Language Technology
http://cs.lth.se/edan20/

Chapter 13: Complements to Sequence-to-Sequence Programming

Pierre Nugues

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

October 6, 2022

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.

Neural nets are now dominant

Parallel Corpora (Swiss Federal Law)

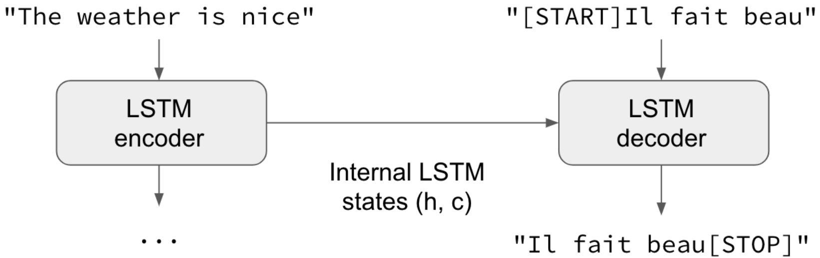
German	French	Italian
Art. 35 Milchtransport	**Art. 35 Transport du lait**	**Art. 35 Trasporto del latte**
1 Die Milch ist schonend und hygienisch in den Verarbeitungsbetrieb zu transportieren. Das Transportfahrzeug ist stets sauber zu halten. Zusammen mit der Milch dürfen keine Tiere und milchfremde 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.

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.

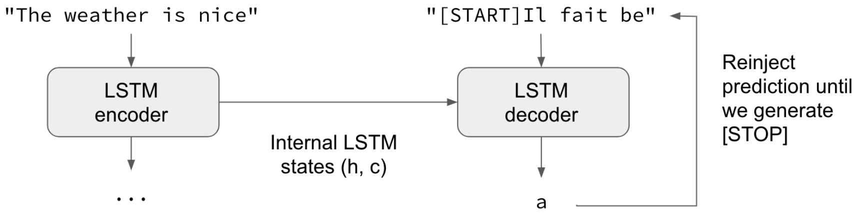
1. We start with input sequences from a language (e.g. English sentences) and corresponding target sequences from another language (e.g. French sentences).
2. An encoder LSTM turns input sequences to 2 state vectors (we keep the last LSTM state and discard the outputs).
3. A decoder LSTM is trained to 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



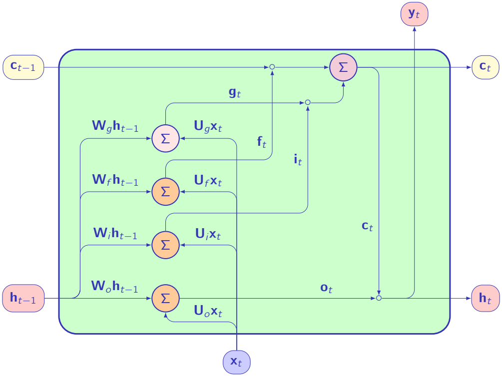
From https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in html

Sequence-to-Sequence Translation



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

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

Improving the Architecture: Encoder-Decoder

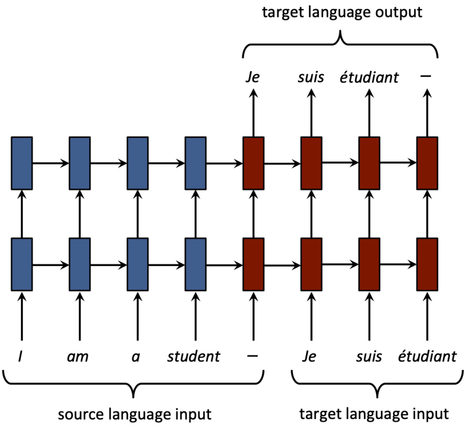
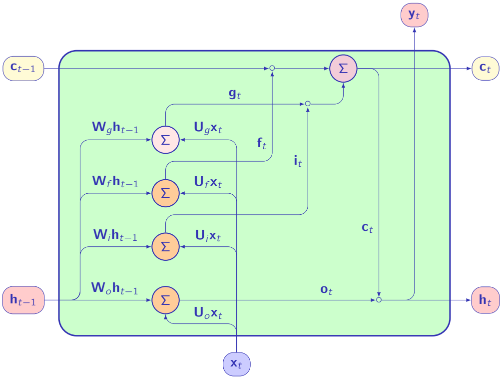


Figure 1: A simplified diagram of NMT.

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

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

The Functional API

The functional API is a second, more flexible API:

- Sequential:

```
seq_model = Sequential()
seq_model.add(layers.Dense(32, activation='relu',
  input_shape=(64,)))
seq_model.add(layers.Dense(32, activation='relu'))
seq_model.add(layers.Dense(10, activation='softmax'))
```

- Functional:

```
input_tensor = Input(shape=(64,))
x = layers.Dense(32, activation='relu')(input_tensor)
x = layers.Dense(32, activation='relu')(x)
output_tensor = layers.Dense(10, activation='softmax')(x)
model = Model(input_tensor, output_tensor)
```

See Chollet, page 175–177

LSTM API

```
encoder_inputs = keras.Input(
     shape=(None, num_encoder_tokens),
     name='encoder_input')
encoder = keras.layers.LSTM(latent_dim,
     return_state=True,
     name='encoder_lstm')
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
```

Then the decoder uses:

```
encoder_states = [state_h, state_c]
```

Code Example

Return values of a LSTM
Jupyter Notebooks: `https://github.com/pnugues/ilppp/blob/master/programs/ch10/python/lstm_output.ipynb`

Further Readings

- For the latest developments, see: http://www.statmt.org/wmt22/
- For a description of systems with attention, see https://www.tensorflow.org/tutorials/text/nmt_with_attention and https://www.tensorflow.org/tutorials/text/transformer
- For an example attention program in Python, see, https://machinelearningmastery.com/encoder-decoder-attention-sequence-to-sequence-prediction
- For another tutorial using pytorch: https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html