CS 6501 Natural Language Processing

Machine Translation, Sequence-to-Sequence Models

Yangfeng Ji

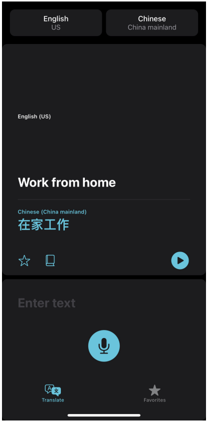
Information and Language Processing Lab
Department of Computer Science
University of Virginia

Overview

1. Noisy Channel Model
2. IBM Models
3. Seq2seq Models
4. Attention Mechanism

Machine Translation Systems

A screenshot of my Apple Translate app



Commerical systems

- Google Translate
- Microsoft Translator
- Apple Translate
- Amazon Translate
- IBM Watson Language Translator
- ...

Problem Formulation

- Goal: translate French to English
- Mathematical formulation

$$P(\boldsymbol{e} \mid \boldsymbol{f}) \tag{1}$$

  where

  - $\boldsymbol{f} = (f_1, \ldots, f_m)$ is a French sentence
  - $\boldsymbol{e} = (e_1, \ldots, e_n)$ is an English translation

Noisy Channel Model

Noisy Channel Model: Definition

Consider a hypothetical communication channel that always adds some some noise to the input clean signal, the task of decoding a noisy channel model is to decode the original clean signal x from the received noisy signal y

- Input signal: x
- Received signal: y
- Task: Decode x from y

Noisy Channel Model: For translation

We can consider machine translation as a decoding task from a noisy channel

- decoding the clean signal from a noisy input, and
- decoding the English text from a foreign language

Source Model $P(\boldsymbol{e})$ Channel Model $P(\boldsymbol{f} \mid \boldsymbol{e})$

Message $\boldsymbol{e}$ $\rightarrow$ $\boxed{\text{Noisy Channel}}$ $\rightarrow$ Message $\boldsymbol{f}$

For example:

- Source language: French
- Target language: English
- Task: Translate French to English

Problem Formulation

- Assume the probabilistic formulation of a noisy channel is $P(\boldsymbol{f} \mid \boldsymbol{e})$, where $\boldsymbol{f} = (f_1, f_2, \ldots, f_m)$ represents a French sentence and $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)$ represents the corresponding English sentence

Source Model $P(\boldsymbol{e})$

Channel Model $P(\boldsymbol{f} \mid \boldsymbol{e})$

$$\text{Message } \boldsymbol{e} \longrightarrow \boxed{\text{Noisy Channel}} \longrightarrow \text{Message } \boldsymbol{f}$$

Problem Formulation

- Assume the probabilistic formulation of a noisy channel is $P(\boldsymbol{f} \mid \boldsymbol{e})$, where $\boldsymbol{f} = (f_1, f_2, \ldots, f_m)$ represents a French sentence and $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)$ represents the corresponding English sentence

Source Model $P(\boldsymbol{e})$ Channel Model $P(\boldsymbol{f} \mid \boldsymbol{e})$

Message $\boldsymbol{e}$ $\longrightarrow$ $\boxed{\text{Noisy Channel}}$ $\longrightarrow$ Message $\boldsymbol{f}$

- The task is to find out $P(\boldsymbol{e} \mid \boldsymbol{f})$, which is the probabilistic formulation of the translation task

Problem Formulation

- Assume the probabilistic formulation of a noisy channel is $P(\boldsymbol{f} \mid \boldsymbol{e})$, where $\boldsymbol{f} = (f_1, f_2, \dots, f_m)$ represents a French sentence and $\boldsymbol{e} = (e_1, e_2, \dots, e_n)$ represents the corresponding English sentence

Source Model $P(\boldsymbol{e})$ Channel Model $P(\boldsymbol{f} \mid \boldsymbol{e})$

$$\text{Message } \boldsymbol{e} \longrightarrow \boxed{\text{Noisy Channel}} \longrightarrow \text{Message } \boldsymbol{f}$$

- The task is to find out $P(\boldsymbol{e} \mid \boldsymbol{f})$, which is the probabilistic formulation of the translation task
- To solve the problem, we need the Bayes theorem

$$P(\boldsymbol{e} \mid \boldsymbol{f}) = \frac{P(\boldsymbol{f} \mid \boldsymbol{e}) \cdot P(\boldsymbol{e})}{P(\boldsymbol{f})} \tag{2}$$

where we need an extra component, the prior distribution of $\boldsymbol{e}$, $P(\boldsymbol{e})$

Problem Formulation

- Assume the probabilistic formulation of a noisy channel is $P(\boldsymbol{f} \mid \boldsymbol{e})$, where $\boldsymbol{f} = (f_1, f_2, \ldots, f_m)$ represents a French sentence and $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)$ represents the corresponding English sentence

Source Model $P(\boldsymbol{e})$ Channel Model $P(\boldsymbol{f} \mid \boldsymbol{e})$

Message $\boldsymbol{e}$ $\longrightarrow$ $\boxed{\text{Noisy Channel}}$ $\longrightarrow$ Message $\boldsymbol{f}$

- The task is to find out $P(\boldsymbol{e} \mid \boldsymbol{f})$, which is the probabilistic formulation of the translation task
- To solve the problem, we need the Bayes theorem

$$P(\boldsymbol{e} \mid \boldsymbol{f}) = \frac{P(\boldsymbol{f} \mid \boldsymbol{e}) \cdot P(\boldsymbol{e})}{P(\boldsymbol{f})} = \frac{P(\boldsymbol{f}, \boldsymbol{e})}{\sum_{\boldsymbol{e}'} P(\boldsymbol{f}, \boldsymbol{e}')} \qquad (2)$$

where we need an extra component, the prior distribution of $\boldsymbol{e}$, $P(\boldsymbol{e})$

Two Components

For machine translation,

$$P(\boldsymbol{e} \mid \boldsymbol{f}) = \frac{P(\boldsymbol{e})P(\boldsymbol{f} \mid \boldsymbol{e})}{P(\boldsymbol{f})} \tag{3}$$

Translating from $\boldsymbol{f}$ to $\boldsymbol{e}$ is essentially a decoding (prediction) problem, therefore we can ignore the

$$\hat{\boldsymbol{e}} = \operatorname*{argmax}_{\boldsymbol{e}} P(\boldsymbol{e} \mid \boldsymbol{f}) = \operatorname*{argmax}_{\boldsymbol{e}} P(\boldsymbol{e})P(\boldsymbol{f} \mid \boldsymbol{e}) \tag{4}$$

Two Components

For machine translation,

$$P(\boldsymbol{e} \mid \boldsymbol{f}) = \frac{P(\boldsymbol{e})P(\boldsymbol{f} \mid \boldsymbol{e})}{P(\boldsymbol{f})} \tag{3}$$

Translating from $\boldsymbol{f}$ to $\boldsymbol{e}$ is essentially a decoding (prediction) problem, therefore we can ignore the

$$\hat{\boldsymbol{e}} = \operatorname*{argmax}_{\boldsymbol{e}} P(\boldsymbol{e} \mid \boldsymbol{f}) = \operatorname*{argmax}_{\boldsymbol{e}} P(\boldsymbol{e})P(\boldsymbol{f} \mid \boldsymbol{e}) \tag{4}$$

- $P(\boldsymbol{e})$: the language model
  - n-gram language models
  - recurrent neural network langauge models

Two Components

For machine translation,

$$P(\boldsymbol{e} \mid \boldsymbol{f}) = \frac{P(\boldsymbol{e})P(\boldsymbol{f} \mid \boldsymbol{e})}{P(\boldsymbol{f})} \tag{3}$$

Translating from $\boldsymbol{f}$ to $\boldsymbol{e}$ is essentially a decoding (prediction) problem, therefore we can ignore the

$$\hat{\boldsymbol{e}} = \operatorname*{argmax}_{\boldsymbol{e}} P(\boldsymbol{e} \mid \boldsymbol{f}) = \operatorname*{argmax}_{\boldsymbol{e}} P(\boldsymbol{e})P(\boldsymbol{f} \mid \boldsymbol{e}) \tag{4}$$

- $P(\boldsymbol{e})$: the language model
  - n-gram language models
  - recurrent neural network langauge models
- $P(\boldsymbol{f} \mid \boldsymbol{e})$: the translation model
  - A probabilistic mapping from English to Franch

Two Components (Cont.)

Divide one big problem into two subproblems: $P(\boldsymbol{e})$ and $P(\boldsymbol{f} \mid \boldsymbol{e})$, then solve them separately (with extra resources)

Source Model $P(\boldsymbol{e})$

Channel Model $P(\boldsymbol{f} \mid \boldsymbol{e})$

Message $\boldsymbol{e}$ → $\boxed{\text{Noisy Channel}}$ → Message $\boldsymbol{f}$

Two Components (Cont.)

Divide one big problem into two subproblems: $P(\boldsymbol{e})$ and $P(\boldsymbol{f} \mid \boldsymbol{e})$, then solve them separately (with extra resources)

- For example, $P(\boldsymbol{e})$ is essentially a language model, which can be trained with as many training example as we have (no annotation needed)

Two Components (Cont.)

Divide one big problem into two subproblems: $P(\boldsymbol{e})$ and $P(\boldsymbol{f} \mid \boldsymbol{e})$, then solve them separately (with extra resources)

Source Model $P(\boldsymbol{e})$ Channel Model $P(\boldsymbol{f} \mid \boldsymbol{e})$

Message $\boldsymbol{e}$ $\longrightarrow$ $\boxed{\text{Noisy Channel}}$ $\longrightarrow$ Message $\boldsymbol{f}$

- For example, $P(\boldsymbol{e})$ is essentially a language model, which can be trained with as many training example as we have (no annotation needed)
- A key problem in statistical machine translation is to model $P(\boldsymbol{f} \mid \boldsymbol{e})$. In this lecture we will discuss two methods
    - IBM Model 1
    - IBM Model 2
    - IBM Model $\cdots$

IBM Models

Challenge of Probability Definition

For modeling the probability distribution of a long sequence: a similar argument has also been used in the language modeling task.

Given

- an English sentence $\boldsymbol{e}$ with n words $(e_1, \ldots, e_n)$ and
- a French sentence $\boldsymbol{f}$ with m words $(f_1, \ldots, f_m)$,

directly modeling

$$P(\boldsymbol{f} \mid \boldsymbol{e}) = P(f_1, \ldots, f_m \mid e_1, \ldots, e_n) \tag{5}$$

is challenging.

Example

Consider the specific example

$$e = \text{And the program has been implemented}$$
$$f = \text{Le programme a ete mis en application}$$

- To directly model the conditional probability of $\boldsymbol{f}$ given $\boldsymbol{e}$, $P(\boldsymbol{f} \mid \boldsymbol{e})$ defines a probability on a 13-dimensional space[1]

[1] 13 is the total number of English and Franch words in the source and target sentences.

Example

Consider the specific example

$$e = \text{And the program has been implemented}$$
$$f = \text{Le programme a ete mis en application}$$

- To directly model the conditional probability of $\boldsymbol{f}$ given $\boldsymbol{e}$, $P(\boldsymbol{f} \mid \boldsymbol{e})$ defines a probability on a 13-dimensional space[1]
- Each f_j depends on only part of $\boldsymbol{e}$. In other words, (1) there are alignments between French and English words, and (2) word dependency only exists between source words and target words with alignments.

[1] 13 is the total number of English and Franch words in the source and target sentences.

Example (Cont.)

For the previous example, here are some example alignments between words in English and French

	1	2	3	4	5	6	7
e	And	the	program	has	been	implemented	
f	Le	programme	a	ete	mis	en	application

Example (Cont.)

For the previous example, here are some example alignments between words in English and French

	1	2	3	4	5	6	7
e	And	the	program	has	been	implemented	
f	Le	programme	a	ete	mis	en	application

Example (Cont.)

For the previous example, here are some example alignments between words in English and French

	1	2	3	4	5	6	7
$\boldsymbol{e}$	And	the	program	has	been	implemented	
$\boldsymbol{f}$	Le	programme	a	ete	mis	en	application

- If we use $a_j = i$ to represent that the j-th word in $\boldsymbol{f}$ is aligned with the i-th word in $\boldsymbol{e}$. For the abovementioned example, we have $a_1 = 2, \quad a_6 = 6$

Example (Cont.)

For the previous example, here are some example alignments between words in English and French

	1	2	3	4	5	6	7
$\boldsymbol{e}$	And	the	program	has	been	implemented	
$\boldsymbol{f}$	Le	programme	a	ete	mis	en	application

- If we use $a_j = i$ to represent that the j-th word in $\boldsymbol{f}$ is aligned with the i-th word in $\boldsymbol{e}$. For the abovementioned example, we have $a_1 = 2, \quad a_6 = 6$
- With the *explicit* alignments, we can simplify the conditional probability. For example,

$$P(f_1 \mid \boldsymbol{e}, a_1) = P(f_1 \mid e_2, a_1) \tag{6}$$

Alignments

Similarly, we introduce new alignment variables $\boldsymbol{a} = (a_1, \ldots, a_m)$

$$P(f_1, \ldots, f_m, a_1, \cdots, a_m \mid e_1, \ldots, e_n) \tag{7}$$

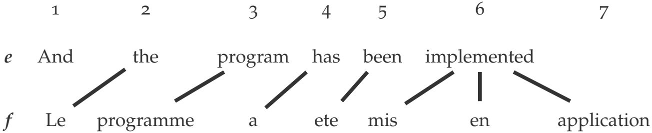
where $a_j \in \{0, 1, \ldots, n\}$

Alignments

Similarly, we introduce new alignment variables $\boldsymbol{a} = (a_1, \ldots, a_m)$

$$P(f_1, \ldots, f_m, a_1, \cdots, a_m \mid e_1, \ldots, e_n) \tag{7}$$

where $a_j \in \{0, 1, \ldots, n\}$



$$a_1 = 2 \qquad a_2 = 3$$
$$a_3 = 4 \qquad a_4 = 5$$
$$a_5 = 6 \qquad a_6 = 6$$
$$a_7 = 6$$

Further Factorization

Further break down $P(\boldsymbol{f}, \boldsymbol{a} \mid \boldsymbol{e})$ into two parts:

$$P(\boldsymbol{f}, \boldsymbol{a} \mid \boldsymbol{e}) = P(\boldsymbol{a} \mid \boldsymbol{e}) P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e}) \tag{8}$$

- Alignment $P(\boldsymbol{a} \mid \boldsymbol{e})$
- Translation with a given alignment $\boldsymbol{a}$, $P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e})$
- Both $P(\boldsymbol{a} \mid \boldsymbol{e})$ and $P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e})$ can be further factorized

IBM Model 1: $P(\boldsymbol{a} \mid \boldsymbol{e})$

In IBM Model 1, all alignments are equally likely

$$P(\boldsymbol{a} \mid \boldsymbol{e}) = \prod_{j=1}^{m} q(a_j = i \mid j, n, m) = \frac{1}{(n+1)^m} \tag{9}$$

where m and n are the lengths of $\boldsymbol{f}$ and $\boldsymbol{e}$ respectively

IBM Model 1: $P(\boldsymbol{a} \mid \boldsymbol{e})$

In IBM Model 1, all alignments are equally likely

$$P(\boldsymbol{a} \mid \boldsymbol{e}) = \prod_{j=1}^{m} q(a_j = i \mid j, n, m) = \frac{1}{(n+1)^m} \tag{9}$$

where m and n are the lengths of $\boldsymbol{f}$ and $\boldsymbol{e}$ respectively

- Uniform distribution: major simplification, great starting point

$$q(a_j = i \mid j, n, m) = \frac{1}{(n+1)} \tag{10}$$

IBM Model 1: $P(\boldsymbol{a} \mid \boldsymbol{e})$

In IBM Model 1, all alignments are equally likely

$$P(\boldsymbol{a} \mid \boldsymbol{e}) = \prod_{j=1}^{m} q(a_j = i \mid j, n, m) = \frac{1}{(n+1)^m} \tag{9}$$

where m and n are the lengths of $\boldsymbol{f}$ and $\boldsymbol{e}$ respectively

- Uniform distribution: major simplification, great starting point

$$q(a_j = i \mid j, n, m) = \frac{1}{(n+1)} \tag{10}$$

- Why $n+1$? A: some words $\boldsymbol{f}$ cannot find an alignment with any word $\boldsymbol{e}$, so we need a dummy token.

IBM Model 1: $P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e})$

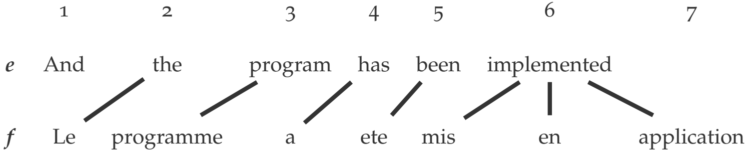
The translation probability with the alignment $\boldsymbol{a}$ on condition

$$P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e}) = \prod_{j=1}^{m} t(f_j \mid e_{a_j}) \tag{11}$$

- f_j: the j-th French word
- $a_j = i$: the alignment of the j-th French word
- e_{a_j}: the aligned English word of the j-th French word
- $t(f_j \mid e_{a_j})$: the translation probability from the $a_j(= i)$-th English word to the j-th French word

Example (Cont.)

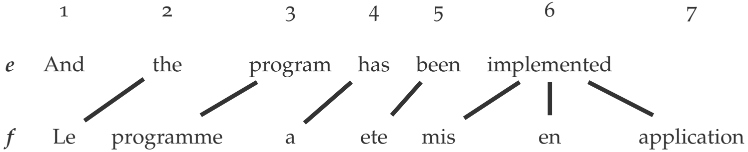
Given the previous example



we have the translation probability with alignment $P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e})$

Example (Cont.)

Given the previous example



we have the translation probability with alignment $P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e})$

$$
\begin{aligned}
P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e}) = & t(\text{Le} \mid \text{the}) \cdot t(\text{programme} \mid \text{program}) \cdot \\
& t(\text{a} \mid \text{has}) \cdot t(\text{ete} \mid \text{been}) \cdot \\
& t(\text{mis} \mid \text{implemented}) \cdot t(\text{en} \mid \text{implemented}) \cdot \\
& t(\text{application} \mid \text{implemented})
\end{aligned}
$$

IBM Model 1: Final result

The final probability $P(\boldsymbol{f} \mid \boldsymbol{e})$ after we have each pieces of information

$$\begin{aligned} P(\boldsymbol{f} \mid \boldsymbol{e}) &= \sum_{\boldsymbol{a}} P(\boldsymbol{f}, \boldsymbol{a} \mid \boldsymbol{e}) \\ &= \sum_{\boldsymbol{a}} P(\boldsymbol{a} \mid \boldsymbol{e}) P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e}) \\ &= \sum_{\boldsymbol{a}} \frac{1}{(n+1)^m} \prod_{j=1}^{m} t(f_j \mid e_{a_j}) \end{aligned} \tag{12}$$

Basic idea: Break a big conditional probability into small pieces on word pairs

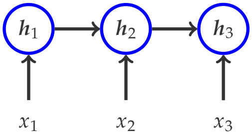
Seq2seq Models

Recurrent Neural Networks (RNNs)

A simple RNN is defined by the following recursive function

$$\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \tag{13}$$

and depicted as



where

- $\boldsymbol{h}_{t-1}$: hidden state at time step $t-1$
- $\boldsymbol{x}_t$: input at time step t
- $\boldsymbol{h}_t$: hidden state at time step t

Probabilistic Modeling in Neural MT

For a machine translation problem

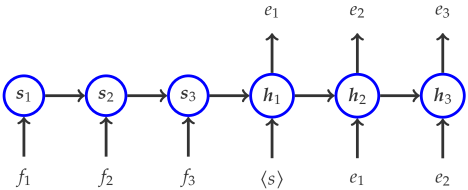
- Input: $\boldsymbol{f} = (f_1, f_2, \ldots, f_m)$
- Output: $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)$

Neural machine translation usually model the conditional probability $P(\boldsymbol{e} \mid \boldsymbol{f})$ directly as

$$
\begin{aligned}
P(\boldsymbol{e} \mid \boldsymbol{f}) &= P(e_1 \mid \boldsymbol{f}) \cdot P(e_2 \mid e_1, \boldsymbol{f}) && (14)\\
&\quad \cdot P(e_3 \mid \boldsymbol{e}_{1:2}, \boldsymbol{f}) \cdots && (15)\\
&\quad \cdot P(e_n \mid \boldsymbol{e}_{1:n-1}, \boldsymbol{f}) && (16)\\
&= \prod_{i=1}^{m} P(e_i \mid \boldsymbol{e}_{1:i-1}, \boldsymbol{f}) && (17)
\end{aligned}
$$

Neural Sequence-to-sequence Models

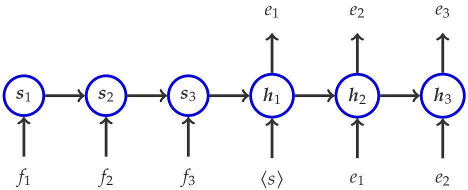
Ideally, a sequence-to-sequence model offers a natural framework of mapping a sequence to another sequence *step-by-step*



[Sutskever et al., 2014]

Neural Sequence-to-sequence Models

Ideally, a sequence-to-sequence model offers a natural framework of mapping a sequence to another sequence *step-by-step*

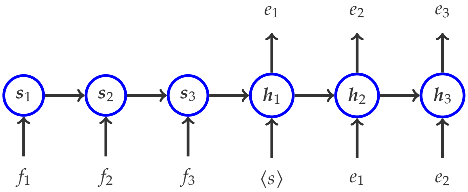


- The first RNN encoding the input sequence $\boldsymbol{f}$ is called the **encoder**
- The second RNN decoding the output sequence $\boldsymbol{e}$ step-by-step is called the **decoder**

[Sutskever et al., 2014]

Neural Sequence-to-sequence Models

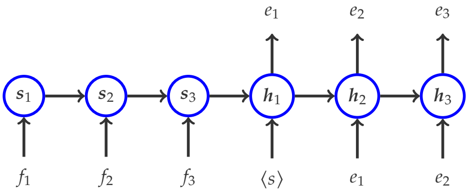
Ideally, a sequence-to-sequence model offers a natural framework of mapping a sequence to another sequence *step-by-step*



- The first RNN encoding the input sequence $\boldsymbol{f}$ is called the **encoder**
- The second RNN decoding the output sequence $\boldsymbol{e}$ step-by-step is called the **decoder**
- In a broader sense, this model is also called the encoder-decoder model

[Sutskever et al., 2014]

Neural Sequence-to-sequence Models (Cont.)



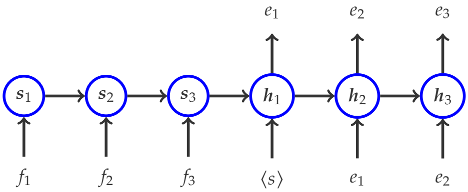
Specifically, for each decoding step, we can write the conditional probability as

$$P(e_i \mid \boldsymbol{e}_{1:i-1}, \boldsymbol{f}) = \mathrm{softmax}(\boldsymbol{W}_o \boldsymbol{h}_i) \tag{18}$$

where $\boldsymbol{h}_i$ is the hidden state on step i

Tricks (I)

A basic seq2seq model is nothing more than connecting two RNNs (LSTMs) together



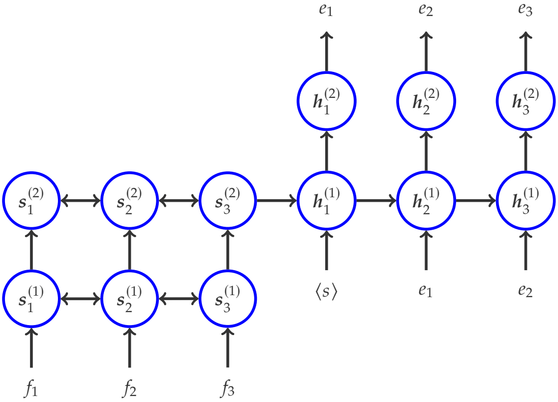
There are some additional tricks that to make it work

- two different LSTMs: one for input sequence and the other for output sequence
- it can greatly improve the performance on the output side, by reversing the order of input sequence

[Sutskever et al., 2014]

Tricks (II)

- stacked (*or* deep) LSTM with multiple layers on both encoder and decoder, which has much more potential than single-layer LSTMs
- bi-directional LSTM for the encoder



Seq2seq Models for Machine Translation

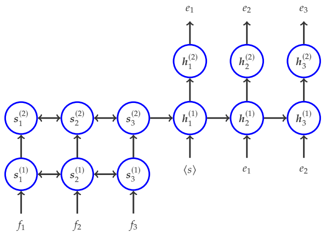
Early ways of using seq2seq models for machine translation

- Use seq2seq model evaluation scores to re-rank the k-best list [Sutskever et al., 2014]

Seq2seq Models for Machine Translation

Early ways of using seq2seq models for machine translation

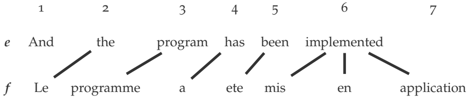
- Use seq2seq model evaluation scores to re-rank the k-best list [Sutskever et al., 2014]
- Use seq2seq models evaluation scores as additional features in the translation model [Cho et al., 2014]



Major limitation: the input signal is probably too weak on the decoder side

Attention Mechanism

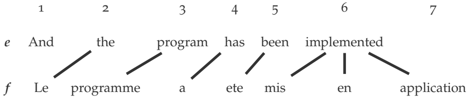
Another important module in statistical machine translation is the alignments offered by $P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e})$, which essentially is a word-level mapping between the input and output



$$
\begin{aligned}
P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e}) = & \, t(\text{Le} \mid \text{the}) \cdot t(\text{programme} \mid \text{program}) \cdot \\
& t(\text{a} \mid \text{has}) \cdot t(\text{ete} \mid \text{been}) \cdot \\
& t(\text{mis} \mid \text{implemented}) \cdot t(\text{en} \mid \text{implemented}) \cdot \\
& t(\text{application} \mid \text{implemented})
\end{aligned}
$$

Alignment in Statistical MT

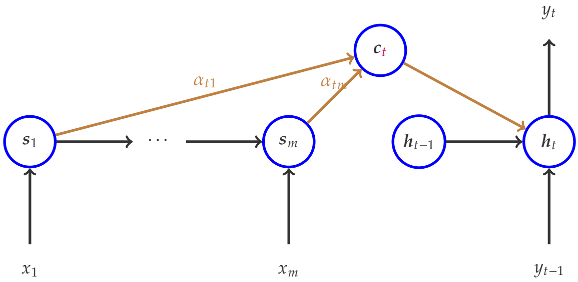
Another important module in statistical machine translation is the alignments offered by $P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e})$, which essentially is a word-level mapping between the input and output



$$
\begin{aligned}
P(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{e}) = & \, t(\text{Le} \mid \text{the}) \cdot t(\text{programme} \mid \text{program}) \cdot \\
& t(\text{a} \mid \text{has}) \cdot t(\text{ete} \mid \text{been}) \cdot \\
& t(\text{mis} \mid \text{implemented}) \cdot t(\text{en} \mid \text{implemented}) \cdot \\
& t(\text{application} \mid \text{implemented})
\end{aligned}
$$

This is also something missed from the seq2seq models we discussed in the previous section.

With the attention mechanism [Bahdanau et al., 2015], the hidden states in the decoder are computed as



$$c_t = \sum_{j=1}^{m} \alpha_{tj} s_j \qquad h_t = f(h_{t-1}, y_{t-1}, c_t) \tag{19}$$

where c_t is dynamically changing over time

Attention Weights

In [Bahdanau et al., 2015], the attention weights are compuated as

$$\alpha_{tj} = \frac{\exp(\phi(\boldsymbol{h}_{t-1}, \boldsymbol{s}_j))}{\sum_{j'=1}^{m} \exp(\phi(\boldsymbol{h}_{t-1}, \boldsymbol{s}_{j'}))}. \tag{20}$$

where $\phi(\boldsymbol{h}_{t-1}, \boldsymbol{h}_j)$ is specifically defined as

$$\phi(\boldsymbol{h}_{t-1}, \boldsymbol{s}_j) = \boldsymbol{v}_a^{\top} \tanh(\mathbf{W}_{ao} \boldsymbol{h}_{t-1} + \mathbf{W}_{ai} \boldsymbol{s}_j) \tag{21}$$

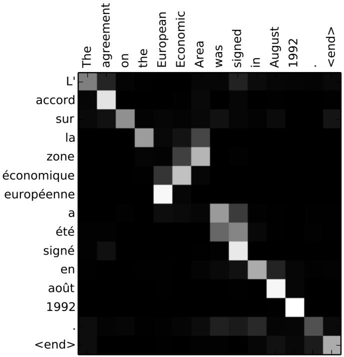
with parameters $\mathbf{W}_{ao}, \mathbf{W}_{ai}$ and $\boldsymbol{v}_a$.

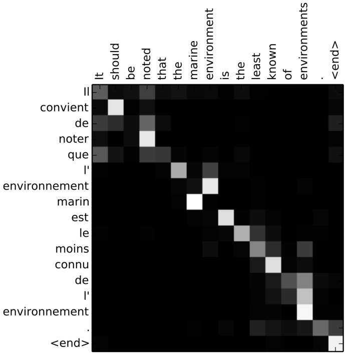
The softmax function in Equation 20 implies

$$\sum_{j=1}^{m} \alpha_{tj} = 1 \tag{22}$$

Example: Attention Weights in Machine Translation

By visualizing the attention weights, we can see the (soft) alignments between the source sentence and the target translation[2]





[2]The examples are from [Bahdanau et al., 2015], in which the translation is from English to French.

Reference

Bahdanau, D., Cho, K., and Bengio, Y. (2015).
Neural machine translation by jointly learning to align and translate.
In *ICLR*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014).
Learning phrase representations using rnn encoder-decoder for statistical machine translation.
arXiv preprint arXiv:1406.1078.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014).
Sequence to sequence learning with neural networks.
In *Advances in neural information processing systems*, pages 3104–3112.