Transformers in Language, Speech and Vision Processing

Marc Evrard, Camille Guinaudeau, François Yvon

LISN — CNRS and Université Paris-Saclay

Transformers in Text and Speech Processing

Humane-AI

Part I

Transformers in Language Processing

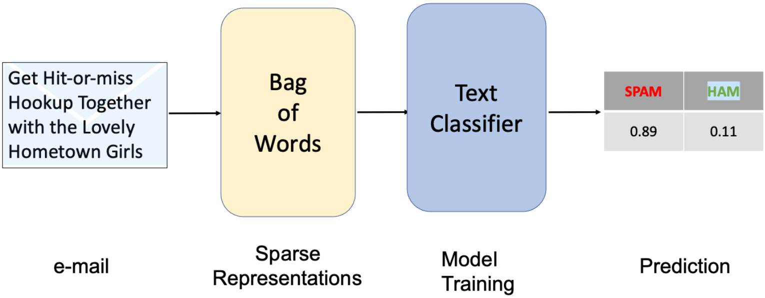
The simplest NLP task: filtering SPAM

- Two types of messages (X) : SPAM ($Y = 0$) and HAM ($Y = 1$)
- Probabilistic decision $y^* = \operatorname{argmax} \mathrm{P}(Y = y | X)$
- Numeric representations for texts: $X \rightarrow E(X)$
  eg. bag-of-words: $E(X) = [c(w_1), c(w_2), \ldots, c(w_N)]$ for a fixed vocabulary $w_1 \ldots w_N$
- Probabilistic model (here: linear, parametric):

$$P(Y = 1|x) \propto \exp \boldsymbol{\theta}^T E(x)$$

- Train $\boldsymbol{\theta}$ with supervision data $\{(x^i, y^i), i = 1 \ldots n_D\}$

The simplest NLP task: filtering SPAM



The simplest NLP task: filtering SPAM

An abstraction for so many problems and tasks

- sentence segmentation: is this contextualized character an EOS ?
- Part-of-speech labelling: is this contextualized word a ?
- prepositional attachement : head = V ou head = N ?
- spell checking : word = *there* ou word = *their ?*
- semantic disambiguation: *bank* = BANK/1 ou BANK/2?
- coreference resolution : (*Marie*, *elle*) coreferent ?, *il* referential ?
- textual entailment : does e_1 imply e_2?
- sentiment analysis : is this text positive / negative / neutral, useful / useless ?
- stance analysis : is this text arguing for / against / neutral ?
- extractive summarization: is this information new / redundant ?

The simplest NLP task: filtering SPAM

The essence of text classification

What we want, what we know:

- turn input X into a (contextual) representation $E(X)$: requires expertise
- train parametric decision function from data : requires supervision

Where NLP has improved:

- **trainable** $E(X)$, **jointly** learned with parameters $\boldsymbol{\theta}$
- generic pre-trained "contextualizers"

.

Let us see how

Language models: probabilistic models for sequences

The simplest model for sequences over a finite alphabet: *n*-gram

$$
\begin{aligned}
\mathrm{P}(w_1 \dots w_L) &= \prod_{i=1}^{L} \mathrm{P}(w_i \mid w_1 \dots w_{i-1}) \qquad (1)\\
&= \prod_{i=1}^{L} \mathrm{P}(w_i \mid w_{i-n+1} \dots w_{i-1}) \qquad (2)
\end{aligned}
$$

(1) is always true. (2) makes a Markov assumption: given short term history $h = w_{i-n+1} \dots w_{i-1}$, words further away do not matter.

Language models: probabilistic models for sequences

The simplest model for sequences over a finite alphabet: *n*-gram

$$\mathrm{P}(w_1 \dots w_L) = \prod_{i=1}^{L} \mathrm{P}(w_i \mid w_1 \dots w_{i-1}) \quad (1)$$

$$= \prod_{i=1}^{L} \mathrm{P}(w_i \mid w_{i-n+1} \dots w_{i-1}) \quad (2)$$

(1) is always true. (2) makes a Markov assumption: given short term history $h = w_{i-n+1} \dots w_{i-1}$, words further away do not matter.

n-gram text generation with [ancestral] sampling

$$w_1 \sim \mathrm{Unif}(W_1); w_2 \sim \mathrm{P}(W_2 \mid w_1); w_3 \sim \mathrm{P}(W_3 \mid w_2 w_1) \dots$$

No length model: make sure to know when to stop

Language models: probabilistic models for sequences

The simplest model for sequences over a finite alphabet: n-gram

$$\mathrm{P}(w_1 \dots w_L) = \prod_{i=1}^{L} \mathrm{P}(w_i \mid w_1 \dots w_{i-1}) \tag{1}$$

$$= \prod_{i=1}^{L} \mathrm{P}(w_i \mid w_{i-n+1} \dots w_{i-1}) \tag{2}$$

(1) is always true. (2) makes a Markov assumption: given short term history $h = w_{i-n+1} \dots w_{i-1}$, words further away do not matter.

n-gram language Id with Bayes rule

$$\mathrm{P}(w_1 \dots w_L; L_1) \geq \mathrm{P}(w_1 \dots w_L; L_2) \Rightarrow \mathrm{P}(L1 \mid w_1 \dots w_L) \geq \mathrm{P}(L_2 \mid w_1 \dots w_L) \tag{3}$$

n-gram models are so simple, yet so difficult

Learning with cheap / free supervision

Parameter estimation just needs data

Maximum likelihood estimates (with 2-word histories: trigrams)

$$\mathrm{P}(w \mid uv) = \frac{c(uvw)}{\sum_{w'} c(uvw')} \tag{4}$$

$c()$ is the count function, $h = uv$ is the history

The art of language modeling

- with $100,000$ words, $100,000^3$ 3-gram counts, most of them 0
- build history classes ($uv \rightarrow h(uw)$) to keep models small
- building history classes ? the science of count smoothing [1992-2012]

n-gram models are so simple, yet so difficult

Learning with cheap / free supervision

Parameter estimation just needs data

Maximum likelihood estimates (with 2-word histories: trigrams)

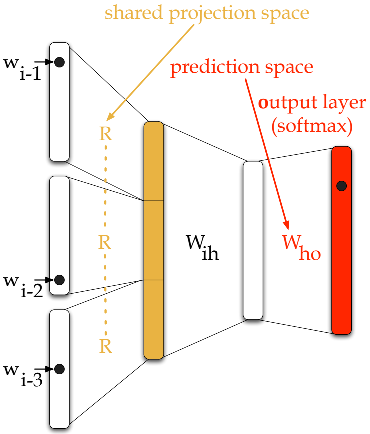
$$\mathrm{P}(w \mid uv) = \frac{c(uvw)}{\sum_{w'} c(uvw')} \tag{4}$$

$c()$ is the count function, $h = uv$ is the history

The art of language modeling

- with $100,000$ words, $100,000^3$ 3-gram counts, most of them 0
- build history classes ($uv \rightarrow h(uw)$) to keep models small
- building history classes ? the science of count smoothing [1992-2012]

Feed-forward language models [Bengio et al., 2003]



$$\boldsymbol{i} = [w_{i-1}^T R, w_{i-2}^T R, w_{i-3}^T R]$$
$$\boldsymbol{h} = \boldsymbol{i}^T W_{ih} + \boldsymbol{b}_{ih}$$
$$\boldsymbol{o} = \tanh(\boldsymbol{h})^T W_{ho} + \boldsymbol{b}_{ho}$$

$$\mathrm{P}(w_i \mid w_{i-3}, w_{i-2}, w_{i-1}) = \frac{\exp \boldsymbol{o}[w_i]}{\sum_w \exp \boldsymbol{o}[w]}$$

- encodes context as $\phi(w_{i-3}, w_{i-2}, w_{i-1})$
- compares $\phi(w_{i-3}, w_{i-2}, w_{i-1})$ and $R(w_i)$

Feed-forward language models [Bengio et al., 2003]

Training FFLMs - maximize log-likelihood [aka cross-entropy]

$$\boldsymbol{\theta}^* = [\mathbf{R}, \mathbf{W}_{ih}, \boldsymbol{b}_i, \mathbf{W}_{ho}, \boldsymbol{b}_o] = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_i \log \frac{\exp \boldsymbol{o}[w_i]}{\sum_w \exp \boldsymbol{o}[w]}$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_i \boldsymbol{o}[w_i] - \log(\sum_w \exp \boldsymbol{o}[w])$$

jointly learns representations / word embeddings ($\mathbf{R}$) and decision rule

- optimize through stochastic gradient and back propagation (chain rule)
- $\operatorname{softmax}(\boldsymbol{x}) = \frac{\exp \boldsymbol{x}}{\sum_k exp \boldsymbol{x}[k]}$ computes dense distributions
- also influencial log-bilinear model [Mnih and Hinton, 2007]: history words are summed, no hidden layer
- computationally demanding (softmax layer)
- superior to discrete (n-gram) LMs across the board [Schwenk, 2007, Le et al., 2012]

Feed-forward language models [Bengio et al., 2003]

FFLMs induce similarities between histories and between words (from [Le et al., 2010])

word (freq.)	*model*	*5 nearest neighbors*
is	standard	was are were been remains
900, 350	1 vector init.	was are be were been
conducted	standard	undertaken launched $270,900 Mufamadi 6.44-km-long
18, 388	1 vector init.	pursued conducts commissioned initiated executed
Cambodian	standard	Shyorongi $3,192,700 Zairian depreciations teachers'
2, 381	1 vector init.	Danish Latvian Estonian Belarussian Bangladeshi
automatically	standard	MSSD Sarvodaya $676,603,059 Kissana 2,652,627
1, 528	1 vector init.	routinely occasionally invariably inadvertently seldom
Tosevski	standard	$12.3 Action,3 Kassouma 3536 Applique
34	1 vector init.	Shafei Garvalov Dostiev Bourloyannis-Vrailas Grandi
October-12	standard	39,572 anti-Hutu $12,852,200 non-contracting Party's
8	1 vector init.	March-26 April-11 October-1 June-30 August4
3727th	standard	Raqu Tatsei Ayatallah Mesyats Langlois
1	1 vector init.	4160th 3651st 3487th 3378th 3558th

1 vector init: share parameters R and W_{ho} during init.

Computational complexity of FFLM

Speeding up the softmax computation

Training objective computes a large sum

$$\ell = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_i \boldsymbol{o}[w_i] - \log(\sum_{w'} \exp \boldsymbol{o}[w])$$

Shortlist-based models [Schwenk, 2007] combine discrete and continuous LMs

$$\mathrm{P}(w \mid h) = \begin{cases} \mathrm{P}_{NN}(w \mid h) \text{ if } w \in \text{shorlist} \\ \alpha(h)\, \mathrm{P}_{KN}(w \mid h) \text{otherwise} \end{cases}$$

$\alpha(h)$ rescales $\mathrm{P}_{KN}(\,|\,)$ for normalization

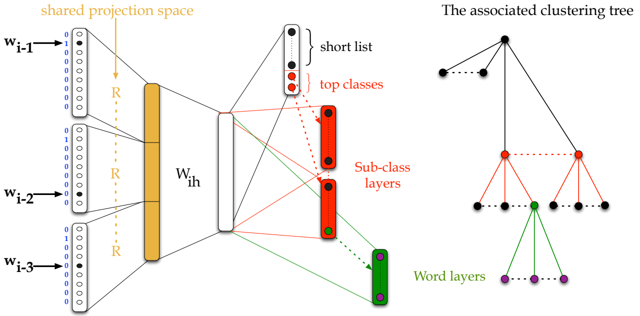
Computational complexity of FFLM

Speeding up the softmax computation

Training objective computes a large sum

$$\ell = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_i \boldsymbol{o}[w_i] - \log(\sum_{w'} \exp \boldsymbol{o}[w])$$

Hierarchical models [Mnih and Teh, 2012, Le et al., 2013] compute a hierarchical softmax



Recurrent Neural Networks as LMs [Mikolov et al., 2010]

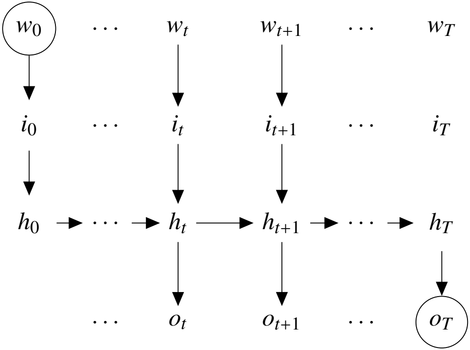
From finite to infinite contexts $\mathrm{P}(w_t \mid w_{<t})$

words: $\boldsymbol{w}_t, t = 1 \dots T, \in \{0, 1\}^{|V|}$

embeddings: $\boldsymbol{i}_t, t = 1 \dots T, \in \mathbb{R}^d$

hidden states: $\boldsymbol{h}_t, t = 1 \dots T, \in \mathbb{R}^p$

output: $\boldsymbol{o}_t, t = 1 \dots T, \in \mathbb{R}^{|V|}$



$$\boldsymbol{i}_t = [w_t^T R]$$

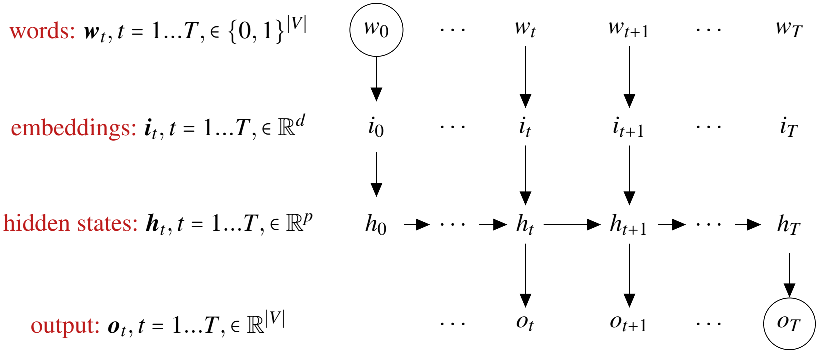
$$\boldsymbol{h}_t = \tanh(\boldsymbol{i}_t^T W_{ih} + \boldsymbol{h}_{t-1}^T W_{hh} + \boldsymbol{b}_{ih})$$

$$\boldsymbol{o}_t = \boldsymbol{h}^T W_{ho} + \boldsymbol{b}_{ho}$$

depends on all past time steps t

Recurrent Neural Networks as LMs [Mikolov et al., 2010]

From finite to infinite contexts $\mathrm{P}(w_t \mid w_{<t})$

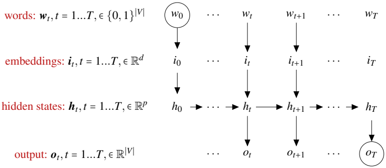


$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_t \log \boldsymbol{o}[w_i] - \log(\sum_w \exp \boldsymbol{o}[w])$$

$$\mathrm{P}(w_{t+1} = k \mid w_{<t}; \theta_{LM}) = \mathrm{softmax}(\boldsymbol{o}_t = W^{ho}\boldsymbol{h}_t + \boldsymbol{b}^o)[k]$$

Recurrent Neural Networks as LMs [Mikolov et al., 2010]

From finite to infinite contexts $\mathrm{P}(w_t \mid w_{<t})$



- train with word prediction objective and cross-entropy loss
- generate through ancestral sampling, one word at a time
- more complex cells ($(\boldsymbol{w}_t, \boldsymbol{h}_t) \rightarrow \boldsymbol{h}_{t+1}$): GRUs, LSTMs
- same issues with $\mathrm{softmax}$; same solutions apply
- stack several hidden layers $\boldsymbol{h}_t^k = f(\boldsymbol{h}_{t-1}^k, \boldsymbol{h}_t^{k-1})$: biRNNs, etc.
- backwards processing computes $\bar{\boldsymbol{h}}_{-1}$
- $[\boldsymbol{h}_t, \bar{\boldsymbol{h}}_t]$ represents word w_t and its context
- $[\boldsymbol{h}_T, \bar{\boldsymbol{h}}_{-1}]$ a better representation: text classification, etc.

Memory cells in RNNs

Mitigating vanishing / exploding gradient

Vanilla RNNs update hidden cells with:

$$\begin{aligned}
\boldsymbol{h}_t &= \tanh(\boldsymbol{i}_t^T W_{ih} + \boldsymbol{h}_{t-1}^T W_{hh} + \boldsymbol{b}_{ih}) \\
&= \tanh(\boldsymbol{i}_t^T W_{ih} + \tanh(\boldsymbol{i}_{t-1}^T W_{ih} + \boldsymbol{h}_{t-2}^T W_{hh} + \boldsymbol{b}_{ih})^T W_{hh} + \boldsymbol{b}_{ih}) \\
\frac{\delta \boldsymbol{h}_t}{\delta \theta} &= (1 - tanh(\boldsymbol{h}_t)^2) \frac{\delta}{\delta \theta} (\boldsymbol{i}_t^T W_{ih} + \boldsymbol{h}_{t-1}^T W_{hh} + \boldsymbol{b}_{ih}) \\
&= (1 - tanh(\boldsymbol{h}_t)^2) \left((\ldots) \frac{\delta \boldsymbol{h}_{t-1}}{\delta \theta}^T W_{hh} + \boldsymbol{h}_{t-1}^T \frac{\delta W_{hh}}{\delta \theta} (\ldots) \right)
\end{aligned}$$

- Gradient wrt W_{ih} and W_{hh} used multiple times;
- Information squashing through $\tanh()$

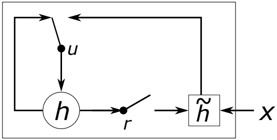
$\Rightarrow$ Unstable results, hardly better than very long range FFLMs

Memory cells in RNNs

Mitigating vanishing / exploding gradient

The Gated Recurrent Unit of [Cho et al., 2014b] learn to manipulate hidden states [vectors]: which part should be copied forward? which part should be forgotten?

$$
\begin{aligned}
\boldsymbol{u}_t &= \sigma(W_{iu}\boldsymbol{i}_t + W_{hu}\boldsymbol{h}_{t-1}) \in [0;1]^d \\
\boldsymbol{r}_t &= \sigma(W_{ir}\boldsymbol{i}_t + W_{hr}\boldsymbol{h}_{t-1}) \in [0;1]^d \\
\tilde{\boldsymbol{h}}_t &= \tanh(\boldsymbol{i}_t^T W_{ih} + (\boldsymbol{h}_{t-1} \odot \boldsymbol{r}_t)^T W_{hh} + \boldsymbol{b}_{ih}) \\
\boldsymbol{h}_{t+1} &= (1 - \boldsymbol{u}_t) \odot \boldsymbol{h}_t + \boldsymbol{u}_t \odot \tilde{\boldsymbol{h}}_t
\end{aligned}
$$



$\sigma()$: sigmoid function, acts as a soft gate:

- $\boldsymbol{r}_t$ resets components of previous state;
- $\boldsymbol{u}_t$ selects new or past hidden state without squashing.

Memory cells in RNNs

Mitigating vanishing / exploding gradient

LSTMs cells [Hochreiter and Schmidhuber, 1997] implement a richer update mechanism as:

$$
\begin{aligned}
\boldsymbol{f}_t &= \sigma(W_{if}\boldsymbol{i}_t + W_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \in [0;1]^d \text{ forget gate} \\
\boldsymbol{e}_t &= \sigma(W_{ie}\boldsymbol{i}_t + W_{he}\boldsymbol{h}_{t-1} + \boldsymbol{b}_e) \in [0;1]^d \text{ input gate} \\
\tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{e}_t^T W_{ic} + \boldsymbol{h}_{t-1}^T W_{hc}) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \\
\boldsymbol{o}_t &= \sigma(W_{io}\boldsymbol{i}_t + W_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \in [0;1]^d \text{ output gate} \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)
\end{aligned}
$$

- $\boldsymbol{h}_t$ represents the hidden state;
- $\boldsymbol{c}$ is the memory cell, part of which is copied forward (no squashing)

Memory cells in RNNs

Mitigating vanishing / exploding gradient

Some lessons learned

- Gated units much better than Vanilla RNN
- GRUs simpler (and faster) than LSTMs
- GRUs and LSTMs equivalent (performance-wise)
- Multiple layers help
- Good implementations are tricky [Merity et al., 2018]: require dropout, improved optimizer, parameter sharing, etc.
- Hyper-parameter search is essential [Melis et al., 2018]

RNNs encode words, RNNs also encode sentences

Solving sentence classification

$\boldsymbol{h}_T = \mathrm{RNN}(w_1 \dots w_T)$ encodes a variable-length sentence in a fixed-length vector.
Decision rule for sentiment analysis, mapping sentences to polarity value (positive, negative, neutral): $w_1 \dots w_T \rightarrow y$

$$\mathrm{P}(y = 1 \mid w_1 \dots w_T; \boldsymbol{\theta}) = \sigma(\boldsymbol{W}^T \boldsymbol{h}_T + b) \qquad \text{sigmoid, again}$$

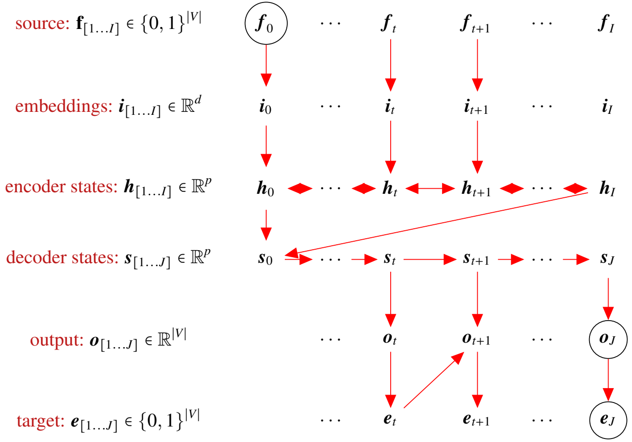
$$\boldsymbol{\theta}^* = \operatorname{argmax} \sum_i \log \mathrm{P}(y^{(i)} \mid w_1^{(i)} \dots w_{T(i)}^{(i)})$$

- improves classification results with multiple layers,
- works for all sentence-level classification (textual entailment, stance classification, etc)
- even better: use $[\boldsymbol{h}_T, \bar{\boldsymbol{h}}_{-1}]$

How about Machine Translation?

Recurrent Neural Networks for Machine Translation

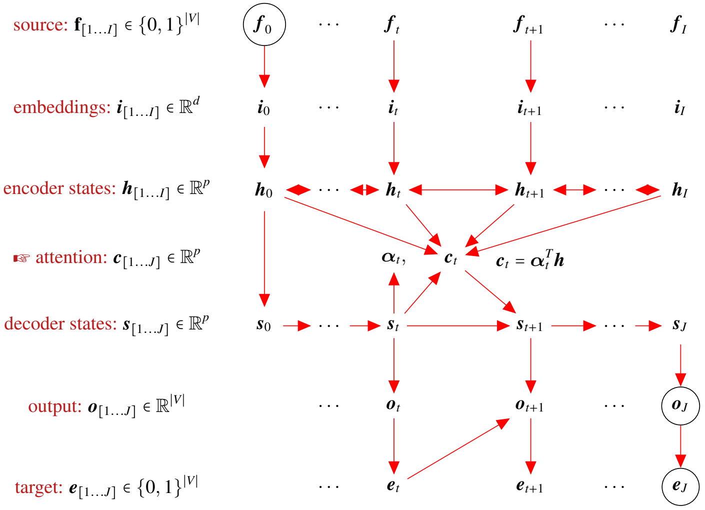
A simple bilingual conditional Langage Model [Cho et al., 2014a]



$$P(e_{t+1} = k | \mathbf{e}_{\leq t}, \mathbf{f}; \theta_{NMT}) = \left[\mathrm{softmax}(o_{t+1} = W^{so} s_{t+1} + +W^{eo} e_t + b^o)\right]_k$$

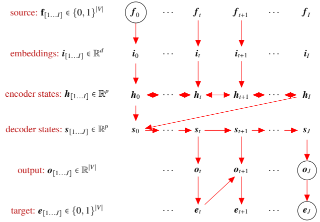
Recurrent Neural Networks for Machine Translation

Attentional NMT: better know what to translate [Bahdanau et al., 2015]



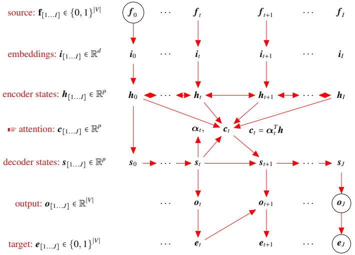
Recurrent Neural Networks for Machine Translation

Equations of the RNN + attention



Recurrent Neural Networks for Machine Translation

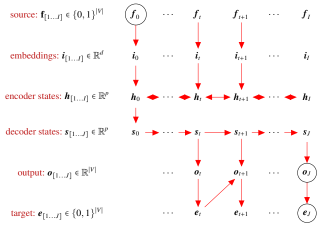
Equations of the RNN + attention



$$
\begin{aligned}
h_i &= \phi(f_i, h_{i-1}) \quad \forall i \in [1 \ldots I] \\
\boldsymbol{\alpha}_{ti} &= \mathrm{softmax}(\boldsymbol{h}^T s_{t-1}) \quad \forall t \in [1 \ldots J], i \in [1 \ldots I] \\
c_t &= \sum_t \alpha_{ti} h_i \quad \forall t \in [1 \ldots J] \\
P(e_t = k | \mathbf{e}_{<t}, \mathbf{f}; \theta_{NMT}) &= [\mathrm{softmax}(o_t = W^{so} s_t + W^{co} c_t + W^{eo} e_{t-1} + b^o)]_k \\
s_{t+1} &= \phi(c_t, s_t) \qquad \forall t \in [1 \ldots J,], \text{with } \phi() = \text{LSTM or GRU or ...}
\end{aligned}
$$

Recurrent Neural Networks for Machine Translation

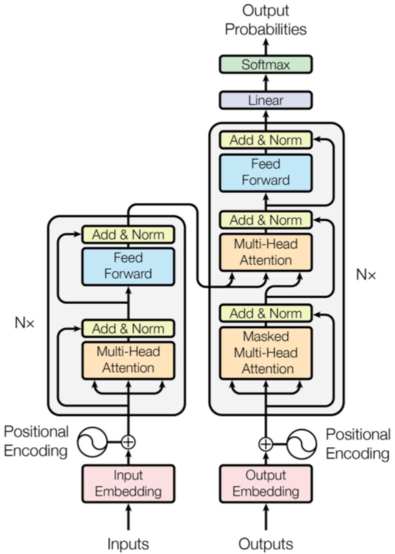
Equations of the RNN + attention

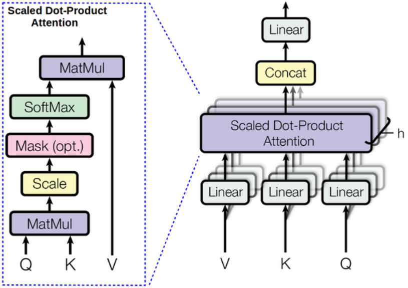


- training (including attention) remains end-to-end
- RNN recursions make training slow
- attention ≈ ? (word) alignment

Faster, Better Encoder-Decoder + Attention : Transformer

Images (C) [Vaswani et al., 2017]





Faster, Better Encoder-Decoder + Attention : Transformer

Heads in Multi-Head Attention

$$\text{The query: } \mathbf{H}_q \in \mathbb{R}^{d_{\text{kv}}} \times \mathbb{R}^{d_{\text{md}}}$$

$$\text{The key: } \mathbf{H}_k \in \mathbb{R}^{d_{\text{kv}}} \times \mathbb{R}^{d_{\text{md}}}$$

$$\text{The value: } \mathbf{H}_v \in \mathbb{R}^{d_{\text{kv}}} \times \mathbb{R}^{d_{\text{md}}}$$

Heads linearly transform matrice $\boldsymbol{I} \in \mathbb{R}^{d_{\text{md}}} \times \mathbb{R}^T$ into matrice $\boldsymbol{O}$ in $\mathbb{R}^{d_{\text{kv}}} \times \mathbb{R}^T$

$$\text{transform input matrix for words: } \mathbf{Q} = \boldsymbol{H}_q \times \mathbf{I} \in \mathbb{R}^{d_{\text{kv}}} \times \mathbb{R}^T$$

$$\text{transform input matrix for contexts: } \mathbf{K} = \boldsymbol{H}_k \times \mathbf{I} \in \mathbb{R}^{d_{\text{kv}}} \times \mathbb{R}^T$$

$$\text{transform input matrix for outputs: } \mathbf{V} = \boldsymbol{H}_v \times \mathbf{I} \in \mathbb{R}^{d_{\text{kv}}} \times \mathbb{R}^T$$

$$\text{compute similarities words/context: } \mathbf{D} = \mathbf{Q}^T \times \mathbf{K} \in \mathbb{R}^T \times \mathbb{R}^T$$

$$\text{compute linear weights: } \tilde{\mathbf{D}} = \text{softmax}\left(\frac{\mathbf{D}}{\sqrt{d_{\text{kv}}}}\right) \in \mathbb{R}^T \times \mathbb{R}^T \text{ columnwise}$$

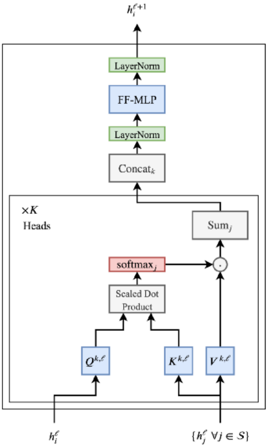
$$\text{linear combination of cols: } \boldsymbol{O} = \tilde{\boldsymbol{D}} \times \mathbf{V} \in \mathbb{R}^T \times \mathbb{R}^{d_{\text{kv}}}$$

Faster, Better Encoder-Decoder + Attention : Transformer

The head computation, columnwise

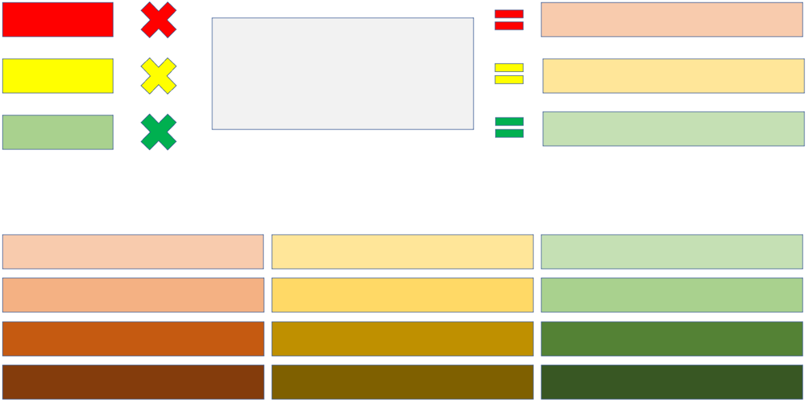
$$\boldsymbol{O}_t = \sum_{s=1}^{T} \mathrm{softmax}\left(\frac{[\mathbf{H}_q \boldsymbol{I}_t]^T [\mathbf{H}_k \boldsymbol{I}_s]}{\sqrt{d_{\mathrm{kv}}}}\right) \mathbf{H}_v \boldsymbol{I}_s$$

$$= \sum_{s=1}^{T} \mathrm{softmax}\left(\frac{\mathbf{Q}_t^T \mathbf{K}_s}{\sqrt{d_{\mathrm{kv}}}}\right) \mathbf{V}_s$$

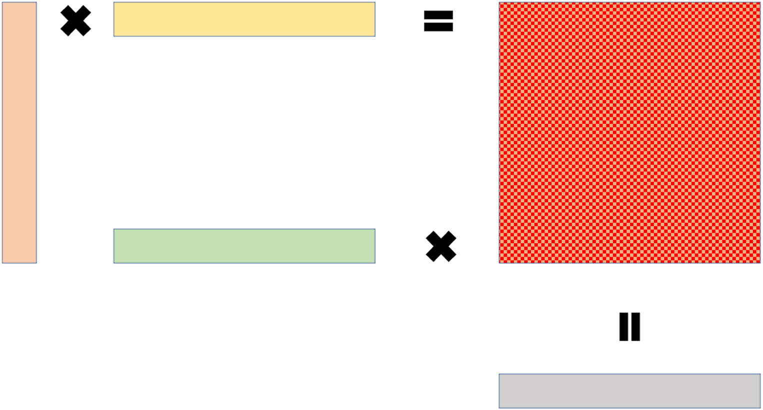


Each output column is a linear combination of all the input columns.

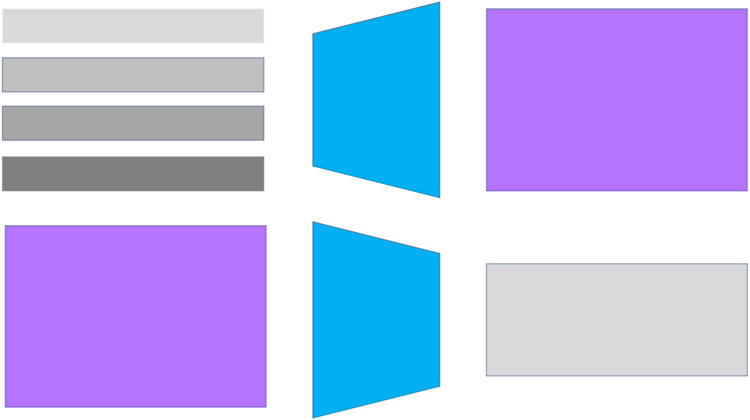
Faster, Better Encoder-Decoder + Attention : Transformer



Faster, Better Encoder-Decoder + Attention : Transformer



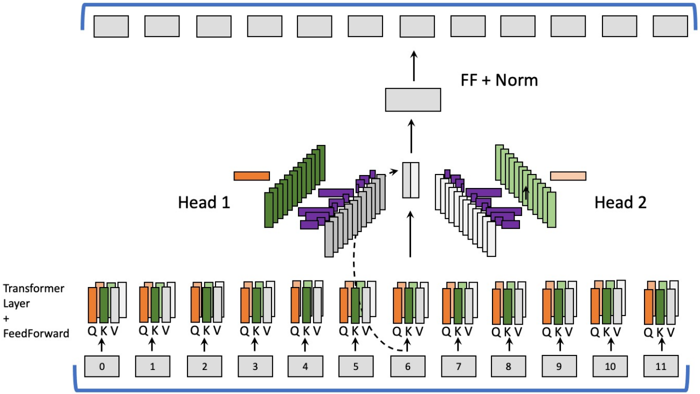
Faster, Better Encoder-Decoder + Attention : Transformer



Faster, Better Encoder-Decoder + Attention : Transformer

When using multiple heads

- one Head : $\mathbf{I}(d_{\mathrm{md}} \times T) \rightarrow \mathbf{O}(d_{\mathrm{kv}} \times T)$
- k Heads : $\mathbf{I}(d_{\mathrm{md}} \times T) \rightarrow [\mathbf{O}_1, \ldots, \mathbf{O}_k](kd_{\mathrm{kv}} \times T)$



Faster, Better Encoder-Decoder + Attention : Transformer

Using multiple heads

- one Head : $\mathbf{I}(d \times T) \rightarrow \mathbf{O}(o \times T)$
- k Heads : $\mathbf{I}(d \times T) \rightarrow [\mathbf{O}_1, \ldots, \mathbf{O}_k](ko \times T)$

Using multiple layers of multiple heads

- compute with k Heads : $\mathbf{I}(d_{\mathrm{md}} \times T) \rightarrow \mathbf{O} = [\mathbf{O}_1 \ldots \mathbf{O}_k](kd_{\mathrm{kv}} \times T)$
- enable: residual (direct) connections $\mathbf{O'} = \mathbf{O} + \mathbf{I}$
- pass $\mathbf{O'}$ through a "linear" layer $\mathbf{O''} = \mathbf{O'} + \mathbf{W}' \times \mathrm{RELU}(\mathbf{WO})$, with $\mathbf{O''} \in \mathbb{R}^{(d \times T)}$
- stack multiple layers $\mathbf{I}_1 \rightarrow \mathbf{I}_2 \rightarrow \mathbf{I}_3 \rightarrow \mathbf{I}_4 \ldots$
- enable residual (direct) connections $\mathbf{I}_{k+1} = \mathbf{O''}_k + \mathbf{O}_k$
- make layers and sublayers comparable through layer normalization (substract mean, divide by stddev)

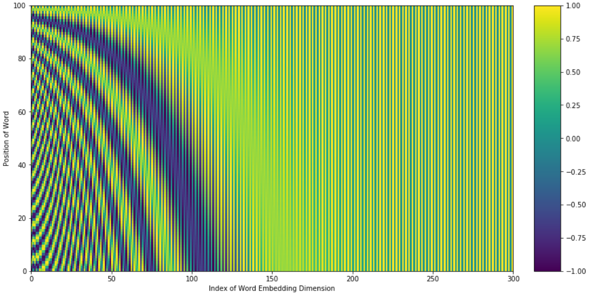
Typical values: 8 Heads of output dimension $o = 64$, 6-12 layers of heads of dimension 512.

Faster, Better Encoder-Decoder + Attention : Transformer

The initial layer: words and positions

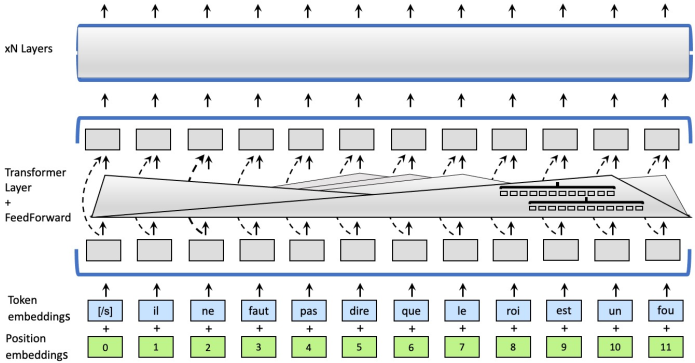
Assuming input $w_1 \dots w_T$ each column in $\mathbf{I_1}$ combines (sums) word embeddings and positional encodings in $\mathbf{P} \in \mathbb{R}^d \times \mathbb{R}^T$.

$$\begin{cases} \mathbf{P}[2i, t] = \sin(t/10000^{2i/d}) \\ \mathbf{P}[2i+1, t] = \cos(t/10000^{2i/d}) \end{cases}$$



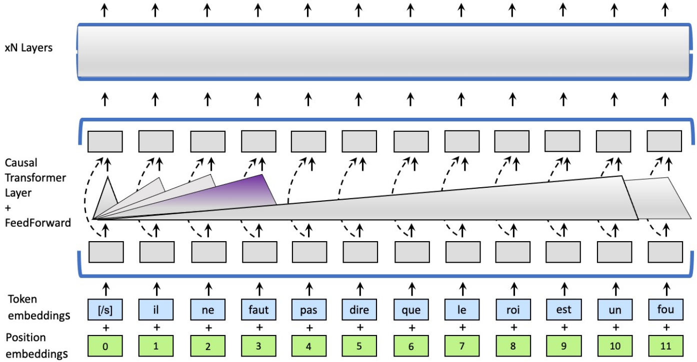
☞ https://medium.com/swlh/elegant-intuitions-behind-positional-encodings-dc48b4a4a5d1

Faster, Better Encoder-Decoder + Attention : Transformer



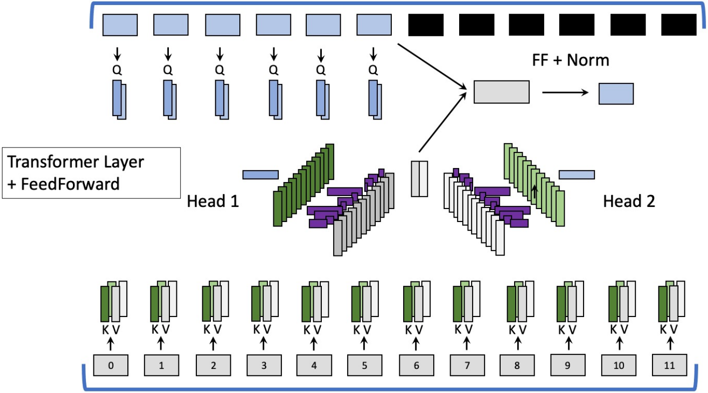
The encoder side, a complete view

Faster, Better Encoder-Decoder + Attention : Transformer



In the decoder: masked, **causal**, self-attention

Faster, Better Encoder-Decoder + Attention : Transformer



- cross-attention queries Q derive from the current **decoder** layer,
- keys and values from the last encoder layer

Faster, Better Encoder-Decoder + Attention : Transformer

Output layer and prediction

- project output of K^{th} layer into $\mathbb{R}^V$ to get logits: $g(\mathbf{W}_{os}\boldsymbol{S}_K[t] + \boldsymbol{b}_o)$
- use $\mathrm{softmax}$ to predict next target word e_t
- collect gradients for training

Also:

- trainable positional encodings
- parameter sharing in the decoder
- variants of the FF layer
- better layer normalisation
- computational speed ups for the attention computation

Faster, Better Encoder-Decoder + Attention : Transformer

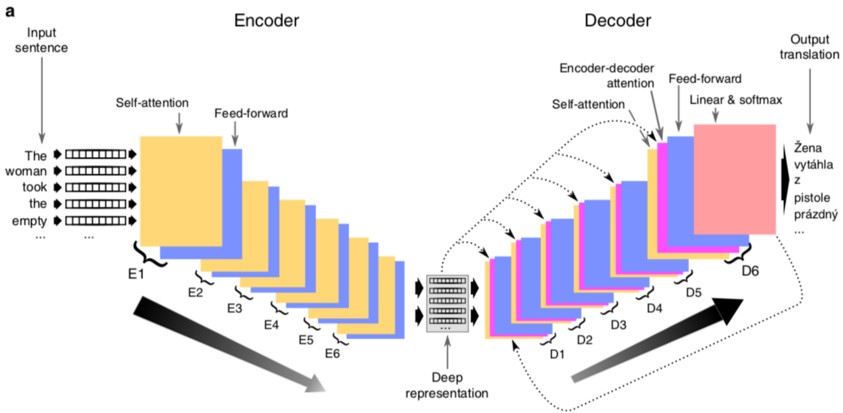
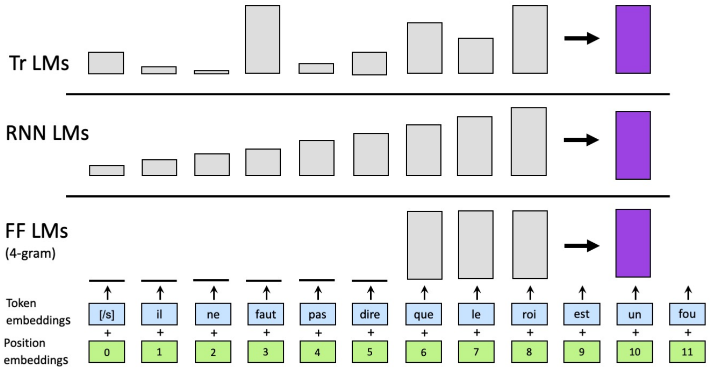


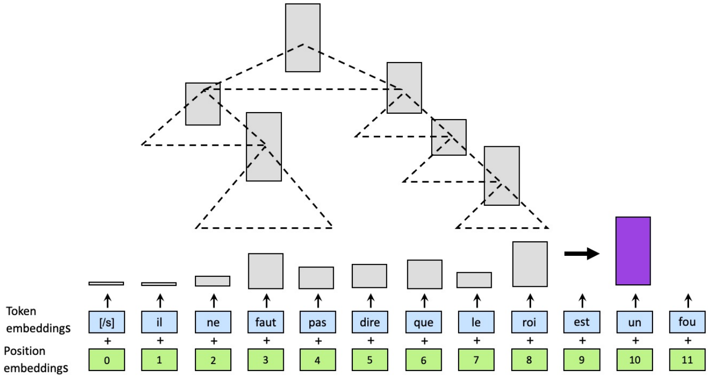
Image from [Popel et al., 2020]

Transformers as “pure” LMs: improving history



Also: relax causality, recompute past representations after each new word [Raffel et al., 2020]

Transformers as “pure” LMs: improving history



Computational issues with Transformers

Attention is quadratic

$$\forall i,j \in [1 \ldots T], \alpha_{i,j} = \mathrm{softmax}\left(\frac{Q_i^T K_j}{\sqrt{d}}\right)$$

The X-former family (surveyed in [Tay et al., 2020])

- Compress: Memory-compressed Transformer
- Approximate dot product with LSH: Reformer
- Use hierararchical attention binary-Tree Transformer
- Use local attention + global (random) states: Sparse Transformer, Longformer, Big Bird
- Approximate dot product with low-rank matrices: Linformer

Contexts up to hundreds of past tokens

Computational issues with Transformers

$\lvert V \rvert$	T	K	L	d_{md}	d_{kv}	d_{ff}	Par. (L)	Par. (I)
32k	512	8	6	512	64	2048	32,8m	49,9m
32k	512	12	12	768	64	3072	49,2m	127m
32k	512	16	24	1024	64	4096	65,5m	342m
32k	512	32	24	1024	128	16384	65,5m	2,38b
32k	512	128	24	1024	128	65536	65,5m	28,7b

Typical dimensions for large scale real-world Transformer models

The infamous < unk >nown word

Closed world assumption

- The support of LM: a fixed vocab V. Sentences with unknowns have 0 probability.
- The support of LM: a fixed vocab $V \cup \{ <\mathrm{unk}> \}$.
  Estimation: all words $\notin V$ are unked [makes < unk > very likely].
- Variant: consider classes of < unk > (proper names, numbers, etc).

Subword units: morphemes, char ngrams, etc

- morph-based LM: require morphogical analysis, < unk >still possible
- letters: no more unknown words - unknown symbols instead ?
- a mixture of words and letters

Shorter units require longer histories [estimation problems], imply longer sentences [computational problems].

The infamous $<$ unk $>$nown word

Closed world assumption

- The support of LM: a fixed vocab V. Sentences with unknowns have 0 probability.
- The support of LM: a fixed vocab $V \cup \{ < \text{unk} > \}$.
  Estimation: all words $\notin V$ are unked [makes $<$ unk $>$ very likely].
- Variant: consider classes of $<$ unk $>$ (proper names, numbers, etc).

Subword units: morphemes, char ngrams, etc

- morph-based LM: require morphogical analysis, $<$ unk $>$still possible
- letters: no more unknown words - unknown symbols instead ?
- a mixture of words and letters

Shorter units require longer histories [estimation problems], imply longer sentences [computational problems].

Subword units in language models: BPEs, wordpieces, etc

Byte pair encoding: *N* deterministic merge operations

1. Make symbol map (greedy)
   `Repeat till done: merge most frequent bigram into a compound symbol`
2. Encode (greedy)
   `split each word into compound symbols`

Example from [Sennrich et al., 2016]

L = { *lower*, *lowest*, *newer*, *wider*, *wide* }

e	r#	3	[er#]	[lo]	w	2	[low]
l	o	2	[lo]	w	i	2	[wi]

Segmentations: [low]+ [er#], [low]+ e+ s+ [t#], n+ e+ w+ [er#], [wid]+ [er#], [wid]+ [e#]

☞ https://github.com/rsennrich/subword-nmt

Computing $\mathrm{P}(w \mid h)$ requires marginalising (summing) over all segmentations of w

Pre-training representations for multi-task learning

The overwhelming majority of these state-of-the-art systems address a benchmark task by applying linear statistical models to adhoc features. In other words, there researchers themselves discover intermediate representations by engineering task-specific features. (...) Although such performance improvements can be very useful in practice, they teach us little about the means to progress toward the broader goals of natural language understanding and the elusive goals of Artificial Intelligence. In this contribution, we try **to excel on multiple benchmarks** while avoiding task-specific enginering. Instead we use a single **learning system able to discover adequate internal representations**. [Collobert et al., 2011]

☞ Image from http://sesamestreet.org

Pre-training representations for multi-task learning

(...) **to excel on multiple benchmarks** while avoiding task-specific enginering. (...) a single **learning system able to discover adequate internal representations**. [Collobert et al., 2011]

A recipe for pre-training

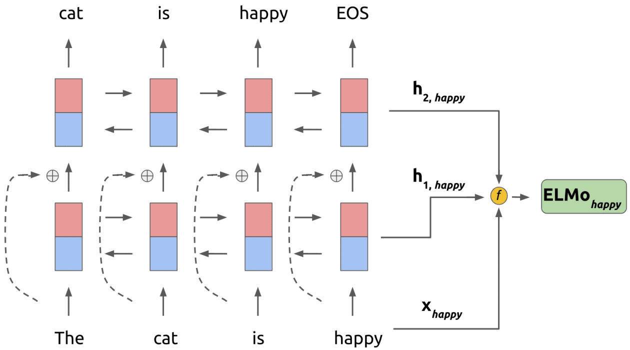
1. train context-free or context-dependent word embeddings on large "general domain" corpus in an unsupervised way.
2. plug-in embeddings into (domain) specific task
3. resume training with a task-dependent loss

Popular implementations:

- ELMO [Peters et al., 2018] uses biRNNs at step1, BERT [Devlin et al., 2019] and GPT-2/3 [Radford et al., 2019] use Transformers
- ELMO and GPT-2/3 use half-contexts and a LM objective, BERT uses full context and two objectives: mask-LM and next sentence prediction

Pre-training representations for multi-task learning

(...) **to excel on multiple benchmarks** while avoiding task-specific enginering. (...) a single **learning system able to discover adequate internal representations**. [Collobert et al., 2011]

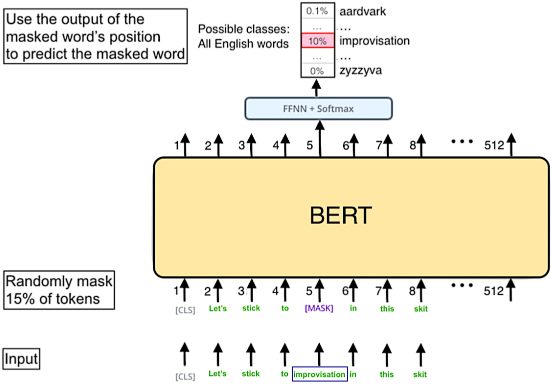


Bottom layer is char n-gram conv. layers + 2 highway layers + linear projection; top layers are bidirectional LSTMs, training objective predicts next word. All layers linearly combined to yield final representation.

☞ Image from https://www.mihaileric.com/posts/deep-contextualized-word-representations-elmo/

Pre-training representations for multi-task learning

(...) **to excel on multiple benchmarks** while avoiding task-specific enginering. (...) a single **learning system able to discover adequate internal representations**. [Collobert et al., 2011]

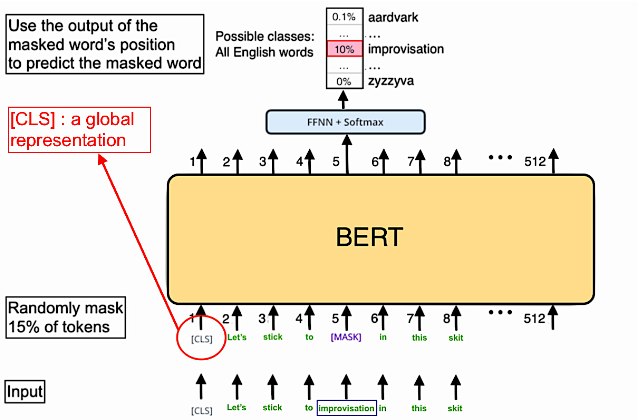


BERT uses a Transformer architecture - Base implementation has 12-24 layers each with 12-16 heads.

☞ Image from https://jalammar.github.io/illustrated-bert/

Pre-training representations for multi-task learning

(...) **to excel on multiple benchmarks** while avoiding task-specific enginering. (...) a single **learning system able to discover adequate internal representations**. [Collobert et al., 2011]

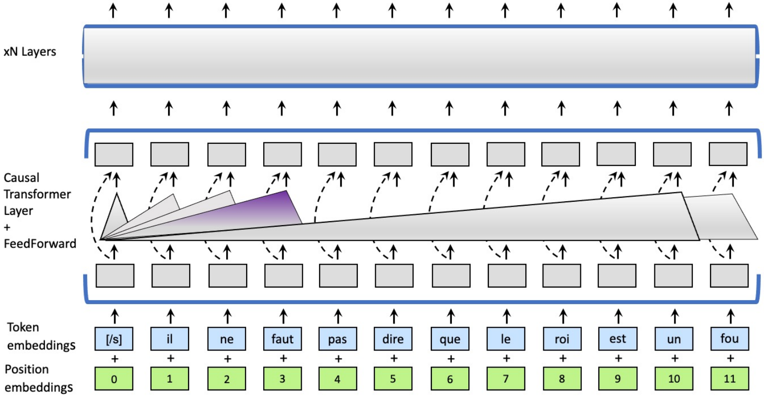


Local and global representations

☞ Image from https://jalammar.github.io/illustrated-bert/

Pre-training representations for multi-task learning

(...) **to excel on multiple benchmarks** while avoiding task-specific enginering. (...) a single **learning system able to discover adequate internal representations**. [Collobert et al., 2011]



GPT, a Transformer with "causal" self-attention, trained with next word prediction

Pre-training representations for multi-task learning

(...) **to excel on multiple benchmarks** while avoiding task-specific enginering. (...) a single **learning system able to discover adequate internal representations**. [Collobert et al., 2011]

The many benefits of LM pre-training

- almost unsupervised learning - leverage huge monolingual corpora
- solve rare "word" issue
- mitigate annotation scarcity
- knowledge transfer between domains or tasks with prompting / priming
  $\Rightarrow$ uses LM as text generator with appropriate initialization

Improve lexical / phrasal / sentential representations accross the board

Pre-training representations for multi-task learning

(...) **to excel on multiple benchmarks** while avoiding task-specific enginering. (...) a single **learning system able to discover adequate internal representations**. [Collobert et al., 2011]

Prompt (ANLI)	The Gold Coast Hotel & Casino is a hotel and casino located in Paradise, Nevada. This locals' casino is owned and operated by Boyd Gaming. The Gold Coast is located one mile west of the Las Vegas Strip on West Flamingo Road. It is located across the street from the Palms Casino Resort and the Rio All Suite Hotel and Casino. Question: The Gold Coast is a budget-friendly casino. True, False, or Neither?
Answer (OK)	Neither
Answer (KO)	True
Answer (KO)	False

Prompt (PIQA)	How to apply sealant to wood.
Answer (OK)	Using a brush, brush on sealant onto wood until it is fully saturated with the sealant.
Answer (KO)	Using a brush, drip on sealant onto wood until it is fully saturated with the sealant.

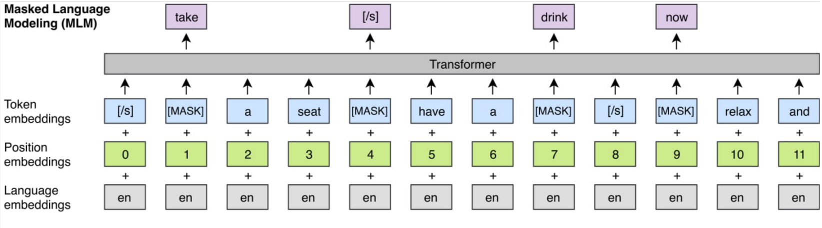
Prompt (COPA)	My body cast a shadow over the grass because
Answer (OK)	the sun was rising.
Answer (KO)	the grass was cut.

☞ exemples from Radford et al. [2019]

Towards multilingual –language agnostic – representations

Computing embeddings such that mutual translations are nearest neighbours

Leaning multilingal contextual embeddings - XLM [Lample and Conneau, 2019]



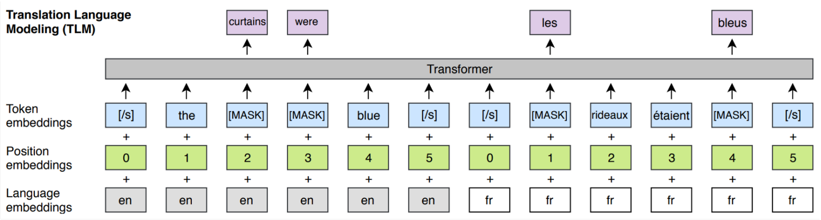
- Train with continuous stream of sentences
- Do not use “next sentence prediction” objective
- Train with language agnostic units and multiple languages

☞ Images © A. Conneau & G. Lample (2018)

Towards multilingual –language agnostic – representations

Computing embeddings such that mutual translations are nearest neighbours

Leaning contextual multilingal embeddings - TLM



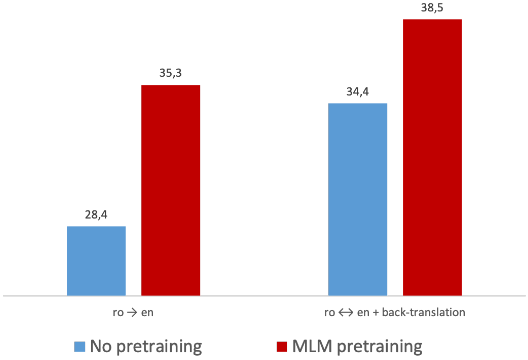
- Learn a shared subword vocabulary
- Train a single Transformer on MLM+TLM using parallel data (supervision)

☞ Images © A. Conneau & G. Lample (2018)

Towards multilingual –language agnostic – representations

Computing embeddings such that mutual translations are nearest neighbours

X-lingual embeddings boost MT and yield unsupervised alignments

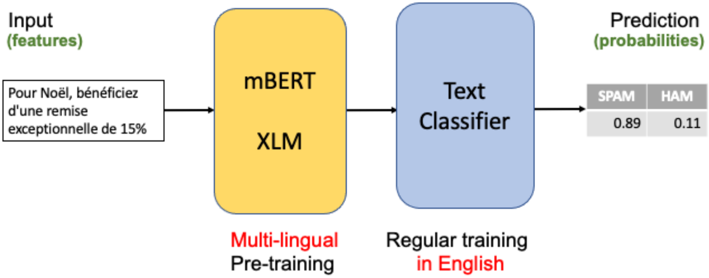


☞ Images © A. Conneau (2019)

Towards multilingual –language agnostic – representations

Computing embeddings such that mutual translations are nearest neighbours

“Zero-shot” X-lingual transfer



- Pretrain with multilingual representations
- Train with annotated texts in English or high-resource languages
- Make predictions for texts in low-resource languages

Towards multilingual –language agnostic – representations

Computing embeddings such that mutual translations are nearest neighbours

Variants:

- BART & mBART: multi-lingual (mono-lingual translation with denoising Transformer)
- smaller models with distillation
- model specialization / adaptation
- etc etc

The many blessings of Transformers

- Powerful extractors for generic features from varied (eg. multilingual, multimodal) contexts
- Large scale pre-training with word prediction tasks
- Seamless integration with powerful DNN predictors for arbitrary tasks
- Convergence in NLP, speed-up flow of innovations across sub-communities

Transformers = the end of NLP?

- Opacity, brittleness, randomness of decisions
- How about structural information on word, sentence, discourse?
- Transformers are huge textual memories
- Embedded biases and potential to harm Bender et al. [2021]
- Data hungryness, computational cost of training [Strubell et al., 2019]

Bibliography I

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Proceedings of the first International Conference on Learning Representations, San Diego, CA, 2015.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? . In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137–1155, 2003. ISSN 1532-4435.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pages 103–111, Doha, Qatar, October 2014a. URL http://www.aclweb.org/anthology/W14-4012.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, 2014b. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D14-1179.

Bibliography II

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. Journal of Machine Learning Research, 12(Aug): 2493–2537, 2011.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N19-1423.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Computation, 9(8): 1735–1780, 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. CoRR, abs/1901.07291, 2019. URL http://arxiv.org/abs/1901.07291.

Hai Son Le, Alexandre Allauzen, Guillaume Wisniewski, and François Yvon. Training continuous space language models: Some practical issues. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 778–788, Cambridge, MA, 2010. URL http://www.aclweb.org/anthology/D/D10/D10-1076.

Hai-Son Le, Alexandre Allauzen, and François Yvon. Continuous space translation models with neural networks. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 39–48, Montréal, Canada, 2012.

Bibliography III

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. Structured output layer neural network language models for speech recognition. Audio, Speech, and Language Processing, IEEE Transactions on, 21(1):197 –206, 2013. ISSN 1558-7916. doi: 10.1109/TASL.2012.2215599.

Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In Proceedings of the International Conference on Learning Representations, ICLR, Vancouver, BC, Canada, 2018. URL https://openreview.net/forum?id=ByJHuTgA-.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing LSTM language models. In Proceedings of the International Conference on Learning Representations, ICLR, 2018. URL https://openreview.net/pdf?id=SyyGPP0TZ.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In Proceedings of the 11th Annual Conference of the International Speech Communication Association (Interspeech 2010), pages 1045–1048. International Speech Communication Association, 2010.

Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modeling. In Proc. ICML '07, pages 641–648, New York, NY, USA, 2007.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In Proceedings of the 29th International Conference on Machine Learning, pages 1751–1758, 2012.

Bibliography IV

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana, 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N18-1202.

Martin Popel, Marketa Tomkova, Jakub Tomek, Łukasz Kaiser, Jakob Uszkoreit, Ondřej Bojar, and Zdeněk Žabokrtský. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. Nature Communications, 11(1):4381, 2020. doi: 10.1038/s41467-020-18073-9. URL https://doi.org/10.1038/s41467-020-18073-9.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://openai.com/blog/better-language-models/.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Holger Schwenk. Continuous space language models. Computer, Speech and Language, 21(3):492–518, 2007. ISSN 0885-2308.

Bibliography V

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany, August 2016. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1355. URL https://aclanthology.org/P19-1355.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient Transformers: A survey, 2020. URL http://arxiv.org/pdf/2009.06732.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.