

# 7. Machine Translation, Seq2seq

LING-581-Natural Language Processing 1

---

Instructor: Hakyung Sung  
October 2, 2025

\*Acknowledgment: These course slides are based on materials from CS224N @ Stanford University

# Table of contents

1. Machine translation
2. Neural machine translation
3. Wrap-up
4. Review: Dependency parser training

# Review

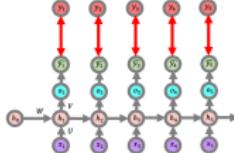
---

# Review

- RNNs
- Problems with RNNs
- LSTMs
- Bidirectional RNNs

# Review: Problems with RNNs

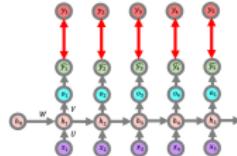
$$\begin{aligned} & \frac{\partial L_0}{\partial W} \frac{\partial L_0}{\partial y_0^c} \frac{\partial y_0^c}{\partial o_0} \frac{\partial o_0}{\partial h_0} + \frac{\partial L_0}{\partial y_0^c} \frac{\partial y_0^c}{\partial o_2} \frac{\partial o_2}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L_0}{\partial y_0^c} \frac{\partial y_0^c}{\partial o_4} \frac{\partial o_4}{\partial h_4} \frac{\partial h_4}{\partial W} \\ & + \frac{\partial L_0}{\partial y_0^c} \frac{\partial y_0^c}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_2} \frac{\partial h_2}{\partial W} - \frac{\partial L_0}{\partial y_0^c} \frac{\partial y_0^c}{\partial o_6} \frac{\partial o_6}{\partial h_6} \frac{\partial h_6}{\partial h_2} \frac{\partial h_2}{\partial W} \end{aligned}$$



- RNNs are equivalent to a deep network of depth  $T$  when unrolled over time ( $T$  = sequence length/time steps)

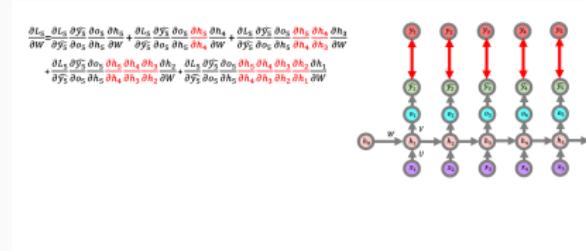
# Review: Problems with RNNs

$$\begin{aligned} & \frac{\partial L_0}{\partial W} \frac{\partial L_0}{\partial y_0^c} \frac{\partial y_0^c}{\partial o_0} \frac{\partial o_0}{\partial h_0} + \frac{\partial L_0}{\partial y_0^c} \frac{\partial y_0^c}{\partial o_2} \frac{\partial o_2}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L_0}{\partial y_0^c} \frac{\partial y_0^c}{\partial o_4} \frac{\partial o_4}{\partial h_4} \frac{\partial h_4}{\partial W} \\ & + \frac{\partial L_0}{\partial y_0^c} \frac{\partial y_0^c}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial W} - \frac{\partial L_1}{\partial y_1^c} \frac{\partial y_1^c}{\partial o_0} \frac{\partial o_0}{\partial h_0} \frac{\partial h_0}{\partial h_2} \frac{\partial h_2}{\partial W} \\ & - \frac{\partial L_1}{\partial y_1^c} \frac{\partial y_1^c}{\partial o_2} \frac{\partial o_2}{\partial h_2} \frac{\partial h_2}{\partial h_4} \frac{\partial h_4}{\partial W} - \frac{\partial L_1}{\partial y_1^c} \frac{\partial y_1^c}{\partial o_4} \frac{\partial o_4}{\partial h_4} \frac{\partial h_4}{\partial h_6} \frac{\partial h_6}{\partial W} \end{aligned}$$



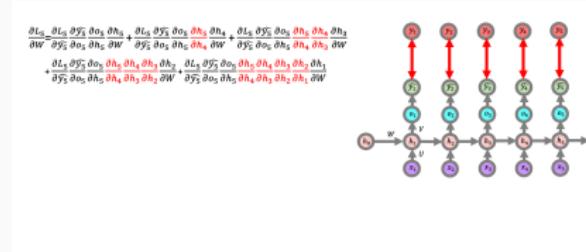
- RNNs are equivalent to a deep network of depth  $T$  when unrolled over time ( $T = \text{sequence length/time steps}$ )
- Parameter sharing:** the same weight matrices are multiplied at each time step.

# Review: Problems with RNNs



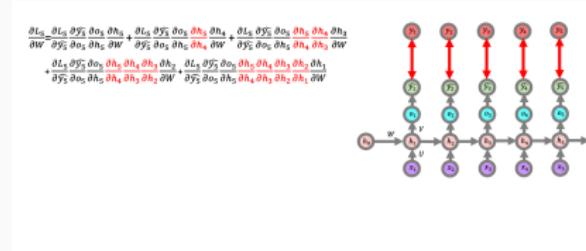
- RNNs are equivalent to a deep network of depth  $T$  when unrolled over time ( $T$  = sequence length/time steps)
- Parameter sharing:** the same weight matrices are multiplied at each time step.
- If each multiplication factor:

# Review: Problems with RNNs



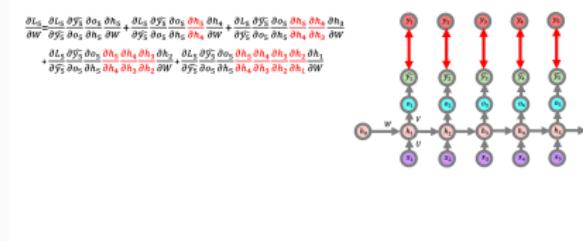
- RNNs are equivalent to a deep network of depth  $T$  when unrolled over time ( $T$  = sequence length/time steps)
- **Parameter sharing:** the same weight matrices are multiplied at each time step.
- If each multiplication factor:
  - is  $< 1$ , gradients shrink exponentially (vanishing).

# Review: Problems with RNNs



- RNNs are equivalent to a deep network of depth  $T$  when unrolled over time ( $T$  = sequence length/time steps)
- Parameter sharing:** the same weight matrices are multiplied at each time step.
- If each multiplication factor:
  - is  $< 1$ , gradients shrink exponentially (vanishing).
  - is  $> 1$ , gradients grow exponentially (exploding).

# Review: Problems with RNNs

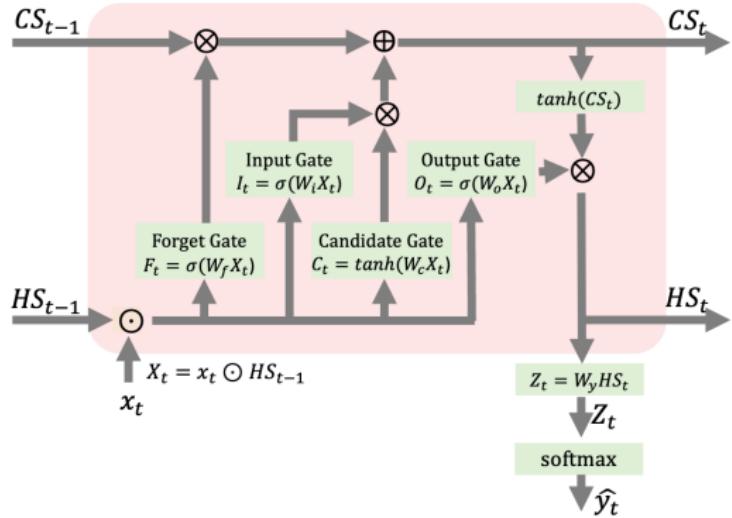


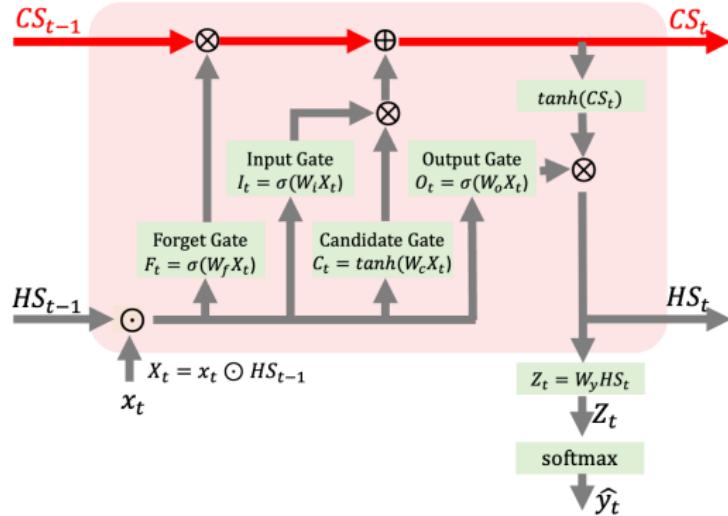
- RNNs are equivalent to a deep network of depth  $T$  when unrolled over time ( $T = \text{sequence length/time steps}$ )
- **Parameter sharing:** the same weight matrices are multiplied at each time step.
- If each multiplication factor:
  - is  $< 1$ , gradients shrink exponentially (vanishing).
  - is  $> 1$ , gradients grow exponentially (exploding).
- Standard feedforward nets have limited depth, so this extreme behavior is less pronounced.

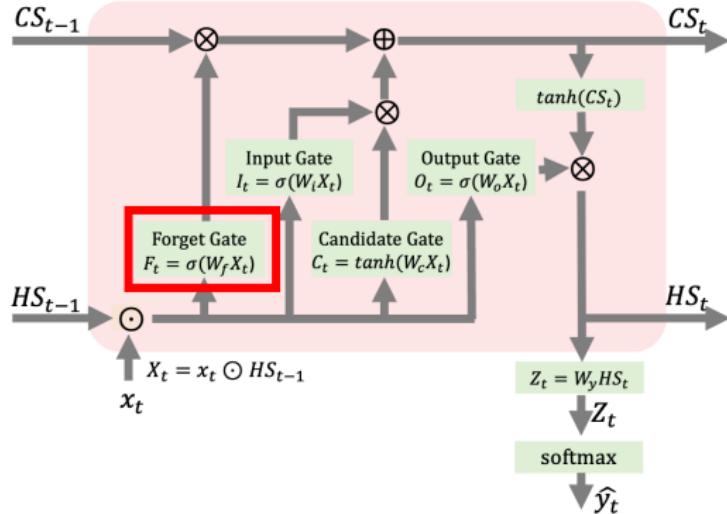
## Vanishing problem: Solution

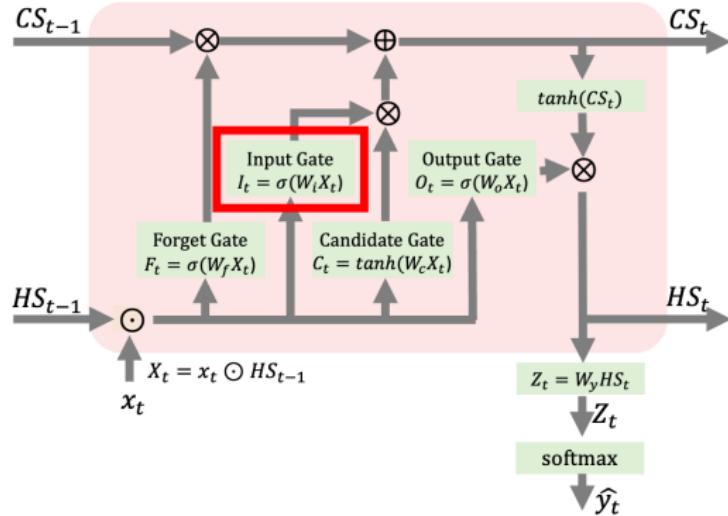
Separate memory cell with gating mechanisms to  
add/erase information.

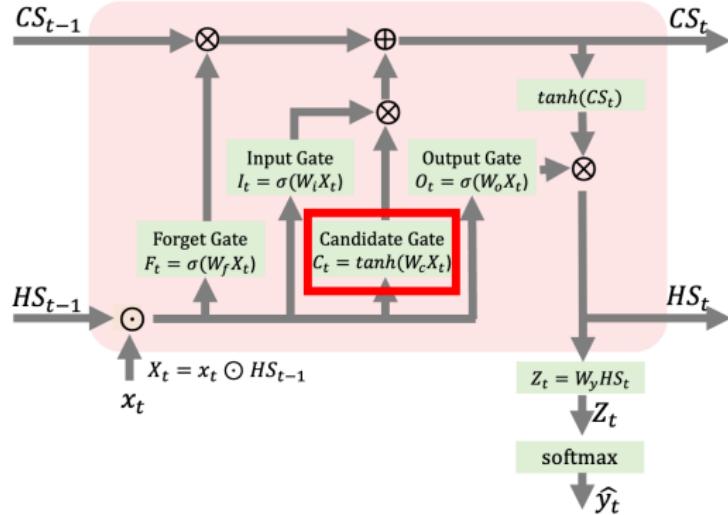
# Review: LSTMs

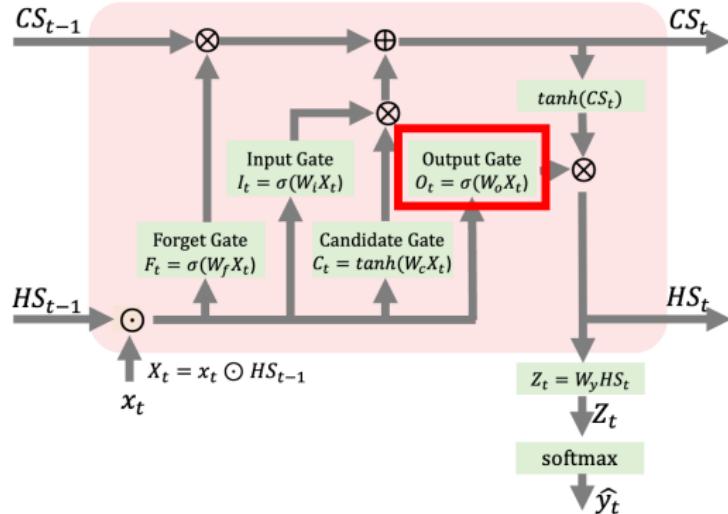








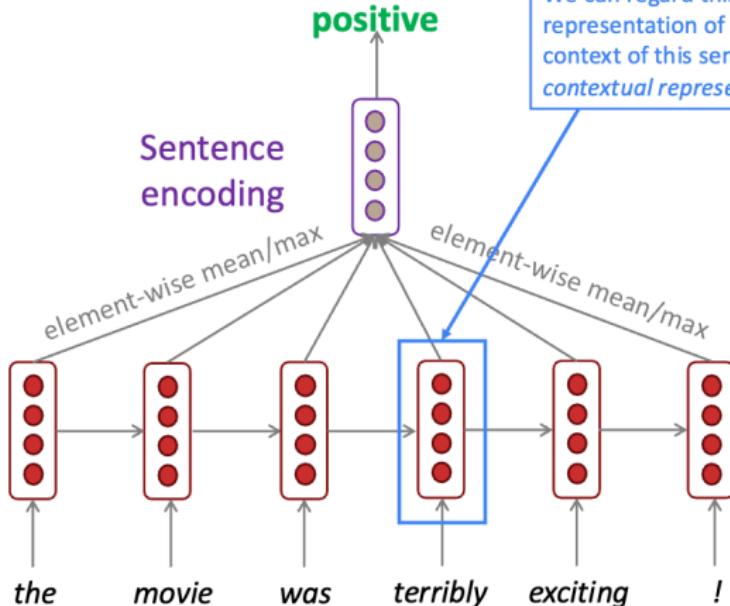




## Review: Bidirectional RNNs

- A standard RNN only uses past context.
- Bidirectional RNNs process the sequence in both directions.

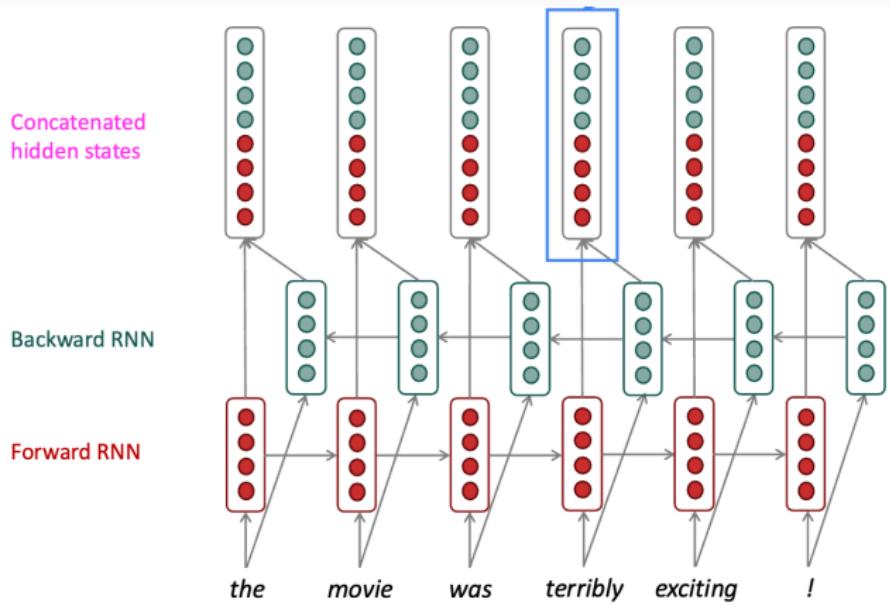
## Task: Sentiment Classification



These contextual representations only contain information about the *left* context (e.g. "the movie was").

What about *right* context?

In this example, "exciting" is in the right context and this modifies the meaning of "terribly" (from negative to positive)

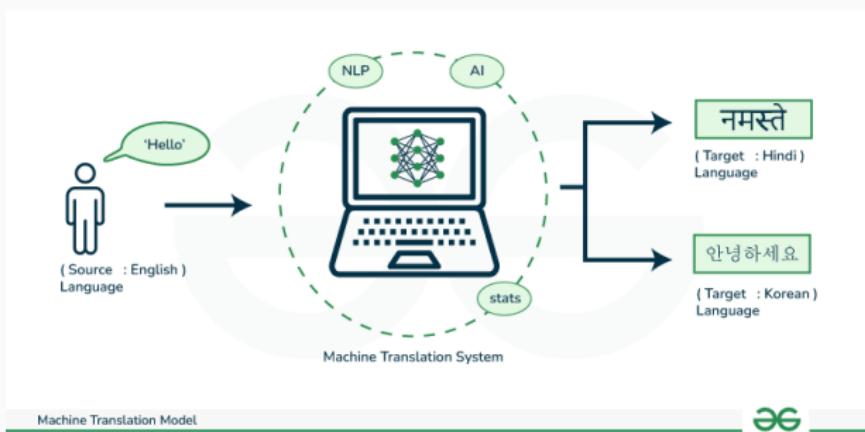


## Machine translation

---

# Pre-neural machine translation

- Machine Translation (MT) is the task of translating a sentence  $x$  from one language (**source language**) to a sentence  $y$  in another language (**target language**)



Source: <https://www.geeksforgeeks.org/nlp/machine-translation-of-languages-in-artificial-intelligence/>

## The early history of MT: 1950s

- Machine translation research began in the early 1950s on machines less powerful than high school calculators (before the term AI was coined)

## The early history of MT: 1950s

- Machine translation research began in the early 1950s on machines less powerful than high school calculators (before the term AI was coined)
- Concurrent with foundational work on automata, formal languages, probabilities, and information theory

## The early history of MT: 1950s

- Machine translation research began in the early 1950s on machines less powerful than high school calculators (before the term **AI** was coined)
- Concurrent with foundational work on automata, formal languages, probabilities, and information theory
- MT heavily funded by military, but basically just **simple rule-based systems** doing word substitution

## The early history of MT: 1950s

- Machine translation research began in the early 1950s on machines less powerful than high school calculators (before the term **AI** was coined)
- Concurrent with foundational work on automata, formal languages, probabilities, and information theory
- MT heavily funded by military, but basically just **simple rule-based systems** doing word substitution
- Human language is more complicated than that, and varies more across languages

## The early history of MT: 1950s

- Machine translation research began in the early 1950s on machines less powerful than high school calculators (before the term **AI** was coined)
- Concurrent with foundational work on automata, formal languages, probabilities, and information theory
- MT heavily funded by military, but basically just **simple rule-based systems** doing word substitution
- Human language is more complicated than that, and varies more across languages
- Little understanding of natural language syntax, semantics, pragmatics ... problem soon appeared intractable...

## 1990s-2010s: Statistical machine translation (SMT)

- **Idea:** Learn a probabilistic model of translation from data.

## 1990s-2010s: Statistical machine translation (SMT)

- **Idea:** Learn a probabilistic model of translation from data.
- Example: Translating from French → English.

## 1990s-2010s: Statistical machine translation (SMT)

- **Idea:** Learn a probabilistic model of translation from data.
- Example: Translating from French → English.
- Goal: Find the best English  $\mathbf{y}$ , given a French  $\mathbf{x}$ :

$$\operatorname{argmax}_y P(y \mid x)$$

## 1990s-2010s: Statistical machine translation (SMT)

- **Idea:** Learn a probabilistic model of translation from data.
- Example: Translating from French → English.
- Goal: Find the best English  $\mathbf{y}$ , given a French  $\mathbf{x}$ :

$$\operatorname{argmax}_y P(y | x)$$

- Directly modeling  $P(y | x)$  is difficult!

## 1990s-2010s: SMT

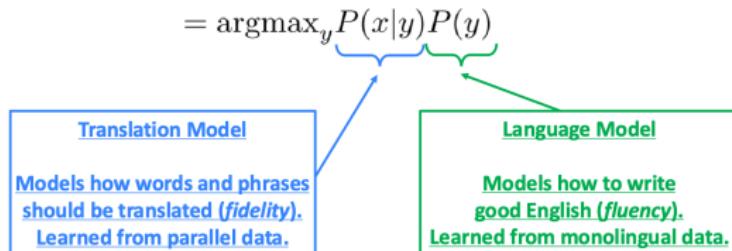
- Using Bayes' Theorem:

$$P(y | x) = \frac{P(x | y) P(y)}{P(x)}$$

- Since  $P(x)$  is fixed (bc we cannot change the input), we can rewrite the search as:

$$\operatorname{argmax}_y P(x | y) \cdot P(y)$$

- This gives two components to be learned separately:
  - Translation Model:  $P(x | y)$
  - Language Model:  $P(y)$



## 1990s–2010s: SMT

---

- How do we build a language model?

## 1990s–2010s: SMT

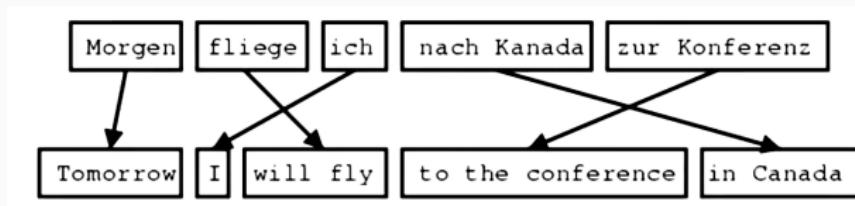
- How do we build a language model?
- How can we learn a translation model  $P(x \mid y)$ ?

## 1990s–2010s: SMT

- How do we build a language model?
- How can we learn a translation model  $P(x | y)$ ?
- Requirement: A large amount of **parallel data** (e.g., pairs of human-translated French/English sentences)

# Learning alignment of SMT

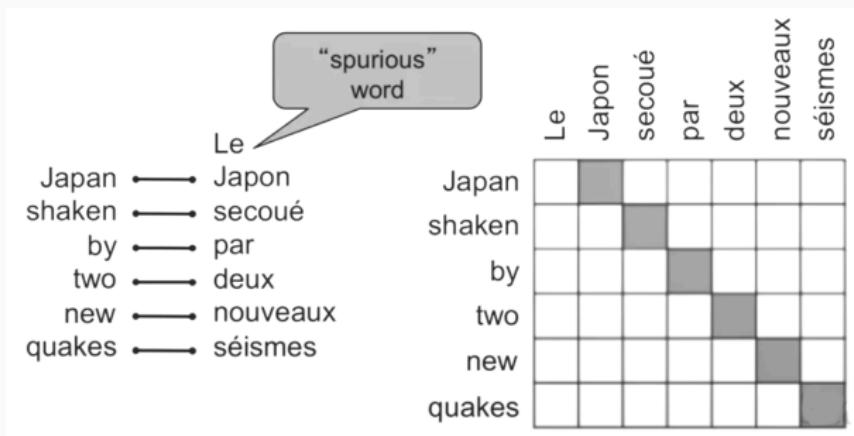
- How to learn translation model  $P(x | y)$  from the parallel corpus?
- Break it down further: Introduce latent  $a$  variable into the model  $P(x, a | y)$
- where  $a$  is the *alignment* (i.e., word-level correspondence between source sentence  $x$  and target sentence  $y$ )



## More notes: Alignment

Alignment is the correspondence between particular words in the translated sentence pair.

- Typological differences between languages lead to complicated alignments
- Some words might have no counterpart (or too many); not one-to-one correspondence



## More notes: Learning alignment

We learn  $P(x, a | y)$  where:

- $y$ : source sentence (e.g., English)
  - $x$ : target sentence (e.g., French)
  - $a$ : word alignment (latent mapping between words in  $y$  and  $x$ )
- 
- Alignment  $a$  is latent: it is not explicitly given in the training data.

## More notes: Learning alignment

We learn  $P(x, a | y)$  where:

- $y$ : source sentence (e.g., English)
  - $x$ : target sentence (e.g., French)
  - $a$ : word alignment (latent mapping between words in  $y$  and  $x$ )
- 
- Alignment  $a$  is latent: it is not explicitly given in the training data.
  - Requires special learning algorithms to estimate parameters with latent variables.

## More notes: Learning alignment

We learn  $P(x, a | y)$  where:

- $y$ : source sentence (e.g., English)
  - $x$ : target sentence (e.g., French)
  - $a$ : word alignment (latent mapping between words in  $y$  and  $x$ )
- 
- Alignment  $a$  is latent: it is not explicitly given in the training data.
  - Requires special learning algorithms to estimate parameters with latent variables.
  - e.g., Expectation-Maximization algorithm

## More notes: Learning alignment

We learn  $P(x, a | y)$  where:

- $y$ : source sentence (e.g., English)
  - $x$ : target sentence (e.g., French)
  - $a$ : word alignment (latent mapping between words in  $y$  and  $x$ )
- 
- Alignment  $a$  is latent: it is not explicitly given in the training data.
  - Requires special learning algorithms to estimate parameters with latent variables.
  - e.g., Expectation-Maximization algorithm
    - **E-step:** given current parameters, estimate how likely each possible alignment is (soft alignment).

## More notes: Learning alignment

We learn  $P(x, a | y)$  where:

- $y$ : source sentence (e.g., English)
  - $x$ : target sentence (e.g., French)
  - $a$ : word alignment (latent mapping between words in  $y$  and  $x$ )
- 
- Alignment  $a$  is latent: it is not explicitly given in the training data.
  - Requires special learning algorithms to estimate parameters with latent variables.
  - e.g., Expectation-Maximization algorithm
    - **E-step:** given current parameters, estimate how likely each possible alignment is (soft alignment).
    - **M-step:** re-estimate translation probabilities  $t(x | y)$  using those expectations.

## 1990s–2010s: SMT

---

- SMT was a huge research field

## 1990s–2010s: SMT

---

- SMT was a huge research field
- The best systems were extremely complex

## 1990s–2010s: SMT

---

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details

## 1990s–2010s: SMT

---

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components

## 1990s–2010s: SMT

---

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components
  - Need to design features to capture particular language phenomena

## 1990s–2010s: SMT

---

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components
  - Need to design features to capture particular language phenomena
- Required compiling and maintaining extra resources

## 1990s–2010s: SMT

---

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components
  - Need to design features to capture particular language phenomena
- Required compiling and maintaining extra resources
  - Like tables of equivalent phrases

## 1990s–2010s: SMT

---

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components
  - Need to design features to capture particular language phenomena
- Required compiling and maintaining extra resources
  - Like tables of equivalent phrases
- Lots of human effort to maintain

## 1990s–2010s: SMT

---

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components
  - Need to design features to capture particular language phenomena
- Required compiling and maintaining extra resources
  - Like tables of equivalent phrases
- Lots of human effort to maintain
  - Repeated effort for each language pair!

## 1990s–2010s: SMT

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components
  - Need to design features to capture particular language phenomena
- Required compiling and maintaining extra resources
  - Like tables of equivalent phrases
- Lots of human effort to maintain
  - Repeated effort for each language pair!
- Era of statistical model then →?

## 1990s–2010s: SMT

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components
  - Need to design features to capture particular language phenomena
- Required compiling and maintaining extra resources
  - Like tables of equivalent phrases
- Lots of human effort to maintain
  - Repeated effort for each language pair!
- Era of statistical model then →?
  - word embedding

## 1990s–2010s: SMT

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components
  - Need to design features to capture particular language phenomena
- Required compiling and maintaining extra resources
  - Like tables of equivalent phrases
- Lots of human effort to maintain
  - Repeated effort for each language pair!
- Era of statistical model then →?
  - word embedding
  - (text classification)

## 1990s–2010s: SMT

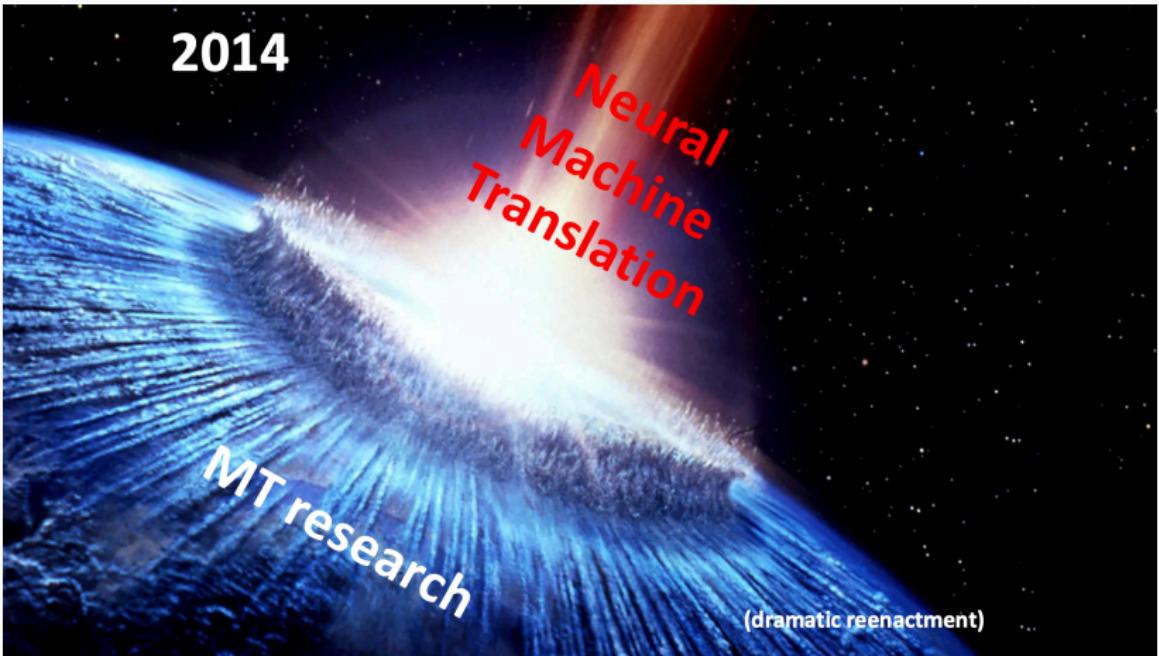
- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components
  - Need to design features to capture particular language phenomena
- Required compiling and maintaining extra resources
  - Like tables of equivalent phrases
- Lots of human effort to maintain
  - Repeated effort for each language pair!
- Era of statistical model then →?
  - word embedding
  - (text classification)
  - dependency parsing

## 1990s–2010s: SMT

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details
- Systems had many separately-designed sub-components
  - Need to design features to capture particular language phenomena
- Required compiling and maintaining extra resources
  - Like tables of equivalent phrases
- Lots of human effort to maintain
  - Repeated effort for each language pair!
- Era of statistical model then →?
  - word embedding
  - (text classification)
  - dependency parsing
  - language modeling

## Neural machine translation

---



## What is neural machine translation?

- Neural machine translation (NMT) is a way to do machine translation with a single end-to-end neural network.

## What is neural machine translation?

---

- Neural machine translation (NMT) is a way to do machine translation with a single end-to-end neural network.
- The neural network architecture is called a sequence-to-sequence (**seq2seq**) and it involves two RNNs (more generally, *neural networks*).

# Seq2Seq Model

- Idea: Mapping one sequence to another.

# Seq2Seq Model

- Idea: Mapping one sequence to another.
- **Encoder:** Reads the input sequence and converts it into a vector representation.

## Seq2Seq Model

- Idea: Mapping one sequence to another.
- **Encoder:** Reads the input sequence and converts it into a vector representation.
- **Decoder:** Generates the output sequence step by step, conditioned on the encoded input.

## Seq2Seq Model

- Idea: Mapping one sequence to another.
- **Encoder:** Reads the input sequence and converts it into a vector representation.
- **Decoder:** Generates the output sequence step by step, conditioned on the encoded input.
- Can be implemented with different architectures:

# Seq2Seq Model

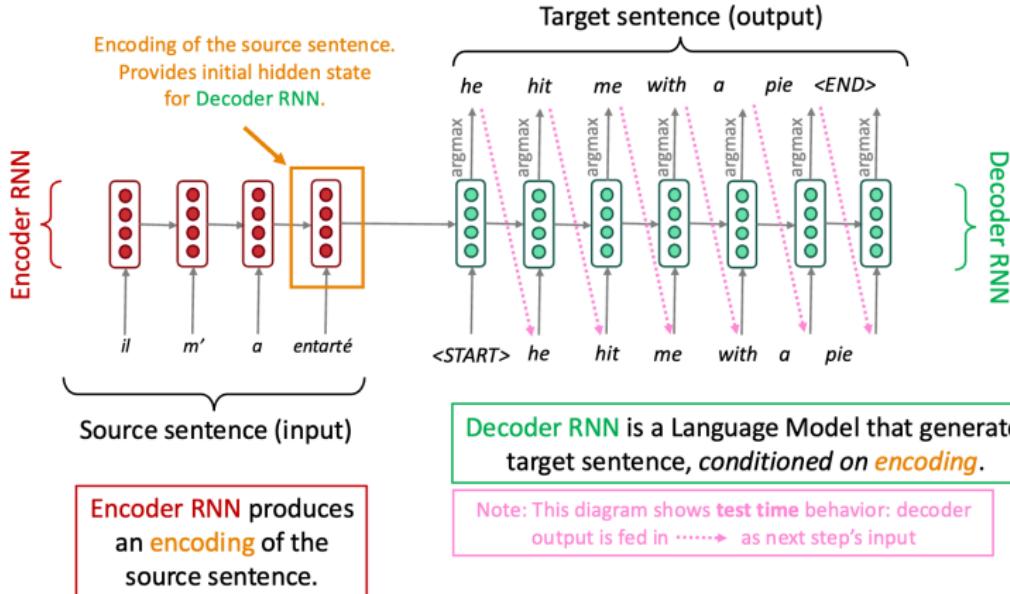
- Idea: Mapping one sequence to another.
- **Encoder:** Reads the input sequence and converts it into a vector representation.
- **Decoder:** Generates the output sequence step by step, conditioned on the encoded input.
- Can be implemented with different architectures:
  - Early models: RNN/LSTM-based encoder-decoder

# Seq2Seq Model

- Idea: Mapping one sequence to another.
- **Encoder:** Reads the input sequence and converts it into a vector representation.
- **Decoder:** Generates the output sequence step by step, conditioned on the encoded input.
- Can be implemented with different architectures:
  - Early models: RNN/LSTM-based encoder-decoder
  - Modern models: Transformer encoder-decoder

# Seq2Seq Model

## The sequence-to-sequence model



## Seq2Seq: Encoder

---

- The encoder reads the input sequence step by step and accumulates information in its hidden states.

## Seq2Seq: Encoder

---

- The encoder reads the input sequence step by step and accumulates information in its hidden states.
- The final hidden state acts as a **context vector** that summarizes the entire input sequence.

## Seq2Seq: Encoder

- The encoder reads the input sequence step by step and accumulates information in its hidden states.
- The final hidden state acts as a **context vector** that summarizes the entire input sequence.
- e.g., The encoder processes the French sentence and compresses it into a single vector representation.

## Seq2Seq: Decoder

---

- The decoder takes the context vector as its initial state and generates the output sequence one token at a time.

## Seq2Seq: Decoder

- The decoder takes the context vector as its initial state and generates the output sequence one token at a time.
- At each step, it uses the previous hidden state, the previously generated token, and the context vector to compute the next hidden state.

## Seq2Seq: Training

- During training, the decoder does not rely on its own previous predictions. Instead, it receives the **ground truth word** from the training data as input.

## Seq2Seq: Training

- During training, the decoder does not rely on its own previous predictions. Instead, it receives the **ground truth word** from the training data as input.
- This strategy is called **teacher forcing**.

## Seq2Seq: Training

- During training, the decoder does not rely on its own previous predictions. Instead, it receives the **ground truth word** from the training data as input.
- This strategy is called **teacher forcing**.
- Advantage: learning becomes stable and converges faster, since errors do not propagate.

## Seq2Seq: Training

- During training, the decoder does not rely on its own previous predictions. Instead, it receives the **ground truth word** from the training data as input.
- This strategy is called **teacher forcing**.
- Advantage: learning becomes stable and converges faster, since errors do not propagate.
- Example: For the target sentence “I love cats”,

## Seq2Seq: Training

- During training, the decoder does not rely on its own previous predictions. Instead, it receives the **ground truth word** from the training data as input.
- This strategy is called **teacher forcing**.
- Advantage: learning becomes stable and converges faster, since errors do not propagate.
- Example: For the target sentence “I love cats”,
  - Step 1: input `<start>` → train model to output “I”

## Seq2Seq: Training

- During training, the decoder does not rely on its own previous predictions. Instead, it receives the **ground truth word** from the training data as input.
- This strategy is called **teacher forcing**.
- Advantage: learning becomes stable and converges faster, since errors do not propagate.
- Example: For the target sentence “I love cats”,
  - Step 1: input `<start>` → train model to output “I”
  - Step 2: feed the true word “I” → train model to output “love”

## Seq2Seq: Training

- During training, the decoder does not rely on its own previous predictions. Instead, it receives the **ground truth word** from the training data as input.
- This strategy is called **teacher forcing**.
- Advantage: learning becomes stable and converges faster, since errors do not propagate.
- Example: For the target sentence “I love cats”,
  - Step 1: input `<start>` → train model to output “I”
  - Step 2: feed the true word “I” → train model to output “love”
  - Step 3: feed the true word “love” → train model to output “cats”

## (Decoding at Test Time)

- At inference time, the true target words are not available.
- The decoder must use its own **predicted word** from the previous step as the next input.
- The process starts with a special `<start>` token and continues until an `<end>` token is generated.
- Example:
  - Step 1: input `<start>` → model predicts “I”
  - Step 2: feed predicted “I” → model predicts “love”
  - Step 3: feed predicted “love” → model predicts “cats”

# Seq2seq

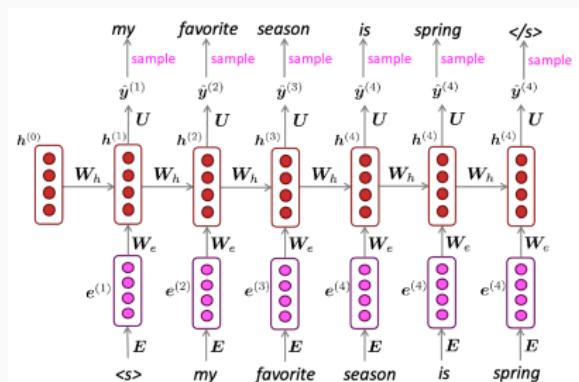
---

- Q: How do we train a seq2seq/NMT system?

- Q: How do we train a seq2seq/NMT system?
- A: Use a large parallel corpus and optimize parameters to maximize the likelihood of the correct target sequence given the source.

# Seq2Seq: Multi-layer RNNs

- RNNs are already *deep* in time: At each timestep, an RNN passes information from the previous hidden state to the next, effectively stacking computations across many steps.

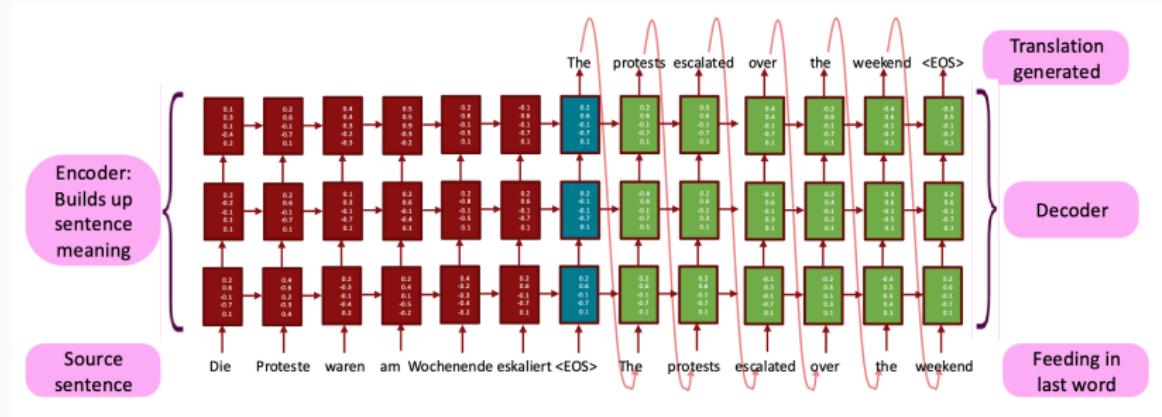


## Seq2Seq: Multi-layer RNNs

---

- We can also add depth in layers: Instead of using just one RNN layer, we can stack multiple RNNs on top of each other, where the output of one layer becomes the input of the next  
**(multi-layer RNNs, stacked RNNs)**

# Seq2Seq: Multi-layer RNNS



## Seq2Seq: Multi-layer RNNs (in practice)

- High-performing RNNs are usually multi-layer (but aren't as deep as convolutional or feed-forward networks)
- e.g., Britz et al. (2017) found that NMT, 2 to 4 layers, is the best for the encoder RNN, and 4 layers is best for the decoder RNN
  - Often 2 layers is a lot better than 1 layer.
  - 3 might be a little better than 2 layers.
- Transformer-based networks (e.g., BERT) are usually deeper, like 12 or 24 layers.

# Advantages of NMT

---

Compared to SMT, NMT has many advantages:

- Better performance

# Advantages of NMT

---

Compared to SMT, NMT has many advantages:

- Better performance
- More fluent

# Advantages of NMT

---

Compared to SMT, NMT has many advantages:

- Better performance
- More fluent
- Better use of context

# Advantages of NMT

---

Compared to SMT, NMT has many advantages:

- Better performance
- More fluent
- Better use of context
- Better use of phrase similarities

# Advantages of NMT

Compared to SMT, NMT has many advantages:

- Better performance
- More fluent
- Better use of context
- Better use of phrase similarities
- A single neural network to be optimized end-to-end

# Advantages of NMT

Compared to SMT, NMT has many advantages:

- Better performance
- More fluent
- Better use of context
- Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No sub-components to be individually optimized

# Advantages of NMT

---

Compared to SMT, NMT has many advantages:

- Better performance
- More fluent
- Better use of context
- Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No sub-components to be individually optimized
- Requires much less human engineering effort

# Advantages of NMT

Compared to SMT, NMT has many advantages:

- Better performance
- More fluent
- Better use of context
- Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No sub-components to be individually optimized
- Requires much less human engineering effort
  - No feature engineering

# Advantages of NMT

Compared to SMT, NMT has many advantages:

- Better performance
- More fluent
- Better use of context
- Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No sub-components to be individually optimized
- Requires much less human engineering effort
  - No feature engineering
  - Same methods for all languages

# Disadvantages of NMT

---

Compared to SMT:

- NMT is less interpretable

# Disadvantages of NMT

---

Compared to SMT:

- NMT is less interpretable
- Hard to debug

# Disadvantages of NMT

---

Compared to SMT:

- NMT is less interpretable
- Hard to debug
- Difficult to control (e.g., can't easily specify rules or guidelines for translation)

# How do we evaluate MT?

---

BLUE (Bilingual Evaluation Understudy)

- BLUE compares the machine-written translation to one or several human-written translations(s), and computes a similarity scores based on:

# How do we evaluate MT?

---

BLUE (Bilingual Evaluation Understudy)

- BLUE compares the machine-written translation to one or several human-written translations(s), and computes a similarity scores based on:
  - n-gram precision (usually for 1, 2, 3, and 4-grams)

# How do we evaluate MT?

---

## BLUE (Bilingual Evaluation Understudy)

- BLUE compares the machine-written translation to one or several human-written translations(s), and computes a similarity scores based on:
  - n-gram precision (usually for 1, 2, 3, and 4-grams)
  - Plus a penalty for too-short system translations

# How do we evaluate MT?

---

## BLUE (Bilingual Evaluation Understudy)

- BLUE compares the machine-written translation to one or several human-written translations(s), and computes a similarity scores based on:
  - n-gram precision (usually for 1, 2, 3, and 4-grams)
  - Plus a penalty for too-short system translations
- BLUE is useful but imperfect

# How do we evaluate MT?

---

## BLUE (Bilingual Evaluation Understudy)

- BLUE compares the machine-written translation to one or several human-written translations(s), and computes a similarity scores based on:
  - n-gram precision (usually for 1, 2, 3, and 4-grams)
  - Plus a penalty for too-short system translations
- BLUE is useful but imperfect
  - There are many valid ways to translate a sentence

# How do we evaluate MT?

---

## BLUE (Bilingual Evaluation Understudy)

- BLUE compares the machine-written translation to one or several human-written translations(s), and computes a similarity scores based on:
  - n-gram precision (usually for 1, 2, 3, and 4-grams)
  - Plus a penalty for too-short system translations
- BLUE is useful but imperfect
  - There are many valid ways to translate a sentence
  - So a good translation can get a poor BLUE score because it has low  $n$ -gram overlap with the human translation

# NMT: the first big success story of NLP deep learning

NMT went from a fringe research attempt in 2014 to the learning standard method in 2016

**2014:** First seq2seq paper published [Sutskever et al. 2014]

**2016:** Google Translate switches from SMT to NMT – and by 2018 everyone has



This is amazing!

- SMT systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by small groups of engineers in a few months

# So, is MT solved?

---

No, many difficulties remain:

- Out-of-vocabulary words
- Domain mismatch between train and test data
- Maintaining context over longer text
- Low-resource language pairs
- Failures to accurately capture sentence meaning
- Pronoun (or zero pronoun) resolution errors
- Morphological agreement errors

## Wrap-up

---

# Wrap-up

---

- New task: Machine translation
- SMT → NMT

## Review: Dependency parser training

---

# Approaches

---

- SpaCy: 11
- PyTorch: 5
- Graph-based parser: 1

## LAS scoreboard (Top 5)

Rank	LAS
1	92.76
2	91.66
3	87.02
4	86.76
5	85.04

Average: 74.5

# Reminder

## 1. Background research brief

Released on Tuesday 09/16/2025

Each group should submit the following to prepare your background-research presentation and to seed your final presentation/paper. Please aim to have a working draft ready for your group check-in on October 9th. After the group meeting, the final version of the draft should be submitted by October 10th (Friday). This is not a graded assignment.

### Things to include

#### 1. Topic / Area

- One sentence stating the focus
- 3-5 keywords

#### 2. Research question / Problem

- 1-2 sentences clearly stating the core question or hypothesis

#### 3. Mini annotated bibliography (3-5 papers) — for each paper include:

- Full citation (consistent style)
- 1-sentence contribution (key finding/idea)
- Method/Data (e.g., corpus, model, experiment)
- Relevance (why it matters for your group project)