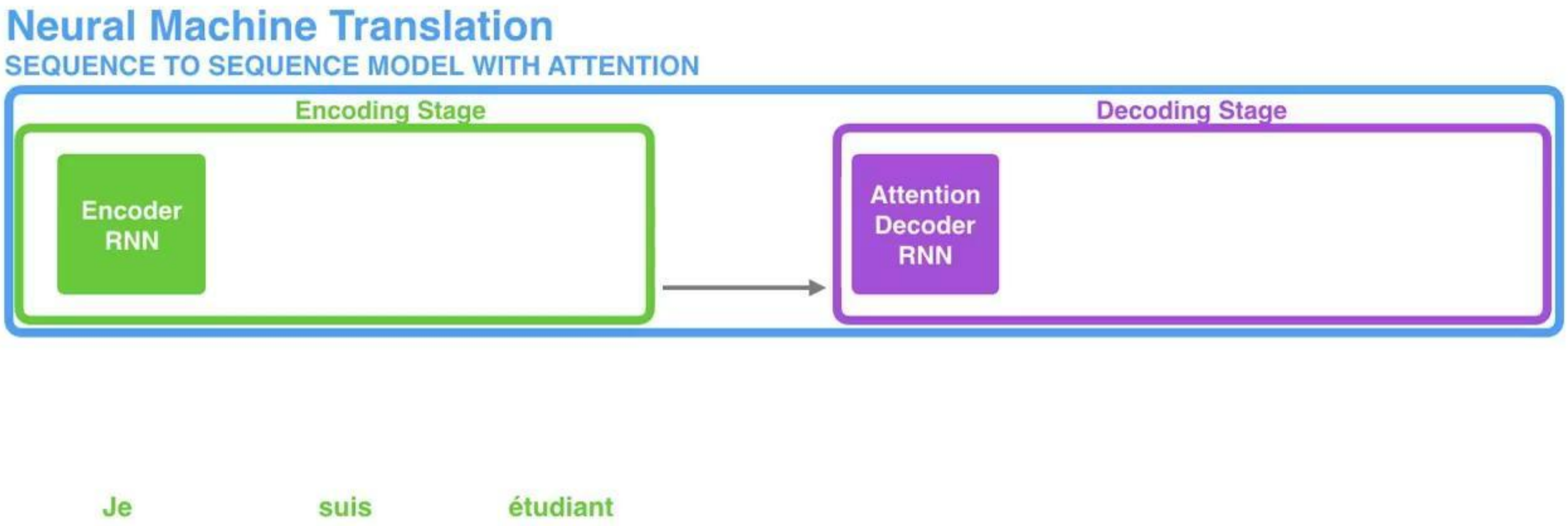
Video Courtesy: Jay Alammar

MT model using encoder-decoder



Video Courtesy: Jay Alammar

MT model using encoder-decoder

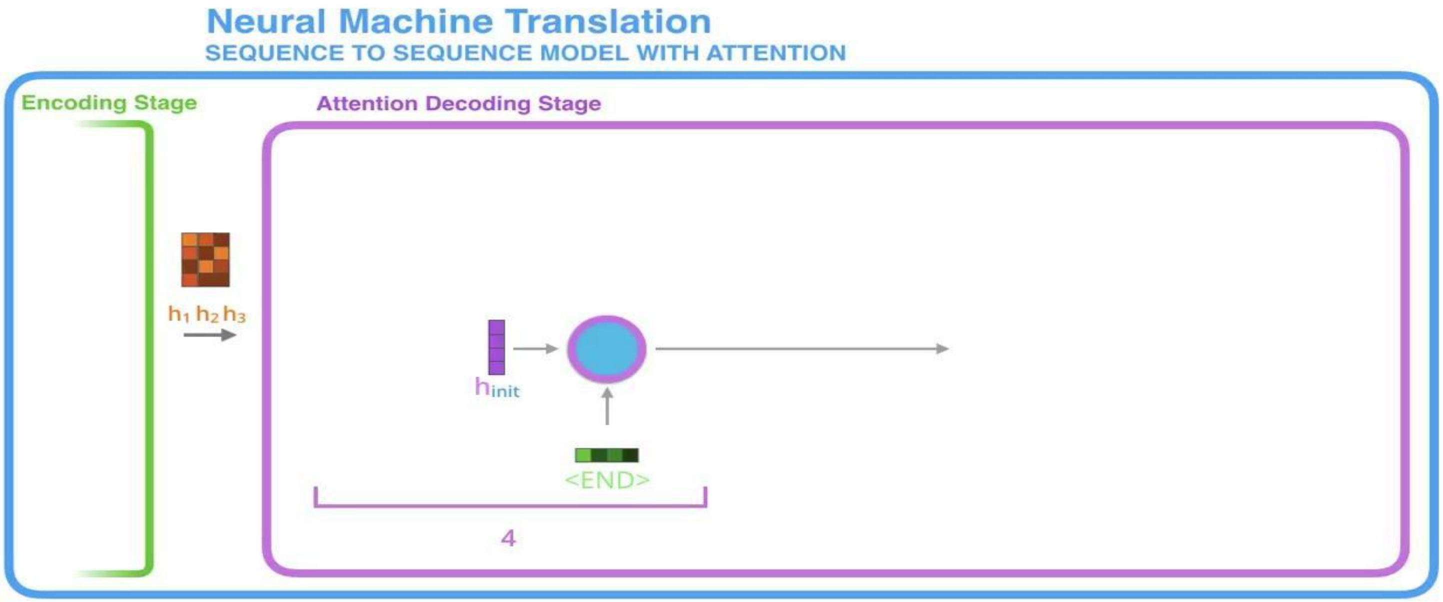


Video Courtesy: Jay Alammar

MT model using encoder-decoder

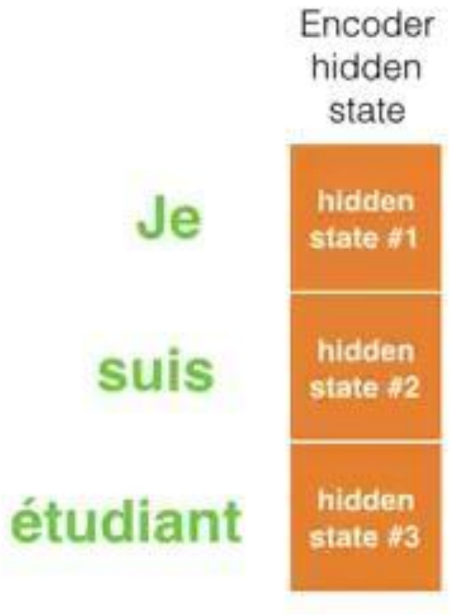
Video Courtesy: Jay Alammar

MT model using encoder-decoder



Video Courtesy: Jay Alammar

MT model using encoder-decoder



Video Courtesy: Jay Alammar

But…

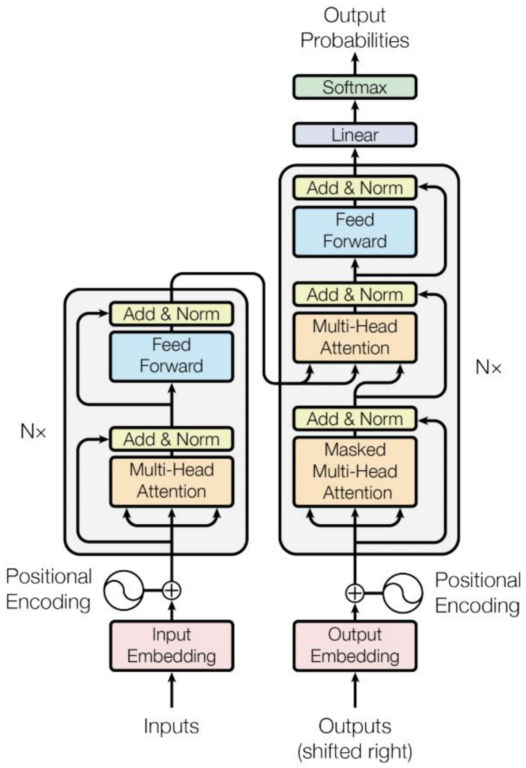
RNN-based encoder decoder works well, but:

- Backpropagation through time and infinite memory is still an issue, even with gated RNNs
- RNNs work sequentially, so parallelization is a challenge

Why do we need GPUs?

- CPUs are latency-optimized, GPUs are bandwidth-optimized
    - The more memory your computational operations require, the more significant the advantage of GPUs over CPUs: matrix multiplication requires more computational operations
- More computing units: better thread parallelism, can hide latency issues
- Faster access to RAMs (VRAMs)
- DNN computations just fit well with GPU architecture
    - Many identical neurons, doing the same computation

Transformer



We will see:

- Self-attention
- Multi-head attention
- Positional encoding
- Byte-pair encoding

Image Courtesy: Attention is all you need

High level view of a transformer

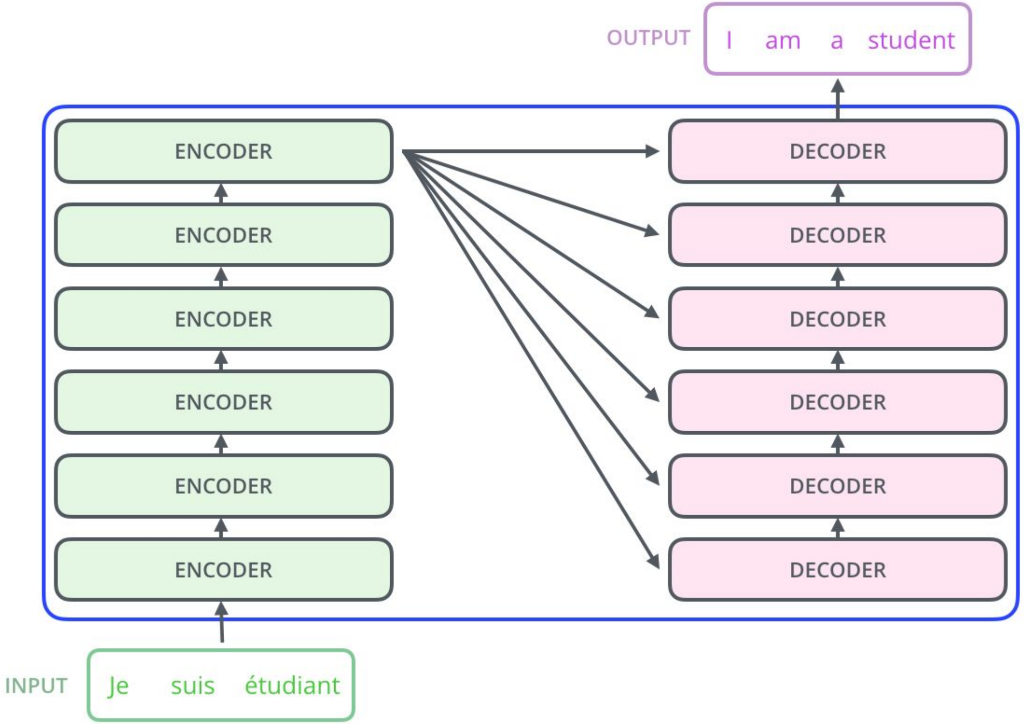


Image Courtesy: Jay Alammar

Training a transformer

- Training a transformer is exceptionally difficult
- Not enough hardware, not enough time
- What to do?

Pretraining

- Transformers can be trained to learn through language representations
- Someone with enough hardware and time can (pre)train a transformer that can learn that language representation, and then we can use that representation for our NLP task

BERT

- BERT = Bidirectional Encoder Representations from Transformers
- Designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers
- Pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models
- Achieved state-of-the-art performance in almost all tasks when it came out
- Uses WordPiece tokenization
- Are massive
  - Base model has 12 encoders, 12 attention heads, 768 hidden units, large model has 24 encoders, 16 heads and 1024 hidden units
  - Base models counts up to 110 million parameters, large has 340 mils.

BERT

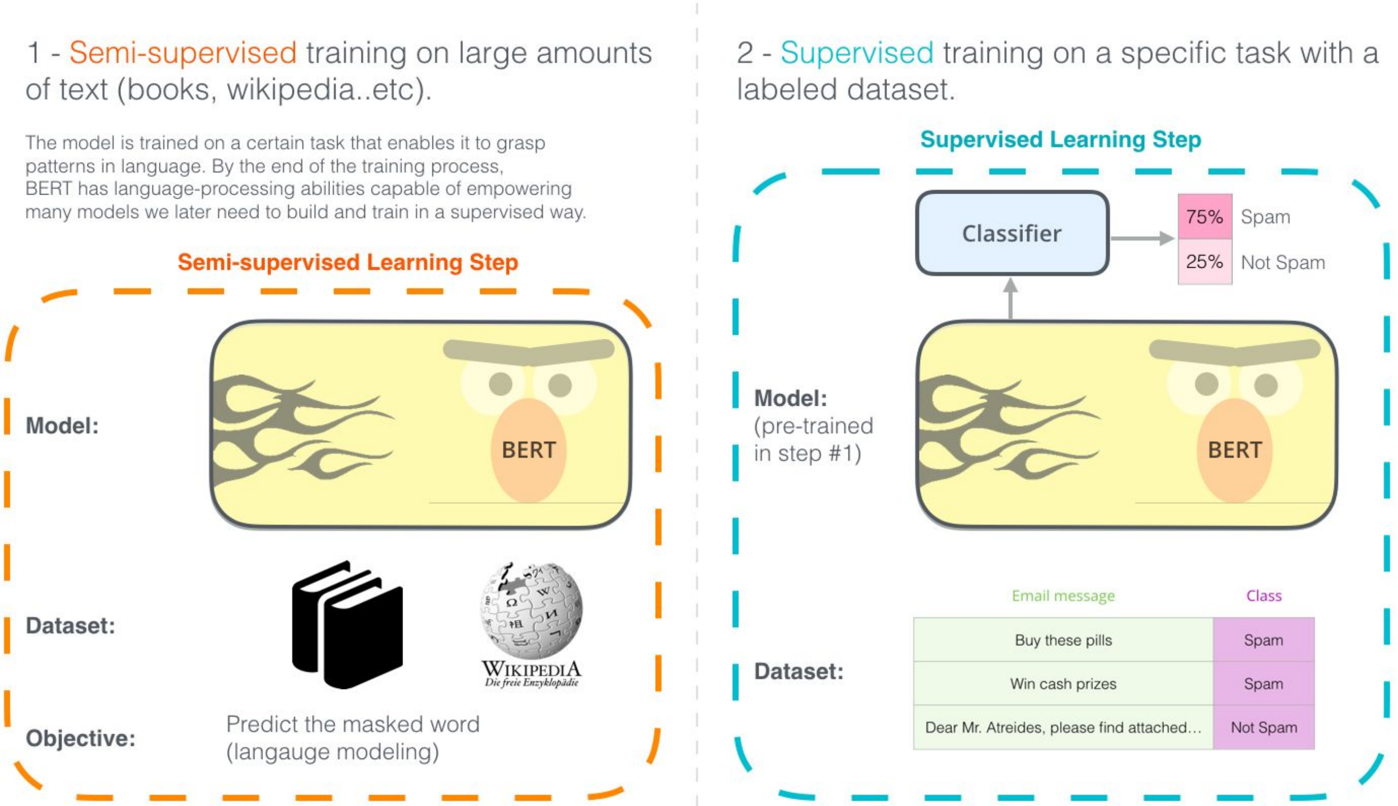


Image Courtesy: Jay Alammar

BERT

- Pretrained on two unsupervised task
    - Masked language modeling
    - Next sentence prediction
- Masked LM
    - Chooses 15% of tokens at random
    - Replaces a token with a [MASK] 80% of the time, with a random token 10% of the time and does not replace 10% of the time
- Next sentence prediction
    - Can model tasks that are not covered by language modeling (QA, inference)
    - Training data includes: 50% of the time B is the actual next sentence that follows A and 50% of the time it is a random sentence from the corpus

How to use BERT

- Task specific-Models (fine-tuning)
- Feature extraction
  - But which one should we use?

Image Courtesy: Jay Alammar