Deep Learning for Natural Language Processing (NLP)

Frédéric Armetta

MCF, LIRIS laboratory, University of Lyon1
frederic.armetta@univ-lyon1.fr

October 8, 2024

Overview

What is NLP ?

"Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data."(wikipedia'21)

Applications of NLP

- Classification
  Sentiment Analysis, Text Classifications, Spam Detection, etc.

Applications of NLP

- Classification
  Sentiment Analysis, Text Classifications, Spam Detection, etc.
- Generation
  Predictive Typing, Question Answering, Text Summarization, Chatbot, etc.

Applications of NLP

- Classification

  Sentiment Analysis, Text Classifications, Spam Detection, etc.
- Generation

  Predictive Typing, Question Answering, Text Summarization, Chatbot, etc.
- Tagging

  Part-of-speech tagging, Spell Checking, etc.

Applications of NLP

- Classification
  Sentiment Analysis, Text Classifications, Spam Detection, etc.
- Generation
  Predictive Typing, Question Answering, Text Summarization, Chatbot, etc.
- Tagging
  Part-of-speech tagging, Spell Checking, etc.
- ... but also, extended applications
  Speech Recognition (sound processing), Character Recognition (image processing), etc.

How to address NLP problems ?

- Symbolic AI
  rule-based-approaches : grammar, vocabulary, etc.
  - + : explicit, determinist
  - - : inability to render complexity (semantics is complex !), specific knowledge to provide for each application field

How to address NLP problems ?

- Symbolic AI
  rule-based-approaches : grammar, vocabulary, etc.
  - + : explicit, determinist
  - - : inability to render complexity (semantics is complex !), specific knowledge to provide for each application field
- Deep Learning
  (the topic for this presentation!)
  - + : generalization, self-acquired knowledge
  - - : availability of datasets, black boxes, explainability

How to address NLP problems ?

- Symbolic AI
  rule-based-approaches : grammar, vocabulary, etc.
  - + : explicit, determinist
  - - : inability to render complexity (semantics is complex !), specific knowledge to provide for each application field
- Deep Learning
  (the topic for this presentation!)
  - + : generalization, self-acquired knowledge
  - - : availability of datasets, black boxes, explainability
- ... but also Hybrid Approaches
  to make benefit from the two ways as far as possible or not ...

Datasets and transfert learning

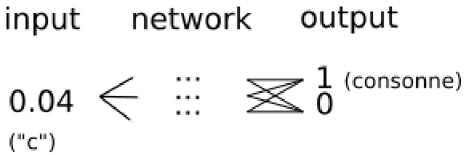
How to benefit from a general purpose dataset for a specific task ?

- (first) Self-supervised learning
  Huge datasets to educate the network
  - wikipedia, large corpus
  - big networks, heavy time and energy consumption
- (second) Fine-tuning
  Specialize to a specific task, mainly educate the last layers of the network
  - load a pre-trained general purpose network for the first layers
  - adapt the network (last layers) to the dedicated task
  - modest size dataset
  - restricted learning time
  - greatly enhanced accuracy !

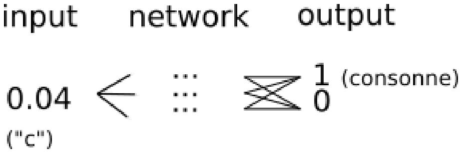
How to feed neural networks with texts ?

"If the network can't efficiently differentiate letters or words, it's a bad start ..."

First idea : letters or words in [0, 1]



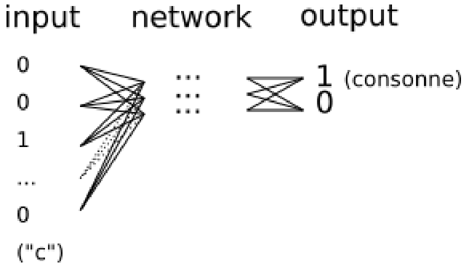
First idea : letters or words in [0, 1]



⇒ Difficult to separate 26 letters, very difficult to separate 5000 words !

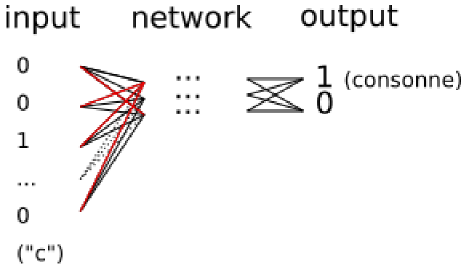
One-hot encoding (letters)

$V = \{a, b \cdots, z\}$, $|V| = 26$
$a = (1, 0, \cdots, 0)$
$b = (0, 1, \cdots, 0)$
$\ldots$
$z = (0, 0, \cdots, 1)$



One-hot encoding (letters)

$V = \{a, b \cdots, z\}$, $|V| = 26$
$a = (1, 0, \cdots, 0)$
$b = (0, 1, \cdots, 0)$
$\cdots$
$z = (0, 0, \cdots, 1)$



$\Rightarrow$ Easier task for the network !

One-hot encoding (words)

$V = \{a, cat \cdots, cats\}$, $|V| = 6$
$"a" = (1, 0, \cdots, 0)$
$"cat" = (0, 1, \cdots, 0)$
$\cdots$
$"cats" = (0, 0, \cdots, 1)$

Word Embedding

- One hot encoding is good for network manipulation but ...
- doesn't hold any semantics !

Word Embedding

- One hot encoding is good for the network manipulation but ...
- doesn't hold any semantics !

Word embedding ("Plongement lexical")

Transform words into k-dimensional vectors

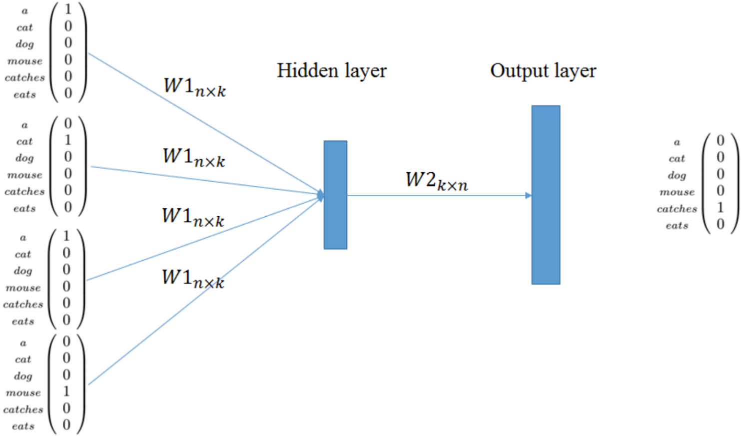
$h_{(k,1)} = W_{(k,n)}.X_{(n,1)}$
$h_i = \sum_{j=1}^{n} W_{ij}.x_j$

How to learn $W_{(k,n)}$?

"A cat catches a mouse"
Word2Vect :

- CBOW : predict a words thanks to words in close proximity
  "a", "cat", "?", "a", "mouse" $\rightarrow$ "catches"
- Skip-gram : predict proximity words thanks to an input word
  "catches" $\rightarrow$ "a", "cat", "a", "mouse"

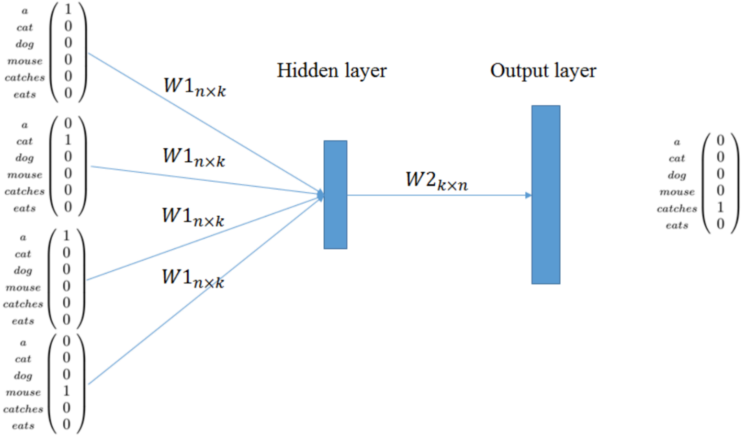
CBOW



word2vec: CBOW model[1]

[1] Symbolic, Distributed and Distributional Representations for Natural Language Processing in the Era of Deep Learning: a Survey, L. Ferrone and F. M. Zanzotto, CoRR, 2017

Skip-gram
general idea : reverse order



outputs ← input
(predict hidden layers for each word of the context)

GloVe

- Similar to word2vect but ...
- additional global statistics inclusion to obtain word vectors (word2vect incorporates only local statistics)
- open-source project at Stanford[2]

[2] GloVe: Global Vectors for Word Representation, J. Pennington and al., Proceedings of EMNLP, 2014, https://aclanthology.org/D14-1162.pdf

GloVe

co-occurrence matrix for the sentence "a cat catches a mouse" with a window size of 1:

	a	cat	catches	mouse
a	0	1	1	1
cat	1	0	1	1
catches	1	1	0	0
mouse	1	0	0	0

GloVe

- Semantic distance between two words thanks to a third one
- homomorphism hypothesis
- and more ...

Table 1: Co-occurrence probabilities for target words *ice* and *steam* with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

context of word i.

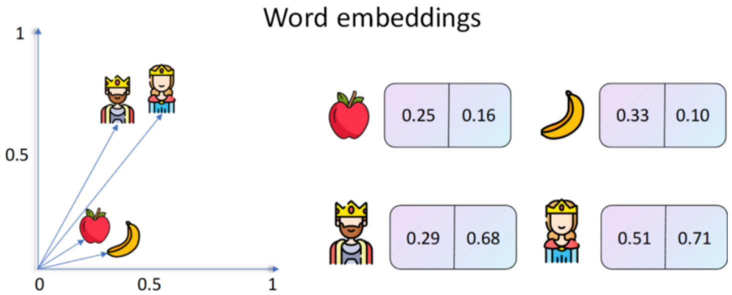
We begin with a simple example that showcases how certain aspects of meaning can be extracted directly from co-occurrence probabilities. Consider two words i and j that exhibit a particular aspect of interest; for concreteness, suppose we are interested in the concept of thermodynamic phase, for which we might take $i = ice$ and $j = steam$. The relationship of these words can be examined by studying the ratio of their co-occurrence prob-

the information present the ratio P_{ik}/P_{jk} in the word vector space. Since vector spaces are inherently linear structures, the most natural way to do this is with vector differences. With this aim, we can restrict our consideration to those functions F that depend only on the difference of the two target words, modifying Eqn. (1) to,

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} . \qquad (2)$$

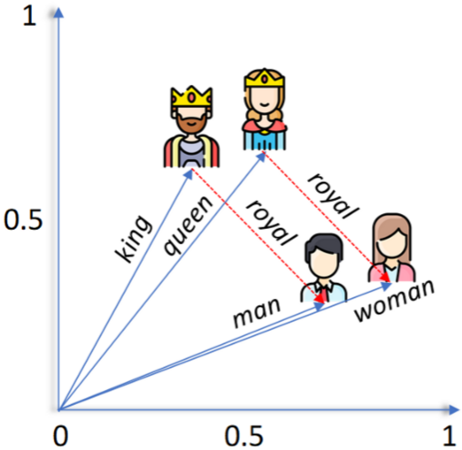
GloVe: Global Vectors for Word Representation, J. Pennington and al., Proceedings of EMNLP, 2014, https://aclanthology.org/D14-1162.pdf

Word Embedding properties



source
https://towardsdatascience.com/deep-learning-for-nlp-word-embeddings-4f5c90bcdab5

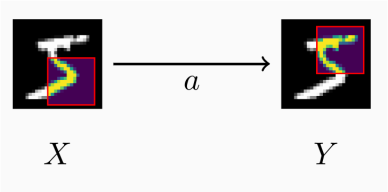
Word Embedding properties



source https://towardsdatascience.com/deep-learning-for-nlp-word-embeddings-4f5c90bcdab5

Why RNN ?

- RNN = Recurrent Neural Network
    - as an encoder: integrate in an internal state (memory) sequences of inputs
    - as a decoder: generate from an internal state (memory) sequences of ouputs



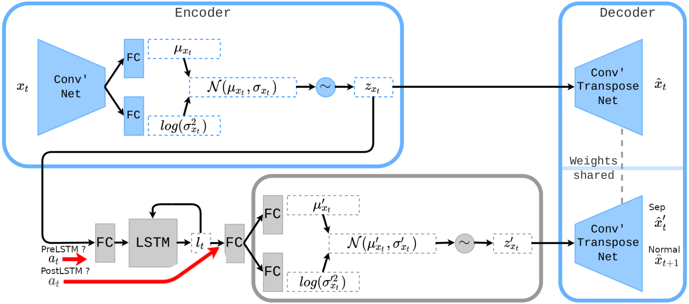


Figure: Accumulate perceptions for learning

The Impact of Action in Visual Representation Learning, Devillers, Chaffraix, Armetta, Duffner, Lefort, ICDL 2022 `https://hal.science/hal-03815546`

Why RNN and NLP ?

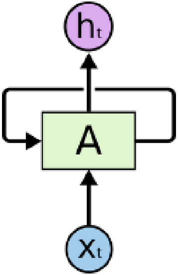
- RNN for NLP
  - as an encoder : build a representation for a sentence/document
  - as an encoder : build an embedding for words depending on the context (see ELMO)
  - as a decoder : text generation (text summarising, machine translation, etc.)

Why to capture the context ?

- the **driver** is outdated
- the **driver** exceeds the speed limit

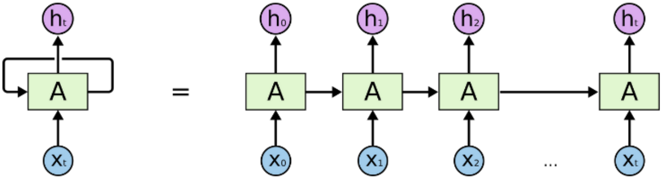
different contexts $\Rightarrow$ different meanings

Recurrent Neural Network (RNN)



source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

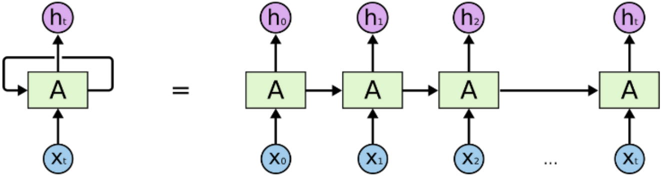
Recurrent Neural Network (RNN)



An unrolled recurrent neural network.

source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Network (RNN)



An unrolled recurrent neural network.

exemple : text completion

---

source

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Network (RNN)

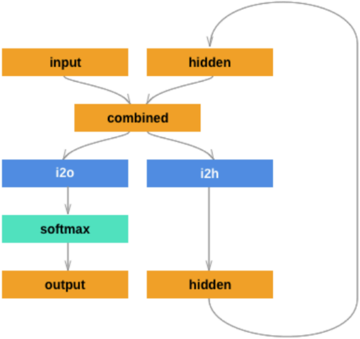
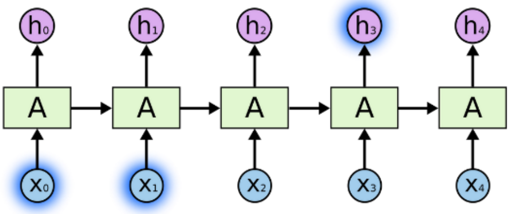


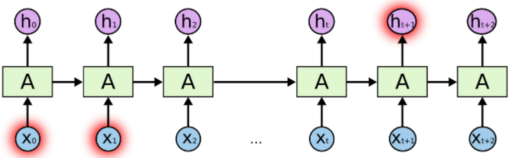
Figure: A simple RNN : `https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial`

Long-Term Dependencies



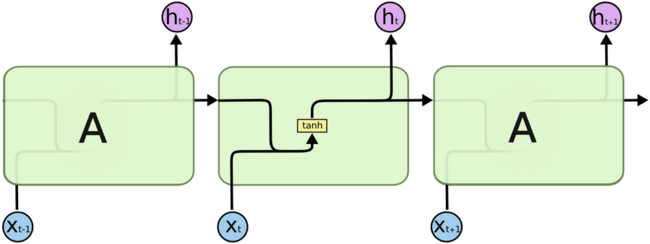
source
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-Term Dependencies



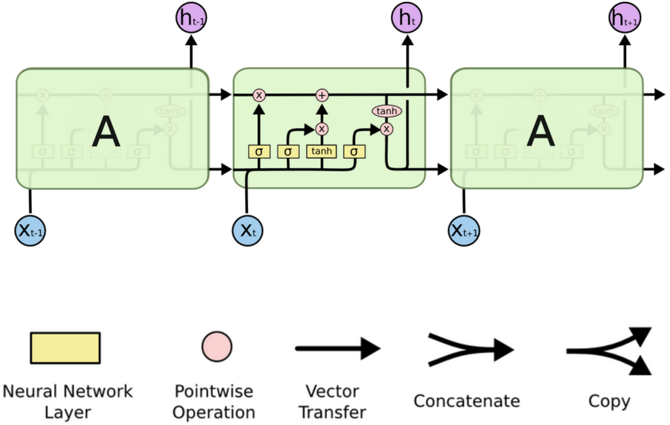
source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long short-term memory (LSTM)



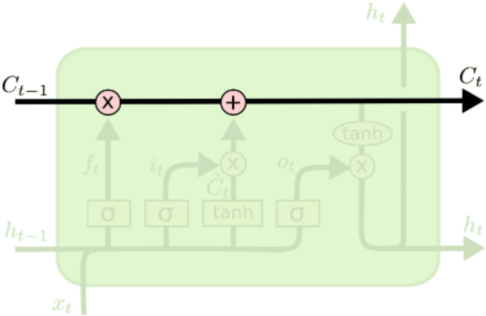
source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long short-term memory (LSTM)



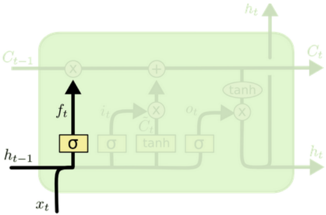
source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long short-term memory (LSTM)



source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

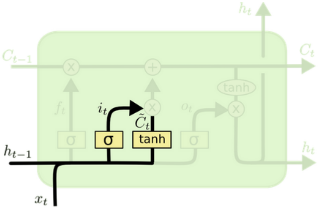
Long short-term memory (LSTM)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

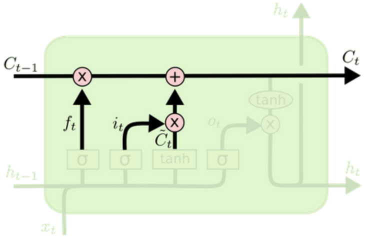
Long short-term memory (LSTM)



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

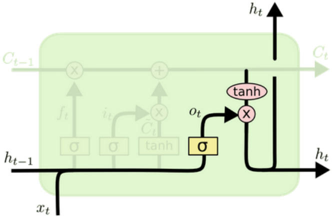
Long short-term memory (LSTM)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long short-term memory (LSTM)

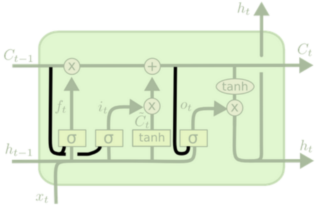


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

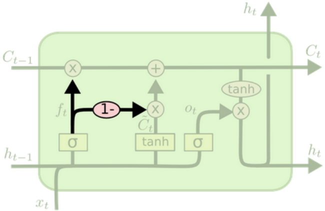
Variant : the gate layers look at the cell state



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

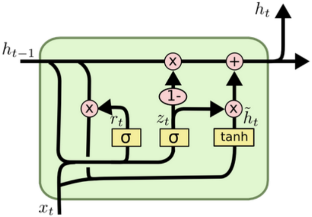
Variant : coupled forget and input gates



$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$

source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure: cell state and hidden state merged

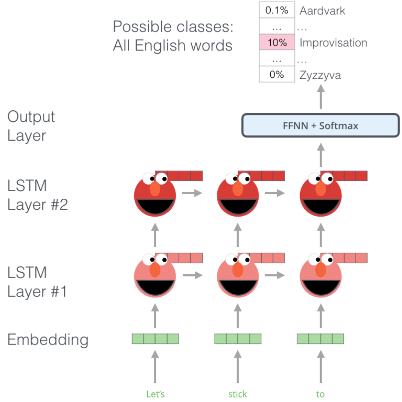
source https://colah.github.io/posts/2015-08-Understanding-LSTMs/

ELMo[2]

- Embeddings from Language Model
- Embeddings are context-sensitive

[2]ME Peters and al., Deep contextualized word representations, 2018, https://arxiv.org/pdf/1802.05365

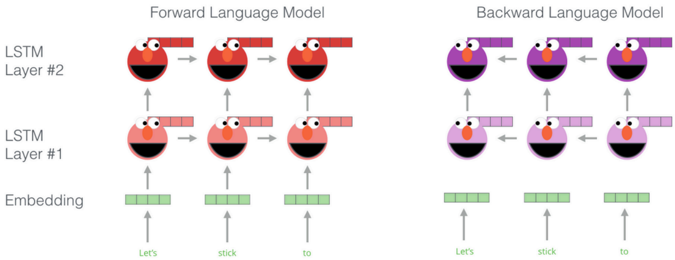
Stacked LSTMs



A step in the pre-training process of ELMo: Given "Let's stick to" as input, predict the next most likely word – a *language modeling* task. When trained on a large dataset, the model starts to pick up on language patterns. It's unlikely it'll accurately guess the next word in this example. More realistically, after a word such as "hang", it will assign a higher probability to a word like "out" (to spell "hang out") than to

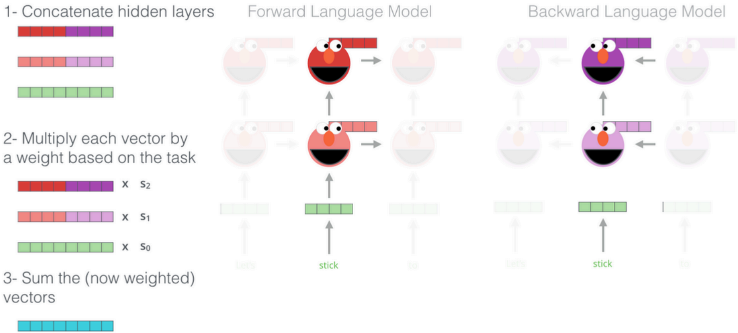
source https://jalammar.github.io/illustrated-bert/

BI-LSTM



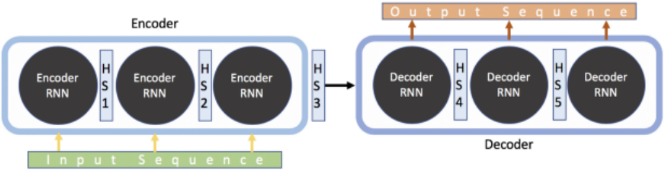
source https://jalammar.github.io/illustrated-bert/

Contextualized embedding



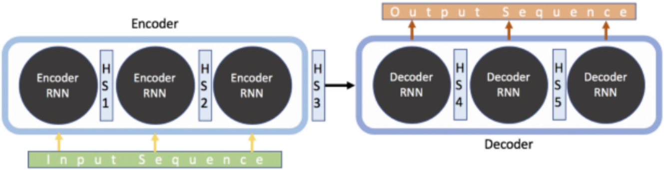
source https://jalammar.github.io/illustrated-bert/

Sequence-2-Sequence Model (Seq2Seq)



source https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263

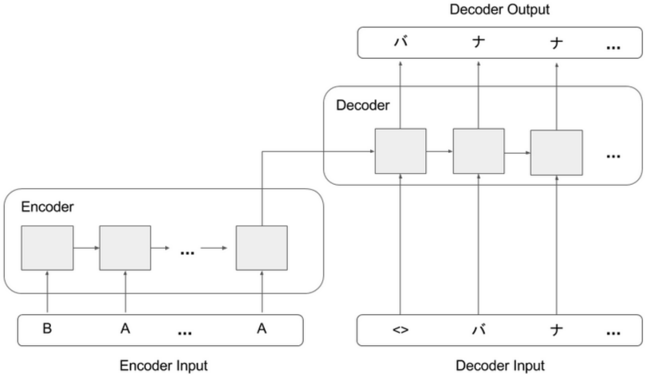
Sequence-2-Sequence Model (Seq2Seq)



exemple : question answering, text traduction, text summarization

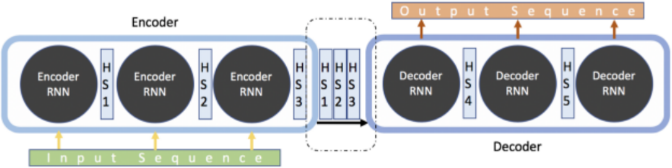
source https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263

Teacher forcing



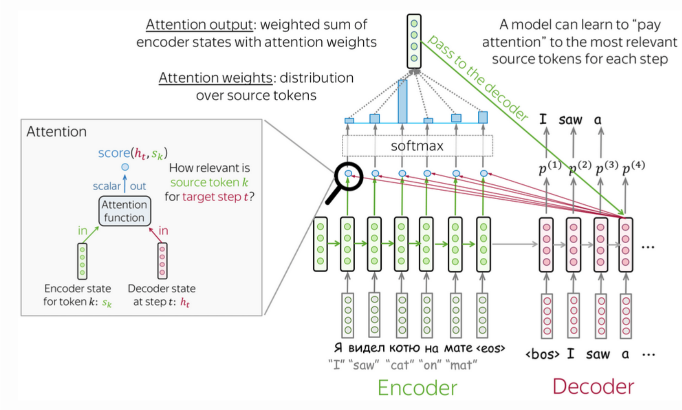
source
https://www.kaggle.com/residentmario/seq-to-seq-rnn-models-attention-teacher-forcing

Attention mechanism



source https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263

Attention mechanism



Attention is all you need[3]

Recurrence no more necessary

[3] Attention Is All You Need, A. Vaswani et al. , NeuIPS proceedings, 2017

Self-attention

Consider the word driver:

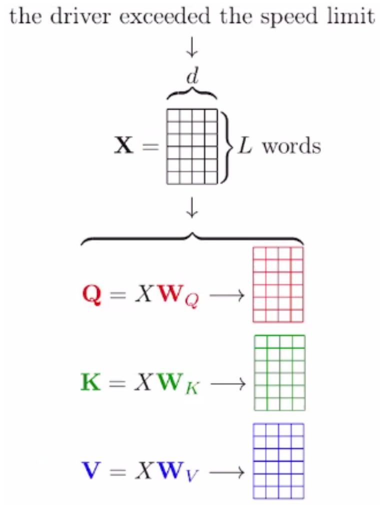
	the	driver	exceeded	the	speed	limit
Embeddings:	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
Attention:	$\lambda_{1,2}$	$\lambda_{2,2}$	$\lambda_{3,2}$	$\lambda_{4,2}$	$\lambda_{5,2}$	$\lambda_{6,2}$

Output: $\mathbf{z}_2 = f\left(\sum_i [\lambda_{i,2} \times g(\mathbf{x}_i)]\right)$

- $(\lambda_{i,j})$ are the attention coefficients, $\sum_i \lambda_{i,j} = 1$, and
- Reflects the influence of $\mathbf{x_i}$ on $\mathbf{x_j}$ (transformed version)

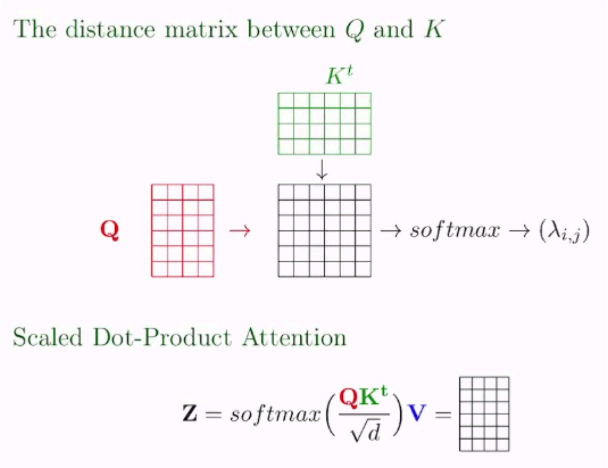
source http://www.ixxi.fr/agenda/seminaires/cycles-de-seminaires/intelligence-artificielle-et-langage/alexandre-allauzen

Queries, Keys, Values



source http://www.ixxi.fr/agenda/seminaires/cycles-de-seminaires/intelligence-artificielle-et-langage/alexandre-allauzen

Transformer



source http://www.ixxi.fr/agenda/seminaires/cycles-de-seminaires/intelligence-artificielle-et-langage/alexandre-allauzen

Multi-head attention

$$\mathbf{X} = \text{[matrix]}$$

$$\downarrow$$

$$\mathbf{W}_Q^{(1)}, \mathbf{W}_K^{(1)}, \mathbf{W}_V^{(1)} \qquad \mathbf{W}_Q^{(2)}, \mathbf{W}_K^{(2)}, \mathbf{W}_V^{(2)}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\mathbf{Q}^{(1)}, \mathbf{K}^{(1)}, \mathbf{V}^{(1)} \qquad \mathbf{Q}^{(2)}, \mathbf{K}^{(2)}, \mathbf{V}^{(2)}$$

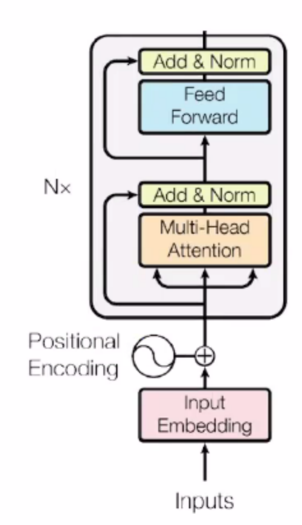
$$\downarrow \qquad\qquad \downarrow$$

$$\mathbf{Z}^{(1)} \qquad\qquad \mathbf{Z}^{(2)}$$

$$\mathbf{Z} = (\mathbf{Z}^{(1)}; \mathbf{Z}^{(2)})\mathbf{W}_o$$

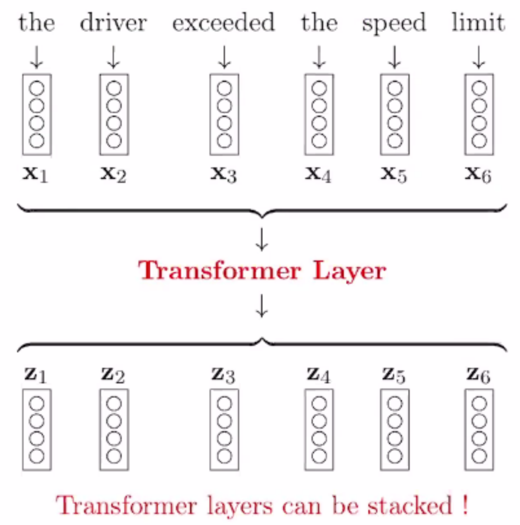
source http://www.ixxi.fr/agenda/seminaires/cycles-de-seminaires/intelligence-artificielle-et-langage/alexandre-allauzen

Transformer



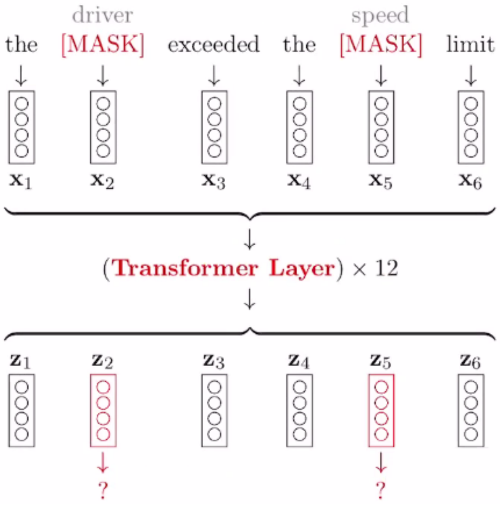
source http://www.ixxi.fr/agenda/seminaires/cycles-de-seminaires/intelligence-artificielle-et-langage/alexandre-allauzen

Transformer



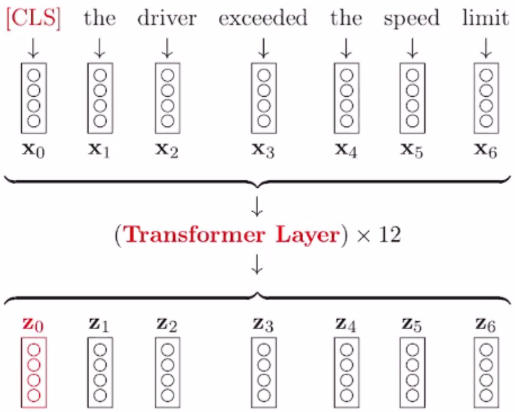
source http://www.ixxi.fr/agenda/seminaires/cycles-de-seminaires/intelligence-artificielle-et-langage/alexandre-allauzen

Learning



source http://www.ixxi.fr/agenda/seminaires/cycles-de-seminaires/intelligence-artificielle-et-langage/alexandre-allauzen

Fine tuning



source http://www.ixxi.fr/agenda/seminaires/cycles-de-seminaires/intelligence-artificielle-et-langage/alexandre-allauzen

My research

Sentiment classification