Sequence processing and language modelling

Alexandre Allauzen

Fall 2023

MILES
Machine Intelligence and Learning Systems

Dauphine | PSL
UNIVERSITÉ PARIS

ESPCI PARIS | PSL

Roadmap

Outline

Sequence classification

my wonderful friend took
me to see this movie
for our anniversary.
it was terrible.

→ ☹ : 0 · · · 1 : ☺

Many examples:

- Properties Detection, content classification for an input text (stance, toxicity, bias, fake news, . . .)
- Paraphrase detection and textual entailment

Output:

A class or a score (regression)

Sequence tagging

Semantic Role Labelling

Assign semantic role to words, *e.g*:

$$y_i \in [\text{ Agent, Patient, Source, Destination, Instrument,} \cdots, \text{ Other }]$$

$\mathbf{x} =$	John	drove	Mary	from	Austin	to	Dallas	in	Peugeot
$\mathbf{y} =$	A	O	P	O	S	O	D	O	I

From words to phrases (with BIO scheme for segmentation)

Output:

A sequence of labels, one for each input token.

Conditionnal generation

QA, Prompt, Summarization, . . .

- Input/Output belong to the same *domain*.
- The prompt is a kind of "prefix"/context of the generation.

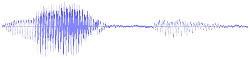
Output:

A text of "arbitrary" length (words, sentences, paragraphs, . . .)

$$\underbrace{\text{tell me more about measure ?}}_{\text{context: } w_1^6}\ \underbrace{\text{A measure is a mapping from ...}}_{\text{generation: } w_7^L}$$

Use the model to generate: $$

Speech recognition / Machine Translation

Input	*Output*
	[Martine, boude]
[il, est, temps]	[es , ist , zeit]
$\mathbf{x}$	$\mathbf{w} = (w_1, w_2, ..., w_I)$

$$P(\mathbf{w}|\mathbf{x}) = \prod_{i}^{I} P(w_i|\mathbf{w}_{<\mathbf{i}}, \mathbf{x})$$

- Evaluate $\mathbf{w}$ in the context $\mathbf{x}$
- Generate $\mathbf{w}$ from $\mathbf{x}$
- Find the best $\mathbf{w}$ given $\mathbf{x}$

Deep-Learning blocks

Encoder:

- Compute a representation of the input
  - can be one vector: extraction, compression, ...
  - can be a new sequence of "annotations" or vectors.
- Extract meaningful information for the downstream task

Decoder (QA, bot, ...)

A generative model for sequence: $P(w_1^L) = \underbrace{P(w_1^K)}_{\text{prompt}}\underbrace{P(w_{K+1}^L|w_1^K)}_{answer}$

Encoder-Decoder (ASR, MT, ...)

$$P(\mathbf{w}|\mathbf{x}) = \prod_i^I P(w_i|\mathbf{w}_{<\mathbf{i}}, \mathbf{x})$$

$$\mathbf{x} \rightarrow \boxed{\text{Encoder}} \rightarrow \boldsymbol{z} \text{ the internal state} \rightarrow \boxed{\text{Decode}} \rightarrow \boldsymbol{w}$$

Outline

Language modelling task

A word prediction game

$$\begin{matrix} \text{time} & \text{goes} & \text{by} & \text{so} \\ w_1 & w_2 & \cdots & w_{i-1} \end{matrix} \longrightarrow w_i = ? \begin{cases} \text{a} \\ \vdots \\ \text{fastly} \\ \vdots \\ \text{slowly} \\ \vdots \end{cases}$$

A probability distribution over words

$$P(w_i | w_1^{i-1}),\ w_i \in \mathcal{V}$$

A probabilistic and generative model

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i | w_1^{i-1}), \quad \forall i, w_i \in \mathcal{V}$$

Challenges

- Large vocabulary (from 10k to millions)
- Very sparse observation
- Large amount of available data but noisy, heterogenous, ...

Count based model (from 80's to 2000)

Count (or n-gram) based model

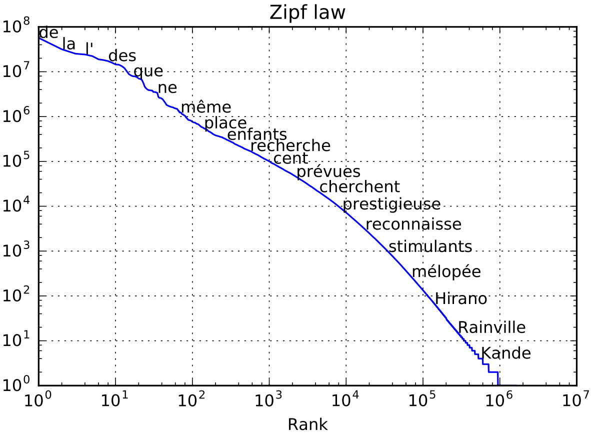
Under a markovian assumption:

$$P(w_i|w_1^{i-1}) \approx P(w_i|\underbrace{w_{i-n+1}^{i-1}}_{\substack{n-1 \\ \text{last words}}}) = \frac{c(w_i|w_1^{i-n+1})}{c(w_1^{i-n+1})}$$

Lack of generalization

- smoothing methods as a workaround
- but no similarity between words

The Zipf law



Zipf law
de
la
l'
des
que
ne
même
place
enfants
recherche
cent
prévues
cherchent
prestigieuse
reconnaisse
stimulants
mélopée
Hirano
Rainville
Kande
10^8
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
Rank

A second life as an unsupervised learning task

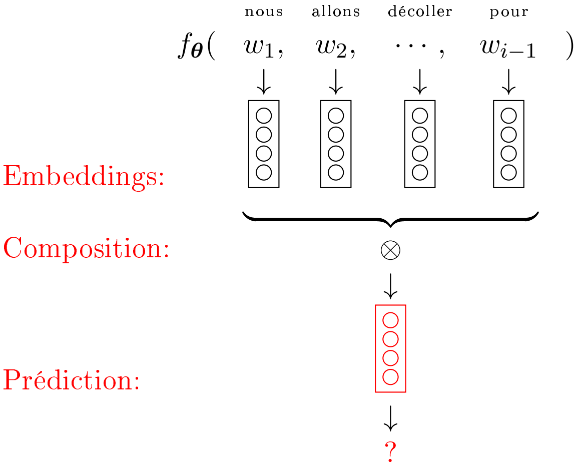
"Language Models are Unsupervised Multitask Learners"

- Leveraging the huge amount of unlabeled texts
- To pre-train word representations
- Along with their contextualization at the sentence level
- That can be fine-tuned for downstream tasks

A profusion of architectures

- Starting with convolution networks [3]
- More recently ELMo and ULMFit with LSTM [7, 6]
- BERT and GPT with Transformers [4, 8]

Neural Language Model



First step: word embeddings [1]

Learning

Leverage all the data you can access !

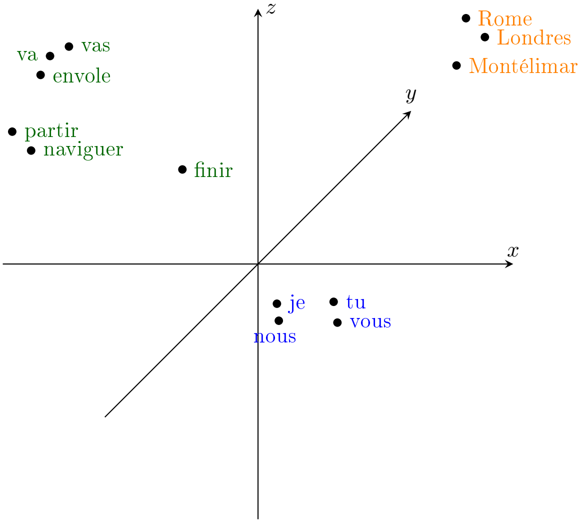
...	...	...	...	...
tu	vas	partir	pour	Montélimar
je	m'	envole	pour	Londres
ils	vont	finir	à	Montluçon
vous	préférez	naviguer	vers	Brest
...	...	...	...	...

Prédiction

nous décollons pour $\rightarrow$?

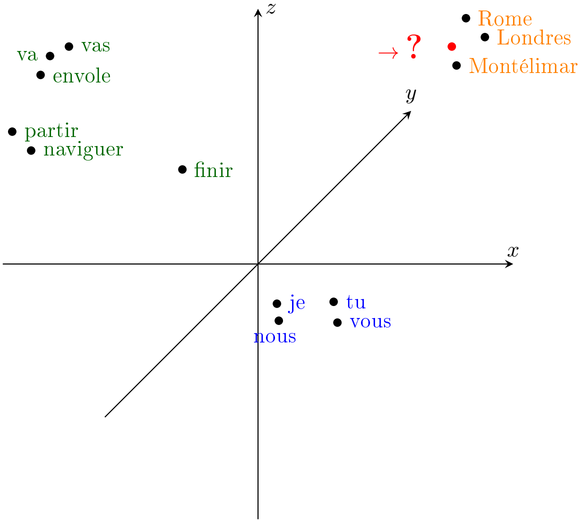
Intuition

nous décollons pour → ?



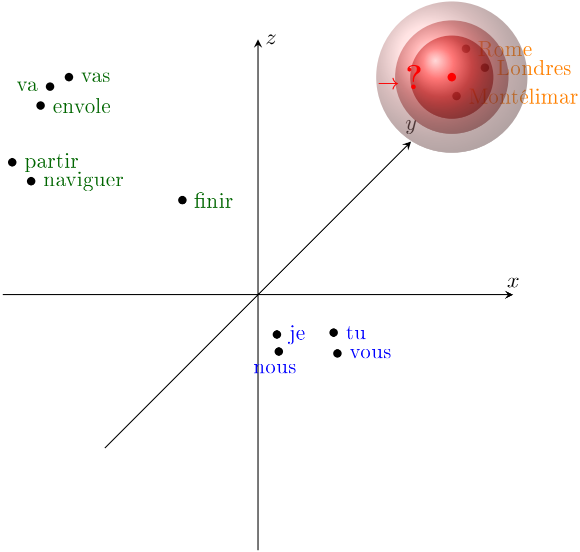
Intuition

nous décollons pour → ?



Intuition

nous décollons pour → ?



Architecture for Neural Language Model

The goal

$$P(\mathbf{w}) = \prod_{i}^{I} P(w_i|\mathbf{w_{<i}})$$

Oldies but goodies

- n-gram (with NNet):

$$P(w_i|w_{i-n+1}^{i-1}) = f_{\boldsymbol{\theta}}(w_{i-n+1}^{i-1})$$

- recurrent network:

$$P(w_i|w_1^{i-1}) = f_{\boldsymbol{\theta}}(w_1^{i-1})$$

Transformers

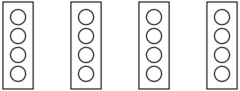
Outline

A dynamical model for sequence - 1

The object understudy

A word sequence or its embedded version

$$
\begin{array}{rccccc}
 & \text{time} & \text{goes} & \text{by} & \text{so} & \cdots \\
\mathbf{w} = & w_1, & w_2, & w_3, & w_4, & \cdots \\
 & \downarrow & \downarrow & \downarrow & \downarrow &
\end{array}
$$



$$\text{Embeddings: } \mathbf{X} = \quad \mathbf{x}_1, \quad \mathbf{x}_2, \quad \mathbf{x}_3, \quad \mathbf{x}_4, \quad \cdots$$

Assumption

This sequence is generated by a discrete dynamical system

- at each time step: a new word is observed
- update the memory or hidden state
- generate the next word given the hidden state

A dynamical model for sequence - 2

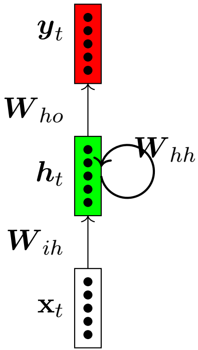
Definition

$$\begin{cases} \mathbf{h}_{t+1} &= f_{\boldsymbol{\theta}}(\overbrace{\mathbf{x}_t}^{\text{observation}}, \underbrace{\mathbf{h}_t}_{\text{recurrence}}), \quad \text{memory} \\ \mathbf{x}_{t+1} &= g_{\boldsymbol{\Phi}}(\mathbf{h}_{t+1}), \quad \text{generation} \end{cases}$$

At each time step:

$$\mathbf{x}_t \rightarrow \mathbf{h}_{t+1} \rightarrow \mathbf{x}_{t+1}$$

The Elman Cell for LM [5]



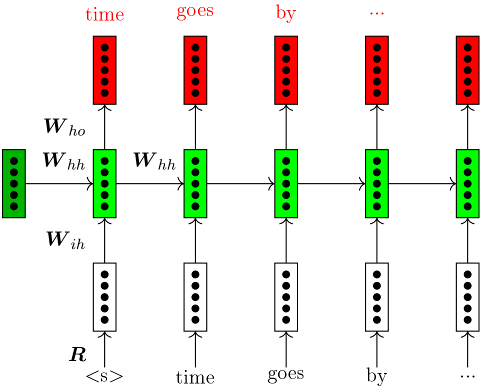
A dynamic system, at time t:

- maintains a hidden vector (the memory): $\boldsymbol{h}_t$
- Updated with the observation of $\mathbf{x}_t$ and $\boldsymbol{h}_{t-1}$
- The (optional) prediction $\boldsymbol{y}_t$ depends on the internal state ($\boldsymbol{h}_t$)
- For a language model, $\mathbf{x}_t$ comes from word embeddings

The parameters are shared !

(Vanilla) Recurrent network language model

Unfolding the structure: a deep-network



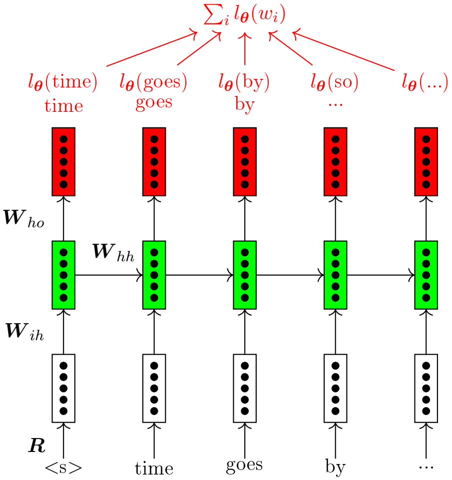
At each step t:

$$\boldsymbol{h}_t = f(\boldsymbol{W}_{ih}\mathbf{x}_t + \boldsymbol{W}_{\boldsymbol{hh}}\boldsymbol{h}_{\boldsymbol{t-1}})$$

$$\boldsymbol{y}_t = g(\boldsymbol{W}_{ho}\boldsymbol{h}_t)$$

g is the softmax function over the vocabulary

Training recurrent language model



Back-Propagation through time or BPTT [9]

Issues with Elman Cell

Gradient vanishing / exploding

- The unfolded network is (very) deep
- The architecture is difficult to train

Long range dependencies

- Difficult to infer long range dependencies
- Unstable dynamical system difficult to control
- No memory managment

Outline

Motivations

- Address the gradient propagation issue
- Allow the model to skip/keep information through time

Starting point

$$\mathbf{h}_{t+1} = f_{\boldsymbol{\theta}}(\overbrace{\mathbf{x}_t}^{\text{observation}}, \underbrace{\mathbf{h}_t}_{\text{recurrence}})$$

- This function is to simple
- Same for output prediction

From recurrent network to LSTM/GRU: the gate

$$\mathbf{h}_t = f(\mathbf{W}_{ih}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1})$$

- What if we want to mitigate the impact of $\mathbf{h}_{t-1}$?
- To reset (softly) the memory for some components

A Gate is a filter

$$\underbrace{\begin{pmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdots \end{pmatrix}}_{\mathbf{h}_{t-1}} \rightarrow \underbrace{\begin{array}{l} \times 1 \\ \times 0 \\ \times 0.5 \\ \times 0.14397 \\ \times 0.88972 \\ \times \cdots \end{array}}_{\text{filter values:}\boldsymbol{r}} \Leftrightarrow \mathbf{h}_{t-1} * \boldsymbol{r}$$

The values of $\boldsymbol{r}$ can be infered as a function of $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$.

Implementation of a gate as a NNet

$$\boldsymbol{r}_t = \sigma(\mathbf{W}_{ir}\mathbf{x}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1})$$

- A simple Linear layer
- Followed by a sigmoid
- Each output component is between 0 and 1
- A soft learnable gate

Gated Recurrent unit (GRU)[2]

The updated hidden state :

$$\mathbf{h}_t = (1 - \mathbf{z}_t) * \mathbf{h}_{t-1} + \mathbf{z}_t * \hat{\mathbf{h}}_t$$

The candidate $\hat{\mathbf{h}}_t$:

$$\hat{\mathbf{h}}_t = \phi_h(\mathbf{W}_{ih}\mathbf{x}_t + \mathbf{W}_{hh}(\boldsymbol{r}_t * \mathbf{h}_{t-1}))$$

The gates:

$$\mathbf{z}_t = \sigma_g(\mathbf{W}_{iz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1})$$
$$\boldsymbol{r}_t = \sigma_g(\mathbf{W}_{ir}\mathbf{x}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1})$$

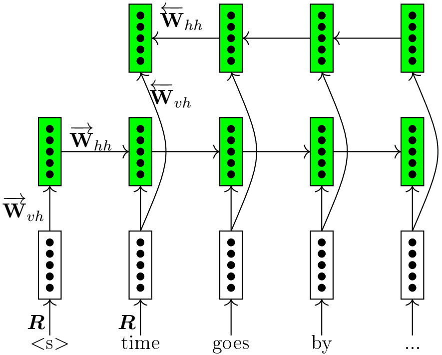
LSTM : Control flow - in one slide

$$\text{Input: } \underbrace{[\mathbf{h}_{t-1}; \mathbf{x}_t]}_{z_t} \longrightarrow \begin{cases} \tilde{C}_t &= \tanh(\mathbf{W}_C z_t + \mathbf{b}_C), \quad \text{basic update} \\ i_t &= \sigma(\mathbf{W}_i z_t + \mathbf{b}_i), \quad \boxed{\text{input gate}} \\ f_t &= \sigma(\mathbf{W}_f z_t + \mathbf{b}_f), \quad \boxed{\text{forget gate}} \\ o_t &= \sigma(\mathbf{W}_o z_t + \mathbf{b}_o), \quad \boxed{\text{output gate}} \end{cases}$$

$$\text{Output: } \mathbf{C}_t = \underbrace{\boxed{f_t} * \boxed{\mathbf{C}_{t-1}}}_{\text{previous state}} + \underbrace{\boxed{i_t} * \boxed{\tilde{\mathbf{C}}_t}}_{\textit{recurrence}}$$

$$\mathbf{h}_t = \boxed{o_t} * \boxed{\tanh(\mathbf{C}_t)}$$

Sentence encoder : the bi-recurrent solution



A each step t, from left to right

- $w_t \rightarrow \mathbf{x}_t$
- $\overrightarrow{\mathbf{h}}_t = f(\overrightarrow{\mathbf{W}}_{vh}\mathbf{x}_t + \overrightarrow{\mathbf{W}}_{hh}\overrightarrow{\mathbf{h}}_{t-1})$

And from right to left

- $w_t \rightarrow \mathbf{x}_t$
- $\overleftarrow{\mathbf{h}}_t = f(\overleftarrow{\mathbf{W}}_{vh}\mathbf{x}_t + \overleftarrow{\mathbf{W}}_{hh}\overleftarrow{\mathbf{h}}_{t-1})$

$[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$: contextualized representation of w_t

Conclusion on recurrent architecture

A powerful architecture for sequence

- Useful for classification
- Sequence tagging and language model
- Encoder / Decoder architecture
- And works with attention (of course)

Some limitations

- Auto-regressive inference (encoder or decoder)
- Issues with long-term memories