Language Technology

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

Chapter 13: Transformers: The Decoder

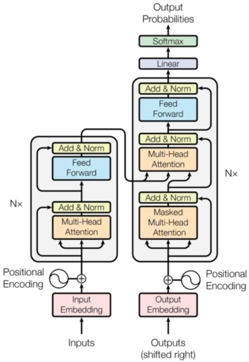
Pierre Nugues

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

September 29, 2022

Transformers: The Decoder

Transformers were originally developed for machine translation
Initial language pairs were: English-French, English-German, and the reverse



The input corresponds to words in the source language (say English) and the output in the target language (French or German)

Language Models

- A language model is a statistical estimate of a word sequence.
- Originally developed for speech recognition
- A causal or autoregressive language model enables to predict the next word given a sequence of previous words:
- Given the sequence $x_1, x_2, \ldots, x_{n-1}$, predict x_n
- Sequences can approximated by:
    - Bigrams, sequences of two words, given the previous word predict the next one
    - Trigrams, sequences of three words, given the two previous words predict the next one
    - N-grams, sequences of N words

Trigrams

Word	Rank	More likely alternatives
We	9	*The This One Two A Three Please In*
need	7	*are will the would also do*
to	1	
resolve	85	*have know do...*
all	9	*the this these problems...*
of	2	*the*
the	1	
important	657	*document question first...*
issues	14	*thing point to...*
within	74	*to of and in that...*
the	1	
next	2	*company*
two	5	*page exhibit meeting day*
days	5	*weeks years pages months*

Language Models and Generation

Using a n-gram language model, we can generate a sequence of words.
Starting from a first word, w_1, we extract the conditional probabilities: $P(w_2|w_1)$.
We could take the highest value, but it would always generate the same sequence.
Instead, we will draw our words from a multinomial distribution using `np.random.multinomial()`.
Given a probability distribution, this function draws a sample that complies the distribution.
Having, $P(want|I) = 0.5$, $P(wish|I) = 0.3$, $P(will|I) = 0.2$, the function will draw wish 30% of the time.

Code Example

Generating sequences with Bayesian probabilities

Jupyter Notebooks: https://github.com/pnugues/edan95/blob/master/programs/5.7-generation.ipynb

Generating Character Sequences with LSTMs

In the previous example, we used words. We can use characters instead.
We also used Bayesian probabilities. We can use LSTMs instead.
This is the idea of Chollet's program, pages 272-278.
X consists of sequences of 60 characters with a step of 3 characters
y is the character following the sequence
Let us use this excerpt:

> *is there not ground for suspecting that all philosophers*

and 10 characters, where ␣ marks a space:

$$\mathbf{X} = \begin{bmatrix} \mathrm{i} & \mathrm{s} & \sqcup & \mathrm{t} & \mathrm{h} & \mathrm{e} & \mathrm{r} & \mathrm{e} & \sqcup & \mathrm{n} \\ \mathrm{t} & \mathrm{h} & \mathrm{e} & \mathrm{r} & \mathrm{e} & \sqcup & \mathrm{n} & \mathrm{o} & \mathrm{t} & \sqcup \\ \mathrm{r} & \mathrm{e} & \sqcup & \mathrm{n} & \mathrm{o} & \mathrm{t} & \sqcup & \mathrm{g} & \mathrm{r} & \mathrm{o} \\ \mathrm{n} & \mathrm{o} & \mathrm{t} & \sqcup & \mathrm{g} & \mathrm{r} & \mathrm{o} & \mathrm{u} & \mathrm{n} & \mathrm{d} \\ \sqcup & \mathrm{g} & \mathrm{r} & \mathrm{o} & \mathrm{u} & \mathrm{n} & \mathrm{d} & \sqcup & \mathrm{f} & \mathrm{o} \end{bmatrix}; \mathbf{y} = \begin{bmatrix} \mathrm{o} \\ \mathrm{g} \\ \mathrm{u} \\ \sqcup \\ \mathrm{r} \end{bmatrix}$$

Generating Character Sequences with LSTMs

In addition, Chollet uses a “temperature” function to transform the probability distribution: sharpen or damp it: $\exp(\frac{\log(x)}{temp}) = x^{\frac{1}{temp}}$

```
def sample(preds, temperature=1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

with the input `[0.2, 0.5, 0.3]`, we obtain:

- Temperature = 2, `[0.26275107 0.41544591 0.32180302]`
- Temperature = 1, `[0.2 0.5 0.3]`
- Temperature = 0.5 `[0.10526316 0.65789474 0.23684211]`
- Temperature = 0.2 `[0.00941176 0.91911765 0.07147059]`

Code Example

From Chollet's github repository:
Jupyter Notebooks: https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/first_edition/8.1-text-generation-with-lstm.ipynb

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 neural 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.

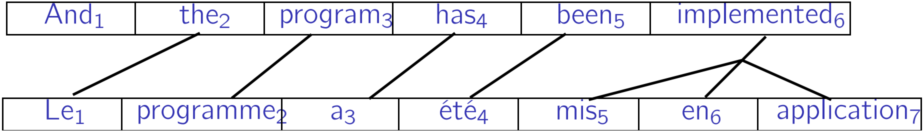
Neural nets are now dominant

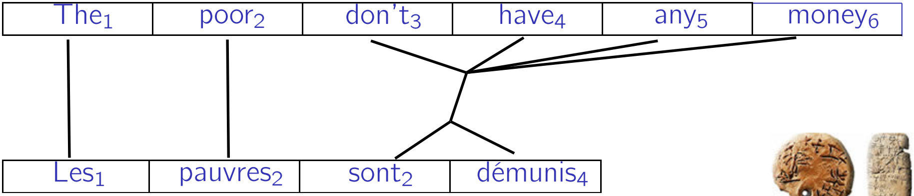
Parallel Corpora (Swiss Federal Law)

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.

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`):

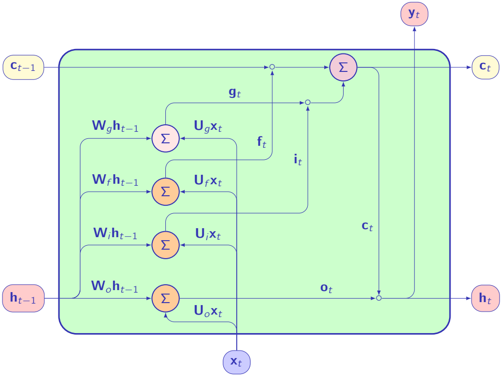
1. The simplest strategy for general sequence learning is to map the input sequence to a fixed-sized vector using one RNN, and then to map the vector to the target sequence with another RNN [...]
2. it would be difficult to train the RNNs due to the resulting long term dependencies [...]. However, the 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 LSTM Architecture



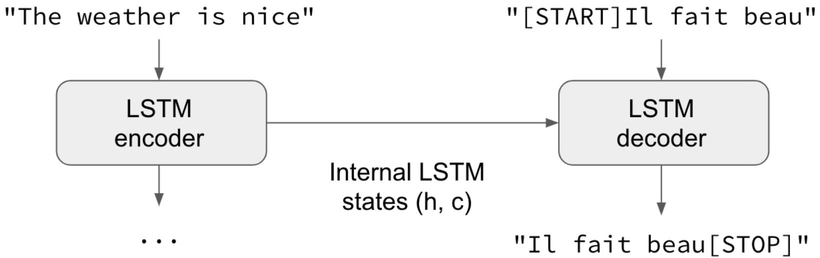
An LSTM unit showing the data flow, where $\mathbf{g}_t$ is the unit input, $\mathbf{i}_t$, the input gate, $\mathbf{f}_t$, the forget gate, and $\mathbf{o}_t$, the output gate. The activation functions have been omitted

Sequence-to-Sequence Translation

We follow and reuse: `https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-i` `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

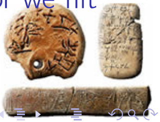


From https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-i
html

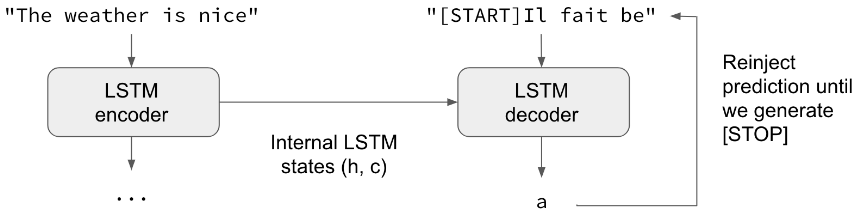
Inference

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-in
html

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