Deep learning e generazione di testo: un approccio all’uso di caratteri al posto delle parole

Giovanni Bonetta, Rossella Cancelliere, Marco Roberti – Dipartimento di Informatica, Torino
Patrick Gallinari – Laboratoire d’Informatique de Paris 6, Paris
The term Deep Learning was introduced to the machine learning community in 1986 (!), but as far back as 1971 a paper by Ivakhnenko described a deep network with 8 layers.

In 1989 Yann LeCun et al. applied the backprop algorithm to a DNN recognizing handwritten ZIP codes on mail. Training required 3 days...

In 1995 Frey demonstrated that it was possible to train (over two days) a network containing six fully connected layers and several hundred hidden units.

Key difficulties were gradient vanishing, lack of training data and limited computing power.

DL is popularized since 2006, first by some academic actors (Hinton et al., “restricted Boltzmann machine”) and then by big players as GAFA (Google, Apple, Facebook, Amazon), BAT (Baidu, Tencent, Alibaba), ..., that shaped the data world.
In 2009, **Google Brain** used Nvidia GPUs to train DNNs. While there, Ng determined that GPUs could increase the speed of learning by about 100 times.

In October 2012, Krizhevsky et al. won the large-scale **ImageNet competition** by a significant margin over shallow NN: some researchers assess this victory constitutes the start of a "deep learning revolution" that has transformed the AI industry.

It definitely changed the way one will exploit data:

- key players have made available platforms (e.g. TensorFlow, PyTorch, ...), that allow the development in a « short time » of complex processing chains and make complex DL methods available for a large community.

- This shift will most probably influence other scientific domains as well in a near future.
Deep learning is now part of state-of-art systems in various disciplines, as image processing and automatic speech recognition, but not only:

Segmentation + classification, Mask R-CNN, (He 2017)

Image Captioning
(Vinyals 2015)
Deep learning is now part of state-of-art systems in various disciplines, as image processing and automatic speech recognition, but not only:

**Games**


Task: data-to-text generation

Image

Signal

Text

Film

<<this is a long sequence of letters, maybe an article, book, report, JSON obj, SQL-like table...>>

..Caption
Description
Summary...
An example from our dataset

name[Ballato's],
eatType[coffee shop],
customer rating[high],
near[Pasticcio]

For a coffee shop highly-rated by customers, head to Ballato's near Pasticcio.

Located near Pasticcio, Ballato's is a high-rated coffee shop.

The Ballato's is a coffee shop near Pasticcio. It has a high customer rating.
name[X-name],
eatType[X-eatType],
customer rating[high],
near[X-near]

For a X-eatType highly-rated by customers, head to X-name near X-near.

Located near X-near, X-name is a high-rated X-eatType.

The X-name is a X-eatType near X-near. It has a high customer rating.
Our approach: let’s use characters!

**Pros:**
- Intrinsically more general
- Smaller vocabulary
- No delexicalization
- “Natural” tokenization

**Cons:**
- Longer sequences
- Semantic is less clear
- Difficult to align (repetitions)
Our approach: let’s use characters!

name[Ballato's],
eatType[coffee shop],
customer rating[high],
near[Pasticcio]

⚠️ The model can now process the data “as-is”

For a coffee shop highly-rated by customers, head to Ballato's near Pasticcio.

Located near Pasticcio, Ballato's is a high-rated coffee shop.

The Ballato's is a coffee shop near Pasticcio. It has a high customer rating.
Encoder-decoder architecture

Decoder

Summary vector \( h_n \)

\(<\text{SoS}>\)  Sei  libero  domani?  <\text{EoS}>\n
Encoder

\( h_1 \)  \( h_2 \)  \( h_3 \)  \( h_4 \)

Are  you  free  tomorrow?
Encoder-decoder architecture

Encoder

Decoder

The fixed size Summary Vector has to keep all the semantic information!
Attention mechanism

Encoder

Decoder

\[ C_i = \sum_{j=1}^{T_x} a_{ij} h_j \]

\[ a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \]

\[ e_{ij} = \text{att}(s_{i-1}, h_j) \]

sei libero domani? <EoS>
The idea is to use a soft switch, $P_{gen}$, that learns to copy or generate the proper token.

Copy = use the attention distribution.

Generate = use the standard Rnn Cell output.

$$P_{final}(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i: w_i = w} a_i$$
Switching GRUs

\[ I \xrightarrow{\text{Encoder GRU1}} \text{Decoder GRU2} \xrightarrow{} \hat{O} \quad \text{Loss}(O, \hat{O}) \]

\[ \hat{O} \xrightarrow{\text{Encoder GRU2}} \text{Decoder GRU1} \xrightarrow{} \hat{I} \quad \text{Loss}(I, \hat{I}) \]
Results

**Input:** name[Caffe Vivaldi],
eatType[pub], food[Puerto Rican], <...>

**Output:** Caffe Vivaldi is a puerto Rican pub <...>

Copy susceptible slot:
- very peaked attention
- $p_{\text{gen}}$ “jumps” from $\sim 0$ to $\sim 1$
Input: <...> customer rating[low], <...>

Output: <...> with a low customer rating <...>

Non-copy susceptible slot:
- less peaked attention
- only the first character is copied
Input: name[Birghiotto], food[Nerd], near[University]

Output: Birghiotto is a Nerd restaurant near University.

You can literally write everything!
Thanks!

Giovanni Bonetta
Dipartimento di Informatica
Università degli Studi di Torino

Patrick Gallinari
Laboratoire d’Informatique de Paris 6
Sorbonne Université

Rossella Cancelliere
Dipartimento di Informatica
Università degli Studi di Torino

Marco Roberti
Dipartimento di Informatica
Università degli Studi di Torino