Language Technology
http://cs.lth.se/edan20/
The Encoder-Decoder Architecture

Pierre Nugues

Pierre.Nugues@cs.lth.se
http://cs.lth.se/pierre_nugues/

September 28, 2023

Machine Translation

- Process of translating automatically a text from a source language into a target language
- Started after the 2nd world war to translate documents from Russian to English
- Early working systems from French to English in Canada
- Renewed huge interest with the advent of the web
- Google claims it has more than 500m users daily worldwide, with 103 languages.
- Massive progress permitted by the encoder-decoder networks

Corpora for Machine Translation

Initial ideas in machine translation: use bilingual dictionaries and formalize grammatical rules to transfer them from a source language to a target language.

Statistical machine translation:

1. Use very large bilingual corpora;
2. Align the sentences or phrases, and
3. Given a sentence in the source language, find the matching sentence in the target language.

Pioneered at IBM on French and English with Bayesian statistics.

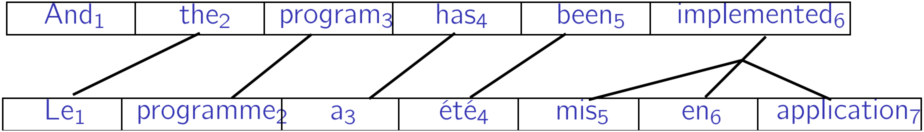
As of today, the encoder-decoder architecture is dominant

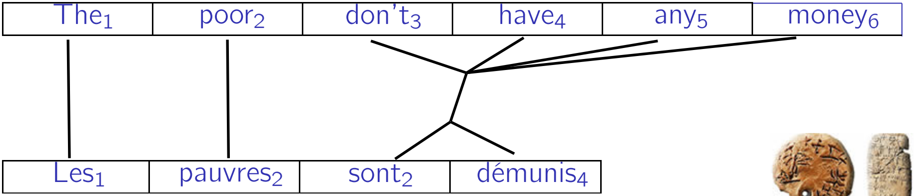
Parallel Corpora (Swiss Federal Law)

German	French	Italian
Art. 35 Milchtransport	**Art. 35 Transport du lait**	**Art. 35 Trasporto del latte**
1 Die Milch ist schonend und hygienisch in den Verarbeitungsbetrieb zu transportieren. Das Transportfahrzeug ist stets sauber zu halten. Zusammen mit der Milch dürfen keine Tiere und milchfremde Gegenstände transportiert werden, welche die Qualität der Milch beeinträchtigen können.	1 Le lait doit être transporté jusqu'à l'entreprise de transformation avec ménagement et conformément aux normes d'hygiène. Le véhicule de transport doit être toujours propre. Il ne doit transporter avec le lait aucun animal ou objet susceptible d'en altérer la qualité.	1 Il latte va trasportato verso l'azienda di trasformazione in modo accurato e igienico. Il veicolo adibito al trasporto va mantenuto pulito. Con il latte non possono essere trasportati animali e oggetti estranei, che potrebbero pregiudicarne la qualità.

Alignment (Brown et al. 1993)

Canadian Hansard





Translations with RNNs

RNN can easily map sequences to sequences, where we have two lists: one for the source and the other for the target

y	Le	serveur	apporta	le	plat
x	The	waiter	brought	the	meal

The **x** and **y** vectors must have the same length.
In our case, *a apporté* is more frequent than *apporta*, but it breaks the alignment, as well as in many other examples

Translation with RNN

To solve the alignment problem, Sutskever al al. (2014) proposed (quoted from their paper, `https://arxiv.org/abs/1409.3215`):

1. The simplest strategy for general sequence learning is to map the input sequence to a fixed-sized vector using one RNN, and then to map the vector to the target sequence with another RNN [...]
2. it would be difficult to train the RNNs due to the resulting long term dependencies [...]. However, the Long Short-Term Memory (LSTM) is known to learn problems with long range temporal dependencies.

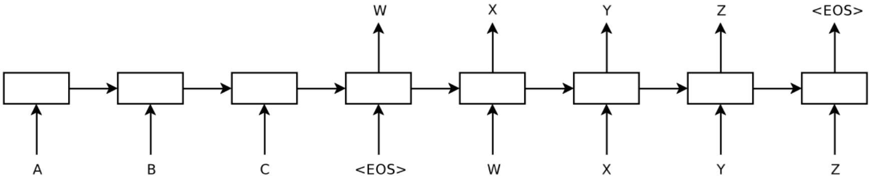
Using the Hidden States

To solve the alignment problem, Sutskever al al. (2014) proposed (quoted from their paper, `https://arxiv.org/abs/1409.3215`):

1. LSTM estimate[s] the conditional probability $p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T)$, where $(x_1, \ldots, x_T)$ is an input sequence and $y_1, \ldots, y_{T'}$ is its corresponding output sequence whose length T' may differ from T.
2. The LSTM computes this conditional probability by:
   1. First obtaining the fixed-dimensional representation v of the input sequence $(x1, \ldots, xT)$ given by the last hidden state of the LSTM, (**encoder**) and then
   2. computing the probability of $y_1, \ldots, y_{T'}$ with a standard LSTM-LM formulation whose initial hidden state is set to the representation v of $x_1, \ldots, x_T$ (**decoder**)

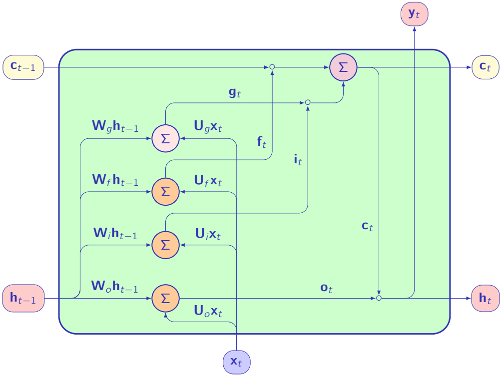
The Encoder-Decoder

1. The source sequence: (A, B, C) results in a hidden-state (not on the picture)
2. The target sequence: (<bos>, X, Y, Z, <eos>) is obtained using an auto-regressive process



After Sutskever al al. (2014)

The LSTM Architecture



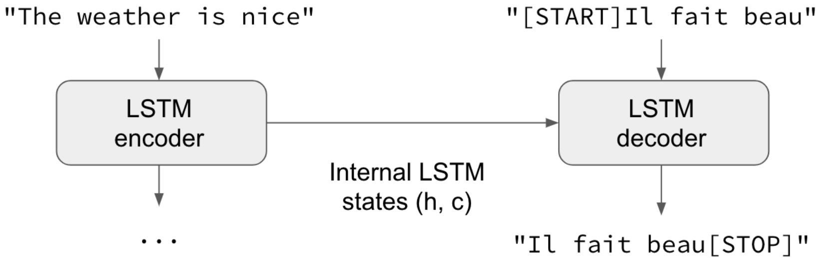
An LSTM unit showing the data flow, where $\mathbf{g}_t$ is the unit input, $\mathbf{i}_t$, the input gate, $\mathbf{f}_t$, the forget gate, and $\mathbf{o}_t$, the output gate. The activation functions have been omitted

Sequence-to-Sequence Translation

We follow and reuse: `https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in html` and `https://keras.io/examples/nlp/lstm_seq2seq/` from Chollet.

1. We start with input sequences from a language (e.g. English sentences) and corresponding target sequences from another language (e.g. French sentences).
2. An encoder LSTM turns input sequences to 2 state vectors (we keep the last LSTM state and discard the outputs).
3. A decoder LSTM is trained to turn the target sequences into the same sequence but offset by one timestep in the future. This training process is called "teacher forcing" in this context.
4. It uses the state vectors from the encoder as initial state. Effectively, the decoder learns to generate `targets[t+1...]` given `targets[...t]`, conditioned on the input sequence.

Sequence-to-Sequence Translation

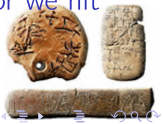


From https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in html

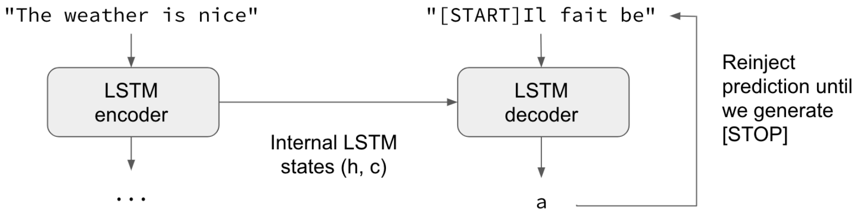
Inference

Following Chollet, in inference mode, to decode unknown input sequences, we:

- Encode the input sequence into state vectors
- Start with a target sequence of size 1 (just the start-of-sequence character)
- Feed the state vectors and 1-char target sequence to the decoder to produce predictions for the next character
- Sample the next character using these predictions (we simply use argmax).
- Append the sampled character to the target sequence
- Repeat until we generate the end-of-sequence character or we hit the character limit.



Sequence-to-Sequence Translation



From https://blog.keras.io/
a-ten-minute-introduction-to-sequence-to-sequence-learning-i
html

Improving the Architecture: Encoder-Decoder

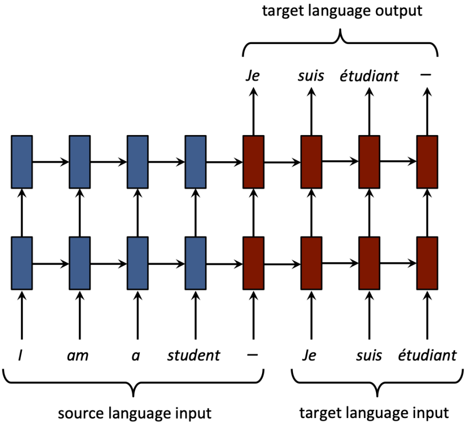
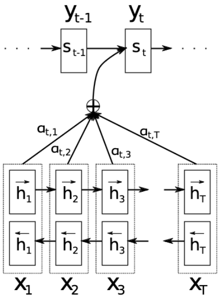


Figure 1: A simplified diagram of NMT.

From: Compression of Neural Machine Translation Models via Pruning by Abigail See, Minh-Thang Luong, and Christopher D. Manning

Attention

1. The target sentence receives only one vector input (hidden state) from the source
2. Its influence decreases as we move forward in the target sentence
3. Long sentences are poorly translated
4. **Attention**: Concatenate a weight product of the source hidden states to the hidden state of the previous target word



From: Neural Machine Translation by Jointly Learning to Align and Translate, zmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio

Improving the Architecture: Adding Attention

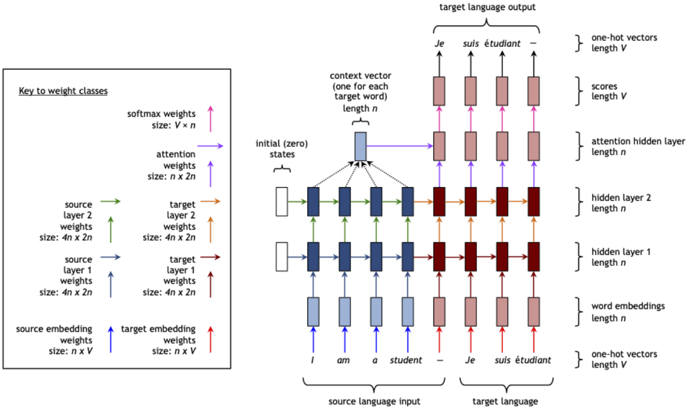


Figure 2: NMT architecture. This example has two layers, but our system has four. The different weight classes are indicated by arrows of different color (the black arrows in the top right represent simply choosing the highest-scoring word, and thus require no parameters). Best viewed in color.

From: Compression of Neural Machine Translation Models via Pruning by Abigail See, Minh-Thang Luong, and Christopher D. Manning

Code example: `https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html`