Transformers in Natural Language Processing

François Yvon(✉)

Université Paris-Saclay, CNRS, LISN, rue John von Neuman, 91 403 Orsay, France
francois.yvon@limsi.fr

Abstract. This chapter presents an overview of the state-of-the-art in natural language processing, exploring one specific computational architecture, the Transformer model, which plays a central role in a wide range of applications. This architecture condenses many advances in neural learning methods and can be exploited in many ways: to learn representations for linguistic entities; to generate coherent utterances and answer questions; to perform utterance transformations, a major application being machine translation. These different facets of the architecture will be successively presented, which will also allow us to discuss its limitations.

Keywords: Natural language processing · Machine learning · Language models · Neural machine translation

1 Introduction

Language technologies are prominent among the applications of Artificial Intelligence (AI) and are now reaching the general public. They are essential for an effective access to textual information available on the Web or in large document databases; they enable for new forms of interaction with the machine, either by voice or by means of writing aids; they help to communicate with other humans, for example through machine translation systems; in a more underground way, these algorithms structure, organize, filter, select, transform and make possible the management of the myriads of texts and audio recordings that circulate continuously on the Web or on social networks.

These technologies are gradually becoming more efficient for ever-increasing and varied uses. Their progress is the result of a combination of several factors: on the one hand, the development of sophisticated machine learning algorithms capable of taking advantage of the high performance computing devices; on the other hand, the possibility to access vast amounts of textual data, whether annotated or not, to feed the training process. Among the algorithms for text processing, neural algorithms and, in particular, the **Transformer** architecture are nowadays at the forefront. Transformers have become central to carry out three types of computations that, until then, required dedicated architectures: first, text mining and information retrieval algorithms, which benefit from the richness of the **internal representations** calculated by this model; second,

M. Chetouani et al. (Eds.): ACAI 2021, LNAI 13500, pp. 81–105, 2023.
https://doi.org/10.1007/978-3-031-24349-3_6

linguistic analysis algorithms, which can take advantage of the Transformers' ability to integrate and model very long-distance dependencies; finally, **text generation algorithms**, which use this model primarily for their predictive ability. If we add that this same architecture is also suitable for the processing of oral or even multimodal data, and that it allows efficient calculations on a very large scale, we can better understand why this model has become the modern workhorse of computational linguists.

This chapter proposes a gentle introduction to the Transformer architecture, adopting an historical perspective, so as to highlight how this model inherits from and extends previous machine learning approaches to Natural Language Processing. We start in Sect. 2 with an introduction to discrete statistical language models, before moving on to feed forward and recurrent neural architectures, enabling us to introduce important concepts such as lexical embeddings and attention. Section 3 then presents the Transformer architecture in details, and showcases its main applications: representation extraction on the one hand, language generation on the other hand. The last section is devoted to multilingual extensions of this model, which will make its genericity and wide applicability more obvious. We conclude in Sect. 5 by introducing the reader to the main limitations of this models and motivate some directions for future research. After studying this chapter, the reader should be in a position to understand why this architecture has been so successful for language modeling, and get a better grasp at the multiple extensions and developments that are happening in other domains such as audio processing or computer vision.

2 Writing Machines: Language Models

2.1 The Simplest Model

Let us consider starting a basic task of language processing: spam filtering. Its probabilistic treatment involves three steps:

1. the collection of a representative set of emails, containing a set D_{ok} of acceptable emails (hams) and a set D_{ko} of unwanted emails (spams);
2. the construction of a numerical representation for texts. A very simple representation encodes each email d as a large binary vector $\mathbf{h}$ in $\{0,1\}^{|V|}$, with V a predefined vocabulary. For each component, $h_w = 1$ if word w appears in the email, 0 otherwise. These representations (so-called "bag-of-words") are inherently sparse, since most of the components of this vector are null;
3. learning a probabilistic model $P(\text{OK}|d) \propto \exp \sum_w \theta_w h_w$,[1] which evaluates the likelihood that a mail is acceptable. The parameter vector $\boldsymbol{\theta}$ weights the contribution of each individual word to the final decision. The estimation of

[1] The notation $P(u|x) \propto \exp f(\theta, x)$ means that the conditional probability of event u given x is *proportional to* the logit $\exp f(\theta, x)$. $P(u|x)$ is obtained by normalizing this term, that is dividing by the summation over all possible outcomes $\sum_{u'} f(\theta, x)$.

$\boldsymbol{\theta}$ is realized by maximizing the log-likelihood of the training data, according to:

$$\ell(\boldsymbol{\theta}) = \sum_{d \in D_{\text{ok}}} \log P(\text{OK}|d) + \sum_{d \in D_{\text{ko}}} \log(1 - P(\text{OK}|d)).$$

This "historical" model is ubiquitous in modern natural language processing: multi-class routing and classification of documents, "sentiment" or opinion analysis (classes correspond to the polarity of the text), textual entailment, aimed at deciding whether a sentence logically implies another sentence, etc. It already highlights three essential concepts of the statistical approach that has become **dominant** since the 1990s to address NLP problems: (a) the computation of numerical representations (here binary representations) to encode linguistic entities and their properties; (b) the use of these representations in probabilistic models evaluating discrete decisions (here: to classify an email in one of the two possible classes); (c) the estimation of model parameters using annotated data (here: correct and incorrect emails). As we will see, the most recent developments in the field continue to rely on these concepts, using neural networks to learn incomparably more sophisticated representations and models than the one outlined above.

2.2 Word Order

Filtering emails is a simple task: useful representations for this task can disregard the order of word, and more broadly, the structure of the document. However, these representations ignore one of the essential properties of texts, namely their organization in a linear sequence of units. **Language models** are probabilistic models designed to take into account this sequentiality. We use $\mathbf{w} = w_1 \dots w_T$ to denote a discrete sequence including T units (words) denoted w_t. In a n-gram language model, the probability of this sequence is written as:

$$\begin{aligned} P(w_1 \dots w_T) &= \textstyle\prod_{t=1}^{T} P(w_t | w_1 \dots w_{t-1}) \\ &= \textstyle\prod_{t=1}^{T} P(w_t | w_{t-n+1} \dots w_{t-1}). \end{aligned} \tag{1}$$

The first line breaks down the probability of the sequence as a product of conditional distributions; the second makes this decomposition tractable by assuming *locality of dependencies* within the sequence. Formally, this means that the probability of occurrence of unit w_t is independent from the past units, given the context composed of the previous $n-1$ words. The corresponding conditional distributions are discrete probabilities that parameterize the model; the bulk of these parameters will be denoted $\boldsymbol{\theta}$. Assuming that the vocabulary V is finite and known, these parameters are in finite numbers. Conceptually, this model is identical to the previous one: it assigns each "document" (here reduced to the few words preceding the current position) to a "class" (here, one word among all possible words). Effective estimation procedures for the n-gram model are based on counts of occurrences in large corpora, and the resulting parameters estimates take the following form for a trigram model ($n = 2$):

$$\forall u, v, w \in V : P(w|uv) = \frac{n(uvw)}{\sum_{w' \in V} n(uvw')}, \tag{2}$$

where $n(uvw)$ is the number of occurrences of the sequence uvw in a training corpus.

The two basic assumptions of the n-grams model (local dependencies, finite vocabulary) are linguistically naive. On the one hand, there are many examples of dependencies between distant words. These dependencies can be syntactic as in "*the decisions of my branch manager are effective*", where the plural agreement is between "*decisions*" and "*are*", separated by four words; they can be semantic, as in "*the judges of the European Court of Justice have decided...*", where "*decided*" can be predicted as a typical action carried out by judges; or even discursive, thematic or stylistic. There are, on the other hand, multiple arguments that oppose the idea of a finite vocabulary: we return to this issue in Sect. 2.4.

Despite their simplicity, language models are useful for a wide range of applications. First, they can be used as automatic *text generators*: it suffices to repeatedly use Eq. (1), sampling at each step the next word conditioned on the previously generated tokens. Second, these models make it possible *to compare several sequences* in order to select the most plausible one, which often will also be the most grammatically correct one. Such decisions are useful for a spell checker, which must choose the best correction; or for a machine translation system, to select the most correct translation hypothesis, etc. Third, they are useful for *comparing languages*: if a language model is trained with French texts and another one with Italian texts, comparing the probabilities of a sentence for these two models provides a way to decide the most likely language of that text. It can also be used for other types of linguistic sequences: sequences of sounds, letters, or even sequences of utterance to model discourse dependencies.

Initially developped for speech processing applications [38,39], language models have quickly became basic tools for the statistical processing of languages and have given rise to countless developments, notably including improvements in their estimation procedures. Pure count-base estimators (Eq. (2))) are in fact not appropriate to model the probability of very rare events. When using vocabularies of several tens of thousands of units, the vast majority of three word sequences are never observed, and using counts yields zero estimates for most parameters. Smoothing methods aim to improve these estimates, for instance by using word clusters. A review of these developments is in [68]; generalizations of the n-gram based on Markov models or stochastic grammars are in [17]; recent introductions to these techniques can be found in various NLP textbooks [26,41,54].

2.3 Neural Models: Smoothing the Context Space

Feedforward Language Models. The next word prediction task implemented in language models is fundamentally a classification task: [9] propose to

implement it using the *feedforward network*) of Fig. 1 (corresponding to a four-gram model, n = 3).

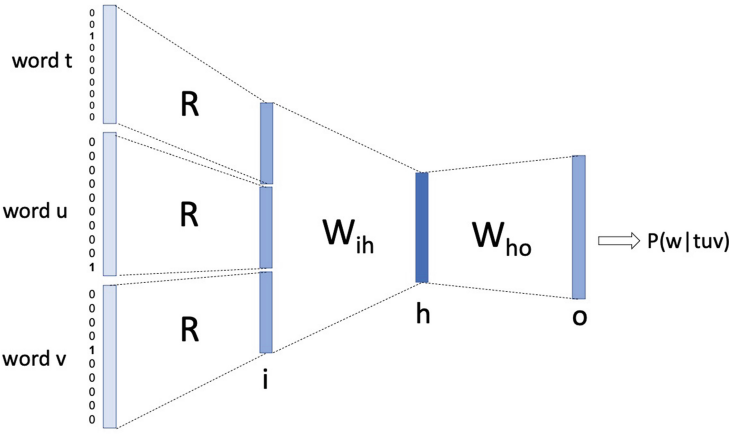

Fig. 1. A multi-layer feedforward network implementing a 4-gram model

The network computing $P(w|tuv)$ inputs the three vectors $\mathbf{t}, \mathbf{u}, \mathbf{v}$ in $\{0,1\}^{|V|}$, where words t, u, and v are replaced by binary vectors whose only non-zero component is the word index in the vocabulary (*one*-hot-encoding). The following computations are then performed:

$$\begin{aligned} \mathbf{i} &= [\mathbf{R};\mathbf{R};\mathbf{R}], \text{ with } \mathbf{R} \in \mathbb{R}^{3\,V| \times d_{\text{md}}} \\ \mathbf{h} &= \phi(\mathbf{W_{ih}}\mathbf{i} + \mathbf{b_{ih}}), \text{ with } \mathbf{W}_{ih} \in \mathbb{R}^{3d_{\text{md}} \times d_{\text{md}}} \text{ et } \mathbf{b}_{ih} \in \mathbb{R}^{d_{\text{md}}} \\ \mathbf{o} &= \mathbf{W}_{ho}\mathbf{h} + \mathbf{b_{ho}}, \text{ with } \mathbf{W_{ho}} \in \mathbb{R}^{d_{\text{md}} \times |V|} \text{ and } \mathbf{b}_{ho} \in \mathbb{R}^{|V|} \\ P(w|tuv) &= \text{softmax}(\mathbf{o})_w, \text{ with } \text{softmax}(\mathbf{x})_t = \frac{\exp(x_t)}{\sum_{t'} \exp(x_{t'})} \end{aligned} \tag{3}$$

These four steps respectively correspond to:

1. the computation of *dense numerical representations*, via the matrix $\mathbf{R}$, which projects each input vector into a d_{md} dimensional space, with $d_{\text{md}} \ll |V|$;
2. the introduction of a "nonlinearity", via the function $\phi()$, the hyperbolic tangent function (tanh) in the original implementation;
3. the calculation of non-normalized logits for each of the words that can follow the context *tuv*, obtained by comparing the output of the hidden layer $\mathbf{h}$ with the lexical output representations $\mathbf{W}_{ho}$
4. the normalization of these scores via the softmax operator, which outputs a probability vector.

Training such models requires to use numerical optimisation methods that adjust parameters in $\boldsymbol{\theta} = \{\mathbf{R}, \mathbf{W_{ih}}, \mathbf{b_{ih}}, \mathbf{W_{ho}}, \mathbf{b_{ho}}\}$ so as to make more likely the associations between contexts and words observed in a large corpus. Formally, for each sequence $[t, u, v, w]$ in the training corpus, one wish that the quantity $\log P(w|tuv) = o_w - \log \sum_{w'} \exp o_{w'}$ will be as large as possible. This leads to maximizing (in $\boldsymbol{\theta}$) the following *cross-entropy* criterion:

$$\ell(\boldsymbol{\theta}) = \sum_{[t,u,v,w]} \log P(w|tuv) = o_w - \log \sum_{w'} \exp o_{w'}. \tag{4}$$

This optimization is typically performed using stochastic gradient descent methods, which update parameters based on gradient values. Note that training again does not require any annotation and can be carried out on huge quantities of texts, as long as they can be segmented into "words".

The shift from discrete models to continuous space representations is computationally intensive, because computing the softmax operator involves a sum over a large vocabulary. Practical solutions are proposed and evaluated in [45,69]; we discuss them in Sect. 2.4. However, this shift has proven decisive to improve the quality of applications such as speech recognition or machine translation. This is because two quantities are learned simultaneously:

- a numerical representation $\mathbf{h}$ summarising the context made of several previous words into a low-dimensional vector, from which the conditional distribution of successor words is calculated. This ability to compute numerical representations of the prediction context is assimilated to the *encoding function* of the neural network.
- a lexical embedding of the vocabulary V into $\mathbb{R}^{d_{\text{md}}}$ through matrix $\mathbf{R}$. This embedding has remarkable properties; in particular, words that share many contexts, which are often semantically related words or words of the same morphological family, tend to get close in the embedding space. The use of these embeddings as generic lexical representations [20] has become widespread with the development of rapid and effective methods for calculating them [11,58].

Recurrent Neural Networks as Language Models. The previous model shares with the n-gram model the use of a context restricted to neighbouring $n-1$ words. The use of *recurrent networks* [27] makes it possible to overcome this limitation and to compute terms such as $P(w_{t+1}|w_1 \dots w_t)$ without resorting to locality assumptions. The strength of this approach and its superiority over the feedforward model are highlighted in [59], who present a network capable of taking into account an unbounded context. It contains the same two components as before, namely: (a) dense numerical representations for lexical units computed by the matrix $\mathbf{R}$; (b) a context encoding function defined here recursively by $\phi()$, which again denotes a non-linear function:

$$\begin{aligned}\mathbf{h_t} &= \phi(\mathbf{W_{h'h}h_{t-1}} + \mathbf{Rw_t} + \mathbf{b_h}) \\ &= \phi(\mathbf{W_{h'h}}\phi(\mathbf{W_{h'h}h_{t-2}} + \mathbf{Rw_{t-1}} + \mathbf{b}_h) + \mathbf{Rw_t} + \mathbf{b}_h).\end{aligned} \tag{5}$$

As before, the final step will project the internal representation $\mathbf{h}_t$ to yield the conditional output distribution associated with context $w_{\leq t} = w_1 \dots w_t$ according to $P(w|w_{\leq t}) = P(w|\mathbf{h_t}) = \text{softmax}(\mathbf{W}_{ho}\mathbf{h_t} + \mathbf{b_o})$. Parameters of the recurrent network $\{\boldsymbol{\theta} = \mathbf{R}, \mathbf{W_{h'h}}, \mathbf{W_{ho}}, \mathbf{b_h}, \mathbf{b_o}\}$ are trained by maximizing the cross-entropy loss function.

Unfolding the recursion (second line of Eq. (5)) makes the functional relationship between $\mathbf{h}_t$ and words $\mathbf{w}_t$ and $\mathbf{w}_{t-1}$, then, by recurrence, with all previous words. This also highlights that the influence of words decreases with their distance to the current position. It also highlights a computational difficulty associated to the direct computation of the gradient (by the rules of derivation of compound functions), which gives rise to numerical instabilities that make learning delicate. Remedies, which involve the use of more complex dependencies between $\mathbf{h_t}$ and $\mathbf{h_{t-1}}$ are proposed by [19,36]. They realize the full potential of these networks, which are then able to partly capture dependencies between distant words - such as the one observed in English between verb and subject, which must agree in number and person regardless of their distance in the sentence (see [50] for a study of such phenomena). Their expressiveness as a computational model is analyzed in [57].

In practice, however, the recursive formulation of the computation of latent representations poses a major problem, as it requires each sequence to be processed word by word from left to right in the order of their appearance. It is impossible to build $\mathbf{h_t}$ without having previously computed $\mathbf{h_{t-1}}$, which itself requires $\mathbf{h_{t-2}}$ etc.; such models are said to be *auto-regressive*. As a result, it is not possible to parallelize the computation of the objective function (Eq. (4))), which significantly slows down the training process.

Recurrent Models as "Pure" Representations: ELMo. Among the many extensions of these models, the most remarkable is their use as "pure" encoders. Let us first note again that learning such language models does not require annotation and can therefore be carried out on very large text corpora. Assuming that the parameters are known, a recurrent network transforms a string of words $w_1 \dots w_T$ into a sequence of vectors $\mathbf{h}_1 \dots \mathbf{h}_T$. The same process can be performed by running the sequence backwards, from w_T down to w_1, yielding another vector sequence $\tilde{\mathbf{h}}_\mathbf{1} \dots \tilde{\mathbf{h}}_\mathbf{T}$. Concatenating the two representations for word w_t yields $[\mathbf{h}_t; \tilde{\mathbf{h}}_t]$, which encodes w_t in a *bidirectional context* integrating both previous and subsequent words. It also turns out that $[\tilde{\mathbf{h}}_\mathbf{1}; \mathbf{h_T}]$ is a very good way to represent the whole variable-length sentence $w_1 \dots w_T$ into a *fixed-size vector*. This vector can then be used to compare sentences or to make predictions about their polarity or their meaning. It finally appears that stacking bi-directional recurrent layers, where $[\mathbf{h}_t; \tilde{\mathbf{h}}_t]$ are used as the input of a new layer, will deliver *deeper and better representations*. These principles are used to construct the ELMo model [60] model, one of the first to highlight the richness of these deep

contextual representations, which can serve as a beneficial plug-and-play pre-processing module for any application dealing with linguistic sequences.

Consider, for instance, the task of textual entailment, which consists of deciding whether sentence w_P logically entails sentence w_C. For this task, we need to predict a Yes/No answer given the two input sentences, yielding a model $P(\text{Yes}\,|w_P, w_C)$. A possible approach encodes each sentence into a single vector (respectively $\text{ELMo}(w_P)$ and $\text{ELMo}(w_C)$) that are then concatenated and used in a log-linear model according to: $P(\text{Yes}\,|w_P, w_C) \propto \exp(\mathbf{W}[\text{ELMo}(w_P); \text{ELMo}(w_C)] + \mathbf{b})$, where matrix $\mathbf{W}$ and vector $\mathbf{b}$ are the model parameters. By *pre-training* the parameters of the ELMo model, then by fine-tuning those of the textual entailment model, it becomes possible to achieve very good performance even when the train data of the textual entailment model is limited.

2.4 Defining the Vocabulary

We left open the question of the support of probability distributions represented by Eqs. (1) and (3). They presuppose the existence of a finite inventory V of discrete units. To model sequences of letters, sounds or syllables, this hypothesis is easy to defend. For sequences of words, it no longer makes sense, as no corpus, however large, can exhaust the word formation processes, not to mention borrowings from other languages, and extra-lexical (names, numbers, acronyms) whose occurrences must also be modelled. This issue has long been a source of difficulty for language models and has justified to take into account very large vocabularies, despite the associated computational problems.

A better trade-off is achieved by *abandoning the notion of word* and segmenting texts into sub-lexical units, by means of processes that are themselves optimized over large corpora to take frequencies of occurrences into account. Frequent words are thus preserved in their integrity, while the rarest words are split into subwords, if necessary reduced to mere sequences of letters. This makes it possible to manipulate medium-size vocabularies (containing tens of thousands of units), while at the same time preserving the ability to compute the probability of arbitrary sequences (possibly including unknown words, made of the concatenation of known subwords). The best known-algorithms for learning such vocabularies are the Byte Pair Encoding (BPE) algorithm [31,70] and the unigram algorithm [22,44].

Example segmentations realized by these algorithms are in Fig. 2.

_tous _les _êtres _humains _n aissent _libres _et _ég aux _en _dign ité _et _en _droits . _Ils _sont _dou és _de _raison _et _de _conscience _et _doivent _agir _les _uns _envers _les _autres _dans _un _esprit _de _fra tern ité .

_all _human _b e ings _a re _bor n _fre e _and _e qu al _in _dign ity _and _ri gh ts .

_alle _M ens ch en _sin d _fre i _un d _g le ich _an _W ü r de _un d _Re ch ten _g eb or en .

Fig. 2. Sub-lexical unit segmentation of the beginning of the French, English and German versions of the Universal Declaration of Human Rights. The vocabulary contains 10,000 units, character '_' identifies word-initial units. With this segmentation model optimized on French texts, only rare words (such as 'dignité', 'fraternité') are segmented. It is also used to segment, *with the same alphabet*, texts written in other languages, here sentences in English and German.

3 The Transformer Model

3.1 Attention, a Fundamental Mechanism

Having established all the necessary basic concepts of LMs, we now turn to the Transformer model, which relies on a more generic and powerful model to encode the context of each decision.

Encoding the Context. The main idea of the Transformer model of [77] is to make the representation of word w_t depend on all preceding words according to $\mathbf{h}_t = \phi(\mathbf{w}_1 \dots \mathbf{w}_t)$, while at the same time removing the recurrence of the computation of $\phi()$ so as to be able to parallelize it. In the Transformer model, this computation is achieved by stacking L layers. Each layer l recombines the representations from the previous layer $\mathbf{h}_1^{(l-1)} \dots \mathbf{h}_t^{(l-1)}$ to construct outputs $\mathbf{h}_1^{(l)} \dots \mathbf{h}_t^{(l)}$ through elementary operations: linear projections, linear combinations, vector concatenation, plus feedforward networks. The recursion of the recurrent model is thus replaced by a *stack of layers, each having a global scope*. The result remains the same as for other language models: a numerical vector representation of the context that summarises all the previous words, based on which one can predicts the next word in the sequence. Figure 3 illustrates the context encodings computed by these various architectures.

Formally, each layer in a Transformer is parameterized by a set of K *attention heads* and by a multi-layer feedforward model. Each attention head is parameterized by three projection matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ in $\mathbb{R}^{d_{\text{md}} \times d_{\text{kv}}}$ and performs the following computations to derive $\mathbf{h}_t^{(l)}$ from the outputs of the previous layer $\{\mathbf{h}_s^{(l-1)}, s = 1 \dots t\}$. For the k^{th} head in layer l:

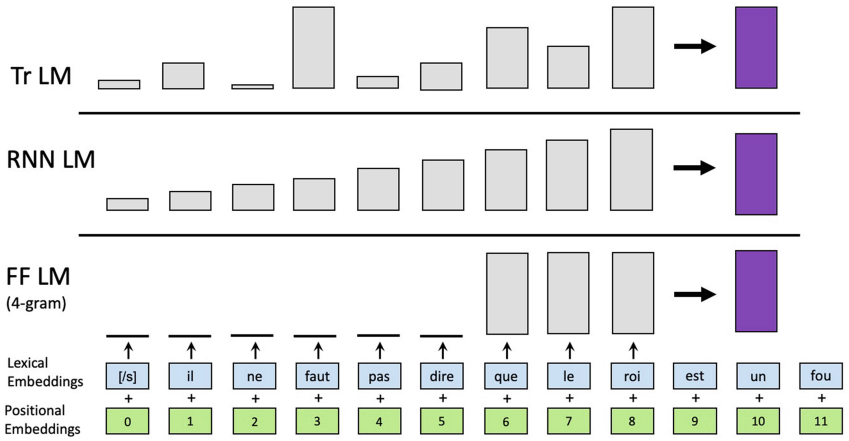

Fig. 3. The encodings of the left context computed by various language models: n-gram feedforward models (FF LM) encode only a small context; Recurrent models (RNNs) assume that close words are more important than remote words; Transformer models (Tr LM) process all the context words on an equal footing.

$$
\begin{array}{lrl}
(6.q)\ \text{Query} & \mathbf{q_t}^{(k,l)} = \mathbf{Q}^{(k,l)}\mathbf{h_t}^{(l-1)} & (\in \mathbb{R}^{d_{\text{kv}}}) \\
(6.k)\ \text{Keys} & \mathbf{k_s}^{(k,l)} = \mathbf{K}^{(k,l)}\mathbf{h_s}^{(l-1)}, \forall s \leq t & (\in \mathbb{R}^{d_{\text{kv}}}) \\
(6.v)\ \text{Values} & \mathbf{v_s}^{(k,l)} = \mathbf{V}^{(k,l)}\mathbf{h_s}^{(l-1)}, \forall s \leq t & (\in \mathbb{R}^{d_{\text{kv}}}) \\
(6.a)\ \text{Attention} & \alpha_s^{(k,l)} = \text{softmax}(\frac{1}{\sqrt{d_{\text{kv}}}}, \mathbf{o_t})_s, \forall s \leq t & (\in [0,1]) \\
& \text{avec } o_{ts} = \mathbf{q_t}^{(k,l)^T}\mathbf{k_s}^{(k,l)}, \forall s \leq t & \\
(6.o)\ \text{Output} & \mathbf{g_t}^{(k,l)} = \sum_{s \leq t} \alpha_s^{(k,l)} \mathbf{v_s}^{(k,l)} & (\in \mathbb{R}^{d_{\text{kv}}})
\end{array}
\tag{6}
$$

The first three steps compute d_{kv}-dimensional projections of their input, respectively called *query*, *key* and *value*. The dot product o_{ts} computes a similarity between the query at position t and the keys at all positions before t (included). These similarities are normalized by the softmax operator, which transforms them into *attention coefficients* in $[0, 1]$. The last step linearly combines the values to generate the output vector.

Each layer comprising several heads, it remains to aggregate their results. Two elementary operations come into play. The first is a transform made by a multi-layer perceptron according to:

$$
\begin{cases}
\mathbf{f}_t^{(l)} = \phi(\mathbf{W_{if}}^{(l)}\mathbf{g}_t^{(l)} + \mathbf{b_{if}}) \in \mathbb{R}^{d_{\text{ff}}}, \text{ with } \phi() \text{ a non-linear function.} \\
\mathbf{h}_t^{(l)} = \mathbf{W}_{fo}\mathbf{f}_t^{(l)} + \mathbf{b_{fo}} \in \mathbb{R}^{d_{\text{md}}}.
\end{cases}
\tag{7}
$$

The input $\mathbf{g}_t^{(l)}$ of this perceptron is the concatenation of the outputs of the K heads, to which we add the output of the previous layer: $\mathbf{g}_t^{(l)} = [\mathbf{g}_t^{(1,l)}; \ldots; \mathbf{g}_t^{(K,l)}] + \mathbf{h}_t^{(l-1)}$. Adding the output of the previous layer serves several purposes: (a) to provide gradients with a direct path from the higher layers to the lower layers; (b) to ensure that $\mathbf{h_t}^{(l-1)}$ and $\mathbf{h_t}^{(l)}$ remain close, and that each word thus retains its singularities, regardless of the influence of its context. One consequence is that both terms must have the same dimensions, which implies $K \times d_{\text{kv}} = d_{\text{md}}$. A typical implementation of this forward propagation step projects $\mathbf{g}_t^{(l)}$ via $\mathbf{W_{if}}^{(l)}$ into a d_{ff}-dimensional vector, with $d_{\text{ff}} \gg d_{\text{md}}$, for instance $d_{\text{ff}} = 4d_{\text{md}}$; the non-linearity of the hidden layer uses function $\phi() = \text{ReLU}$ (for *Rectified Linear Unit*).

The second basic operation normalizes the outputs, so that input and output vectors will remain commensurable throughout the layers. At a high level, each layer simply recombines the current representation at position t so as to incorporate the influence of the preceding words. Unlike the recurrent model, where the influence of the context words is computed in a left-to-right manner, in this model no position is privileged, and each attention head, in each layer, can select the positions that are the most significant for the current position, via the attention coefficients $\boldsymbol{\alpha}$.

Limiting Conditions: Layers 0 and *L*. We still have to describe the inputs and outputs of this system. For the input $\mathbf{h}_1^{(0)}$, one can choose to use either one-hot representations (see Sect. 2.3) or non-contextual representations computed by the skip-gram model [58]. However, lexical representations alone are not sufficient. In fact, equations ((6).[q–v]) do not distinguish between indices, whether close to or distant from the current position t. This illustrates the potential of Transformers to take into account non-local dependencies better than recurrent networks. However, it is useful to introduce the notion of position in the sequence, for example by encoding each index with s a d_{md}-dimensional vector $\mathbf{p}_s$, which is added to the lexical embedding. This *positional encoding* is either learned or computed by a deterministic function defined in [77] as:

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

The output of the last layer $\mathbf{h}_t^{(L)}$ is used to compute the probability of the next word at position $t+1$ and involves the same steps as for the standard neuronal model (Eq. (3)): a linear transformation in a $|V|$-dimensional space to obtain logits, that are then normalized into a probability distribution.

3.2 Causal Transformer as Pure Language Models

The presentation above expresses the computation of $h_t^{(l)}$ as a sequential operation: $\mathbf{h}_1^{(l)}, l = 1 \ldots L$ are first computed, then $\mathbf{h}_2^{(l)}, l = 1 \ldots L$ in the context of $\mathbf{h}_1^{(l)}, l = 1 \ldots L$, etc. This is the most natural and computationally effective

method for language models, since the representation of each word is computed only once. This model is dubbed as *self-attentional* (since the context consists of the previous words in the same sequence) and *causal* (the representation of each word only depends on the previous words). It is used for instance in the GPT-* architectures [14,64]. A non-causal variant recomputes all representations at each time steps, i.e. first $\mathbf{h}_1^{(l)}, l = 1 \dots L$, then $\{\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, l = 1 \dots L\}$: this means, for instance, that the representation $h_1^{(l)}$ will change over time, integrating the context of words to its right as they are revealed. This variant, sometimes called *prefix language model*, is more computationally involved, but seems to yield better results [65].

Like other LMs, these architectures generate texts from left to right, by sampling at time step t the next word w_t according to $P(w_t|\mathbf{h}_{t-1})$. Trained on very large corpora, they generate texts that are often grammatically correct, or even show a form of global consistency. This suggests that Transformers are able to model dependencies between distant words in the same sentence or even in different sentences (when the context is sufficiently large). It is also possible to begin a text and let the model continue the generation. This form of text initialization is dubbed "prompting", where the prompt denotes the user-selected prefix. It turns out that with a proper choice of these "prompts", a language model can perform multiple tasks: when the prompt is composed of a text and comprehension questions, the model generate the answers, when prompted with a sentence in French, the model generates the translation into English, etc. It is possible, at the cost of a short learning stage, or by prompting with a handful of examples, to improve this behavior [64]. Owing to these remarkable properties, some of which are still poorly understood, large language models nowadays constitute a basic building block to address a large number of language processing tasks.

3.3 Transformers as Representations: Bert and Its Clones

As with recurrent models, the use of Transformers as pure contextualized representation extractors has proved to be extremely powerful and effective. Originally proposed in [23], the BERT model is mostly a *non-sequential and non-causal* Transformer. This means that the update formulas of Eq. (6) simultaneously apply to all positions in the input sequence via matrix calculations, and that the contextual representations of each word integrate both its left and right contexts. Learning such model is typically performed by optimizing the reconstruction of a noised input: in its basic form, noising simply randomly hides (masks) some tokens w_t, which the model then seeks to recover based on the representation $h_t^{(L)}$. By maximizing the log-probability of original text, it becomes possible to estimate model parameters (see Fig. 5) (Fig. 4).

As for other language models, this learning process does not require any annotation and can be performed on very large corpora. BERT thus provides contextualised "pre-trained" lexical representations that can be used as input to any automatic processing system [23,52]. Other methods for constructing denoising tasks have also been studied [65]: masking groups of words, parts

Prompt	Automatic Generation
In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.	The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.
Article: Amina Ali Qassim is sitting with her youngest grandchild on her lap, wiping away tears with her headscarf. Only a few months old, this is the baby girl whose ears she desperately tried to cover the night the aerial bombardment started. She lay awake, she says, in a village mosque on the Yemeni island of Birim, counting explosions as the baby cried. It could have been worse though. They could have still been in their house when the first missile landed. "Our neighbor shouted to my husband 'you have to leave, they're coming.' And we just ran. As soon as we left the house, the first missile fell right by it and then a second on it. It burned everything to the ground," Qassim tells us (...) **DL;TR**	Yemen is in the middle of a civil war. Saudi Arabia is leading the coalition bombing campaign. It's been bombing Yemen for more than two months now.
Question: George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? Answer:	dry palms
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 (~ 1.6km) 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?	Neither

Fig. 4. Prompting a language model. By varying the prompt, the same model can be used to perform multiple tasks. From top to bottom, language generation is used to write the continuation of a prompted text, to write a summary, to answer a question, to check the implication between two sentences, etc. Examples are from [14,64]

Input Tous les êtres humains [MASK] libres et égaux en dignité et en droits. [MASK] sont doués de raison et de conscience et doivent [MASK] les uns envers les autres dans un esprit de [MASK].

Output Tous les êtres humains **naissent** libres et égaux en dignité et en droits. **Ils** sont doués de raison et de conscience et doivent **agir** les uns envers les autres dans un esprit de **fraternité**.

Fig. 5. Learning the BERT model with random masking. The model parameters are trained to maximize the probability of recovering the hidden tokens (bold on the figure).

of words, deleting and permuting words, etc. Due to their performance, these models have quickly become central in NLP [71] and were quickly adapted to multiple languages, such as French [46,55], German [16], Dutch [78], Spanish [15], etc. Versions adapted to specialized textual genres such as patents [47], scientific texts [5], tweets [4] or even sub-domains such as medicine [48] or nuclear physics [37] have also developed. A rich literature also studies the empirical behaviour of Transformers, trying in particular to analyze the internal representations $\{h_t^l, t = 1 \dots T, l = 1 \dots L\}$ in relationship to linguistic concepts; or use attention matrices as a source of explanation of the system's decisions. A recent bibliographical survey of this "Bertological" literature has no less than 110 references [67].

3.4 Computational Costs of Transformer-ing

As Transformer based models take a central role in natural language processing, it becomes necessary to take a closer look at the computations performed by these algorithms and to better assess their cost, in a context where the carbon footprint of Artificial Intelligence algorithms is also becoming a concern [35, 73]. A first observation is that, unlike recurring networks, the computations

of Transformers are easy to parallelize. In particular, computing the output representation of a word requires knowledge of its neighboring words, but not of their deep output representations. These output representations can then all be computed simultaneously by implementing Eq. (6) with operations. Table 1 provides indications regarding the size and number of parameters for some recent models.

Table 1. Measuring the size of Transformer models. The number of parameters used for lexical (L) and internal (I) representations are counted separately. Notations k, m, b respectively denote thousands, millions and billions of units.

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

A last important dimension for complexity calculations is the sequences length T, which determines the overall dimension at the input and output of each layer ($T \times d_{\text{md}}$). Sequences typically contains several hundreds of words or even more (2048 for GPT-3). During training, it is necessary to keep the values of all layers in memory, as they are needed to compute the gradient. To speed up calculations, batches of B sequences are processed simultaneously, yielding tensors of dimension $B \times T \times d_{\text{md}}$, whose manipulations are optimized on GPU cards.

The computational complexity of the Transformer operations is dominated by the evaluation of attention matrices in Eq. (6). This computation is linear in d_{md}, but quadratic in T: for each of the T positions, the similarity with all the other positions need to be computed, in order to derive the attentions weights $\boldsymbol{\alpha}$, then the T output values vectors. To reduce this complexity, several directions are considered in the literature. It is first possible to restrict the computation of attention weight to a neighborhood $N(t)$ of w_t, by imposing words outside $N(t)$ to have null weights ($\alpha_t(s) = 0, \forall s \notin N(t)$); note that these words still influence w_t indirectly by influencing its neighbours (or the neighbors of its neighbors) across the multiple computations layers. By choosing neighborhoods $N(t)$ of fixed size S, with S much smaller than T, the attention computation becomes linear in T. There are several other ways to define $N(t)$, using syntactic dependencies, or using random subsets of indices: what matters is that for almost every word, $|N(t)|$ is small and, that for a few positions, $N(t)$ encompasses the whole sequence. Other approaches to speed up these computations focus on effective approximations of dot products (Eq. (6)a). A recent survey of effective implementations of the Transformer model is in [75]; some methods aimed to

reduce the memory footprint are presented in [66]. Since the amount of training data continues to increase the performance for many tasks [42], developing larger models is likely to remain an active area of research [29], posing formidable computational challenges both for learning and inference.

3.5 Transformers: A Flexible Architecture

The language models implemented in Transformer architectures combine all the advantages of neuronal architectures: they can learn both predictive models capable of taking into account long-range dependencies and rich contextual representations for atomic units, which can be pre-trained and then used for multiple language processing tasks. They result in effective implementations [79], and have also been adapted for other types of structured data: acoustic sequences for speech modelling [2], images for artificial vision [63], and even image sequences [74]. Like other neural language models, their behavior remains difficult to control: while some regularities are almost perfectly learned, others are learned only approximately, and it is difficult to predict or understand the reasons for these failures.

4 Towards Multilingualism

4.1 Neural Machine Translation: Conditional Text Generation

A Simple Encoder-Decoder Model. The Transformer model presented above as a language model is initially introduced for machine translation (MT) [77]. This application formally corresponds to the generation (in a "target language") of a sentence $\mathbf{e}$ translating the input "source" sentence $\mathbf{f}$. Viewed as a probabilistic decision, this problem corresponds to finding:

$$\mathbf{e}^* = \text{argmax}_{\mathbf{e}}\, P(\mathbf{e}|\mathbf{f}) = \text{argmax}_{\mathbf{e}} \prod_t P(e_t|\mathbf{f}, \mathbf{e}_{<\mathbf{t}}). \tag{8}$$

This formalization again requires to define a probability distribution over a set of sentences (see Eq. (1)), except that this distribution is conditioned by the input sentence $\mathbf{f}$. The Transformer model computes such a distribution by extending the neural encoder-decoder architectures proposed for MT in [3,18]. These architectures rely on two computation steps:

(a) the computation of a numerical representation (encoding) for $\mathbf{f}$ taking the form of a sequence of numerical vectors $\mathbf{s}_1, \ldots, \mathbf{s}_J$;
(b) the iterative decoding of the translation, by choosing at each step the most likely next word e_t given the source encoding $[\mathbf{g}_1^{(l)}, \ldots, \mathbf{g}_I^{(l)}], l = 1 \ldots L$ as well as the previous target words $\mathbf{e}_{<t}$, encoded as previously as $[\mathbf{h}_1^{(l)}, \ldots, \mathbf{h}_{t-1}^{(l)}], l = 1 \ldots L$.

The first neural MT systems perform these two steps using recurrent networks (Sect. 2.3). In a Transformer-based architecture, step (a) is performed by a non-causal encoder (Sect. 3.3) and step (b) is performed by a causal decoder (Sect. 3.2). During this stage, it is necessary to integrate the double dependency in $\mathbf{f}$ and $\mathbf{e}_{<t}$, since the prediction of the next target word is influenced by these two sequences (see Eq. (8)). This is implemented by the addition of an additional *cross-attentional sublayer* within the decoder. The corresponding computations are similar to those of Eq. (6), using $\mathbf{h}_t^{(l)} \dots \mathbf{h}_t^{(l)}$ for the query, and $\mathbf{g}_1^{(L)}, \dots, \mathbf{g}_J^{(L)}$ for keys and values. In this way, the context vector of each target word integrates not only the target prefix, but also all the words in the source phrase, represented at the last layer (L) of the encoder. As before, the last layer vector of the decoder is projected into a $|V|$—dimensional space, then normalized through the softmax function to provide the desired output distribution $P(\mathbf{e}_t|\mathbf{f}, \mathbf{e}_{<t})$.

Difficulties of Machine Translation

Learning from Parallel Corpora. The learning of conditional models is similar to the learning of language models and consists of optimizing the log-probability of the training sequences, which decomposes into a sum of terms as in Eq. (4). This computation requires both words from the source and the target sentences, which are aligned in large *parallel corpora* matching sentences with their translation. Such resources are now publicly available en masse from resource distribution agencies such as ELDA or the Linguistic Data Consortium. A variety of parallel corpora can be found specialized websites such as OPUS [76].

Machine Translation is Difficult. Once training (which can take days, depending on the amount of available parallel data) is complete, the Transformer is ready to translate. Translation is performed incrementally, word by word, in a greedy manner and poses the same difficult problems as the unconditional generation of texts. It appears, on the one hand, that choosing at each time step the best next word is a risky strategy, since each past error might yield incorrect or simply unusual internal representations, which in turn can cause more errors. This problem is known as the *exposure bias problem* [8] and requires to use of more sophisticated search strategies, such as beam search, to compute the argmax (Eq. (8)). An alternative decoding strategy simultaneously predicts all the target words in parallel, which dramatically speeds up decoding. However, global constraints on the relative positions of words must apply to ensure that the target sentence remains well-formed [32].

Two additional difficulties are directly related to the machine translation problem: in MT, it is necessary to *translate the entire source phase* (each word only once), *without introducing any additional information*. However, these two constraints are not explicitly formulated in Eq. (8): to ensure that the length of the target sentence matches that of the source, and effectively translates all input words, the search algorithm must include additional heuristics: [40] presents the most commonly used ones.

4.2 Multilingual Representations, Multilingual Translations

An additional benefits of numeric representations is that they represent words of different languages in a unified manner. It is then possible, assuming a shared units directory for all languages, to use the same encoders and decoders to process multiple languages. The easiest way to proceed is to implement the same learning procedure as for BERT (Sect. 3.3), inputting sentences in *multiple languages* into the system: this approach is used for mBERT [23] and XLM [21].

Such approaches readily deliver *multilingual contextual representations* that bring together, in the same vector space, units (words, sentences) that are mutual translations. Learning multilingual representations thus makes parallel sentences in multiple languages almost indistinguishable (for the neural network). This enables to transfer processing models and applications from a resource-rich language into languages for which resources do not exist. Let us take the example of a sentiment analysis system, which aims to associate textual comments on a merchant site with satisfaction scores, and assume that we have annotated training examples for language A, but not for language B. Learning to predict the numerical score from *multilingual representations of texts in language* A makes us also able to predict the note of texts in language B *without ever having observed any training example associating a text in language* B *with its evaluation* (see Fig. 6).

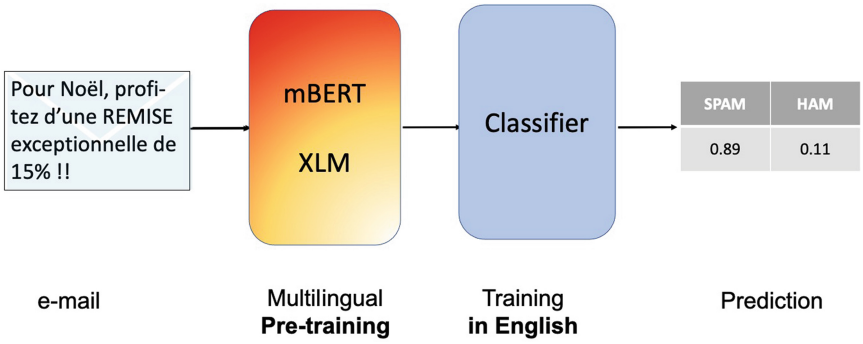

Fig. 6. A multilingual architecture for spam filtering. The second step uses multilingual pre-trained representations which enable to transfer knowledge across languages: French spam mails can then be identify even though the classifier has never seen any French example.

Learning multilingual representations is therefore a major challenge to broaden the spectrum of languages covered by language technologies. Many approaches have also been proposed to train non-contextual multilingual representations, or to adapt existing representations to a specialized domain. A recent survey of these methods is in [72].

Note finally, that the encoder-decoder architecture can also be used to compute *monolingual representations*: this is again achieved by inputting noisy texts

into the encoder, that the decode will then have to recover. All that is needed is the definition of the noising operations used to generate parallel artificial data: masking one or several words, replacing a word with a similar word, changing the order of words are typical noising operations. BART, introduced by [49], has the benefits of a faster learning than BERT (more words are noised in the encoder). With an additional fine fine-tuning stage, BART can also be used as a generation model, as the decoder is non-causal: a possible application there is automatic summarization. Finally, like BERT, BART can be trained multilingually, simultaneously computing multilingual representations and machine translation.

4.3 One Model to Translate Them All

Multilingual translation combines the approaches described in the previous sections: the use of an encoder-decoder architecture, with conditional generation of texts; the use of sentence pairs combining input and output for multiple language pairs. This idea, originally proposed in [30,34] and recently used on a large scale in [1,28] opens new perspectives: (a) operationally it means that we just need one single system to handle all translations between N languages, where $O(N^2)$ where previously required; (b) it also enable to compute translations between languages for which no data is observed (again through cross-lingual transfer, which happens here both in the encoder and in the decoder).

This approach is not without difficulty, in particular from the point of view of collecting and balancing parallel learning data, as well as supporting a variety of linguistic systems, which may, for example, use different writing systems, or manipulate divergent underlying structures at the levels of word or phrases. For such multilingual models, a necessary pre-processing step is to learn a shared tokenization (in word and subwords, see Sect. 2.4) using multilingual corpora, so that all input-outputs in the system use the same vocabulary.

4.4 Machine Translation as a Generic Task

Generalizing Machine Translation. The transition from the unconditional model (Sect. 3) to the the conditional model (Sect. 4.1) outlines the flexibility of numerical representations manipulated by neural networks: by adding a cross-attention mechanism between two Transformers, it is possible to encode two word sequences in a single vector $\mathbf{h}$, from which the next word is generated. This technique enables to encode "generalized" contexts, to model more complex tasks or scenarios. It is for instance possible to handle *multi-source translation scenarios*, corresponding to the generation of a target sentence from several two sentences $\mathbf{f_1}$ and $\mathbf{f_2}$. In such setting, the distribution $P(\mathbf{e}|\mathbf{f_1}, \mathbf{f_2})$ can be obtained by computing the cross-lingual based on a concatenation of the two source encodings. Another illustration of this flexibility is document-level translation, which aims to integrate long-distance dependencies beyond the sentence level. This might be needed to handle pronominal references, as when computing the translation

of "*This bike is broken. It needs a fix.*" from English into French. For this example, the generation of the correct subject pronoun for the second sentence ("*il*" ou "*elle*") requires knowledge of the translation of "*bike*" in the previous sentence: ("*vélo*" will imply a masculine subject, "*bicyclette*" a feminine one). By encoding contexts made of several previous sentences, such difficulties can be addressed [56].

Monolingual and Multimodal Machine Translation. Machine translation is an extreme example of a sequence transduction task, corresponding to a language change, while preserving the global meaning. Similar problems appear in a large number of monolingual tasks: for example, grammar correction can be viewed as a "translation" between a noisy sentence and its correction, a framework that also includes spelling normalization (to turn short texts into standard English). Simplification, paraphrase generation, style transfer (e.g from a formal style to more relaxed style), automatic summarization [51] are other instances of these monolingual translations: assuming the availability of pairs (input, output) to learn the parameters, it will be possible to use Transformer architectures.

The encoder-decoder architecture is also generalized in other ways. By considering pairs associating voice recordings with their transcription, it is possible to apply the same techniques for automatic speech recognition [24,33,43]; or even, when recordings and transcripts are in different languages, to directly translate the speech into foreign text [10]. Similar approaches consider the recognition of patterns in images [25] or the generation of descriptions from images [53]. The application of the Transformers to these other modalities is only starting and is expected to develop, both to learn generation models and to train multimodal representations. An introduction to these exciting developments is presented in this volume in M. Evrard's chapter on *Transformer in Automatic Speech Recognition* and C. Guinaudeau's chapter on *Vision and Multi-modal Transformers*.

4.5 Summary

The Transformer architecture readily generalizes to the conditional generation framework, with an interdependent encoder and decoder, an approach that has quickly become the de facto standard for neural machine translation. When fed with inputs and outputs in multiple languages, this architecture learns multilingual representations that can be used for cross-lingual transfer in many applications. By analogy with the translation task, the same approach model can be used for many monolingual tasks as well as for tasks involving other modalities (speech, image, video).

5 Conclusion

The Transformer architecture, both in its unconditional (Sect. 3) and in its conditional (Sect. 4) versions has quickly emerged as a critical component of all language processing tools and has often led to considerable improvements of the

performance of these systems. This architecture generates contextual representations from vast amounts of raw data; these representations are useful for a wide range of applications, and also enable to transfer learned knowledge between tasks, domains and languages. This provides an operational response to the lack of annotated data that would be necessary to carry out supervised learning in many contexts. It is also used to learn word generation models capable of producing coherent texts, and, at the cost of elementary reformulations, to handle a large number of related tasks: sentiment analysis, textual implication, question answering, summarization, translation, etc. Multilingual and multimodal extensions of these architectures make it possible to build models from heterogeneous data, further opening the range of possible applications. Finally, Transformers define a shared conceptual framework for many communities of researchers and developers, facilitating interdisciplinary exchanges and accelerating the dissemination of effective implementations and sharing of models [12].

Have Transformer "solved natural language processing"? Several limitations of these models are highlighted in recent papers, suggesting many avenues for future research. A first limitation is that these models do not incorporate any linguistic knowledge (regarding the structure of words and phrases), which makes them unsuitable for reproducing the systematic behaviour that is expected when dealing with regular phenomena, such as grammatical agreement, or coreference phenomena. Although possible, the integration of linguistic knowledge runs against the increase in training data and in the number of languages taken into account, and is not very actively researched. Similarly, the world knowledge injected into Transformers is restricted to whatever occurs in the training texts. Although these models are capable of memorizing and restoring many of these factual knowledge, they remain incomplete and their learning is uncertain and non-systematic [62]: it thus seems inappropriate to think that they help us progress towards deep language understanding [7]. For this, the combination of statistical models with knowledge graphs seems to be a promising research direction [13,61].

Another limitation of these architectures is that their "black box" behaviour, which creates multiple problems when these systems are run on a very large scale. In particular, it is extremely hard to explain the decisions made, as they ultimately result from the particular dynamics of model training, and from the nature of the training data. As shown on many occasions [6], these models in particular tend to amplify the biases present in the data, and may, for example, generate uncontrolled statements of a sexist or racist nature. The apparent consistency of automatically generated texts is also misleading, and may fool users into endowing these systems with a form of understanding they do not actually possess. These weaknesses are shared with all probabilistic models, which are constrained in their performance by the limitations of the training data, which are often too rare, incomplete, or biased, and result in systems that may be incomplete and inconsistent.

References

1. Aharoni, R., Johnson, M., Firat, O.: Massively multilingual neural machine translation. 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), pp. 3874–3884. Association for Computational Linguistics, Minneapolis (2019). https://doi.org/10.18653/v1/N19-1388
2. Baevski, A., Zhou, Y., Mohamed, A., Auli, M.: Wav2vec 2.0: a framework for self-supervised learning of speech representations. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 12449–12460. Curran Associates, Inc. (2020)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of the first International Conference on Learning Representations. ICLR, San Diego (2015)
4. Barbieri, F., Camacho-Collados, J., Espinosa Anke, L., Neves, L.: TweetEval: unified benchmark and comparative evaluation for tweet classification. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 1644–1650. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.findings-emnlp.148
5. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3615–3620. Association for Computational Linguistics, Hong Kong (2019). https://doi.org/10.18653/v1/D19-1371
6. Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: 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 2021, pp. 610–623. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3442188.3445922
7. Bender, E.M., Koller, A.: Climbing towards NLU: on meaning, form, and understanding in the age of data. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 5185–5198. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.acl-main.463
8. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 28. Curran Associates, Inc. (2015)
9. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**, 1137–1155 (2003)
10. Bérard, A., Pietquin, O., Besacier, L., Servan, C.: Listen and translate: a proof of concept for end-to-end speech-to-text translation. In: NIPS Workshop on End-to-End Learning for Speech and Audio Processing, Barcelona, Spain (2016)
11. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
12. Bommasani, R., et al.: On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258 (2021)
13. Bosselut, A., Rashkin, H., Sap, M., Malaviya, C., Celikyilmaz, A., Choi, Y.: COMET: commonsense transformers for automatic knowledge graph construction. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4762–4779. Association for Computational Linguistics, Florence (2019). https://doi.org/10.18653/v1/P19-1470

14. Brown, T.B., et al.: Language models are few-shot learners (2020)
15. Cañete, J., Chaperon, G., Fuentes, R., Ho, J.H., Kang, H., Pérez, J.: Spanish pre-trained BERT model and evaluation data. In: PML4DC at ICLR 2020 (2020)
16. Chan, B., Schweter, S., Möller, T.: German's next language model. In: Proceedings of the 28th International Conference on Computational Linguistics, pp. 6788–6796. International Committee on Computational Linguistics, Barcelona (2020). https://doi.org/10.18653/v1/2020.coling-main.598
17. Charniak, E.: Statistical Language Learning. The MIT Press, Cambridge (1993)
18. Cho, K., et al.: 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), pp. 1724–1734. Association for Computational Linguistics, Doha (2014)
19. Cho, K., van Merriënboer, B., Bahdanau, D., Bengio, Y.: 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, pp. 103–111. Association for Computational Linguistics, Doha (2014). https://doi.org/10.3115/v1/W14-4012
20. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(Aug), 2493–2537 (2011)
21. Conneau, A., Lample, G.: Cross-lingual language model pretraining. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019)
22. Deligne, S., Bimbot, F.: Language modeling by variable length sequences: theoretical formulation and evaluation of multigrams. In: 1995 International Conference onAcoustics, Speech, and Signal Processing, ICASSP-95, vol. 1, pp. 169–172. IEEE (1995)
23. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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), pp. 4171–4186. Association for Computational Linguistics, Minneapolis (2019)
24. Dong, L., Xu, S., Xu, B.: Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5884–5888. IEEE (2018)
25. Dosovitskiy, A., et al.: An image is worth 16 × 16 words: transformers for image recognition at scale. In: International Conference on Learning Representations (2021)
26. Eisenstein, J.: Natural Language Processing. The MIT Press, Cambridge (2019)
27. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990). https://doi.org/10.1207/s15516709cog1402_1
28. Fan, A., et al.: Beyond English-centric multilingual machine translation. J. Mach. Learn. Res. **22**(107), 1–48 (2021)
29. Fedus, W., Zoph, B., Shazeer, N.: Switch transformers: scaling to trillion parameter models with simple and efficient sparsity (2021)
30. Firat, O., Cho, K., Bengio, Y.: Multi-way, multilingual neural machine translation with a shared attention mechanism. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 866–875. Association for Computational Linguistics (2016). https://doi.org/10.18653/v1/N16-1101

31. Gage, P.: A new algorithm for data compression. Comput. Users J. **12**(2), 23–38 (1994)
32. Gu, J., Bradbury, J., Xiong, C., Li, V.O.K., Socher, R.: Non-autoregressive neural machine translation. In: Proceedings of the International Conference on Representation Learning, ICLR 2018 (2018)
33. Gulati, A., et al.: Conformer: convolution-augmented transformer for speech recognition. In: Proceedings of the Interspeech 2020, pp. 5036–5040 (2020)
34. Ha, T.H., Niehues, J., Waibel, A.: Toward multilingual neural machine translation with universal encoder and decoder. In: Proceedings of the 13th International Workshop on Spoken Language Translation, IWSLT 2016, Vancouver, Canada (2016)
35. Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D., Pineau, J.: Towards the systematic reporting of the energy and carbon footprints of machine learning. J. Mach. Learn. Res. **21**(248), 1–43 (2020)
36. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
37. Jain, A., Ganesamoorty, M.: NukeBERT: a pre-trained language model for low resource nuclear domain. CoRR abs/2003.13821 (2020)
38. Jelinek, F.: Statistical Methods for Speech Recognition. The MIT Press, Cambridge (1997)
39. Jelinek, F., Mercer, M.: Interpolated estimation of Markov source parameters from sparse data. In: Proceedings of the Workshop on Pattern Recognition in Practice, pp. 381–397. Amsterdam (1980)
40. Johnson, M., et al.: Google's multilingual neural machine translation system: enabling zero-shot translation. Trans. Assoc. Comput. Linguist. **5**, 339–351 (2017)
41. Jurafsky, D., Martin, J.H.: Speech and Language Processing, 3éme édition, 2018 edn. Prentice Hall (2000)
42. Kaplan, J., et al.: Scaling laws for neural language models (2020)
43. Karita, S., Soplin, N.E.Y., Watanabe, S., Delcroix, M., Ogawa, A., Nakatani, T.: Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration. In: Proceedings of the Interspeech 2019, pp. 1408–1412 (2019)
44. Kudo, T.: Subword regularization: improving neural network translation models with multiple subword candidates. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 66–75. Association for Computational Linguistics, Melbourne (2018). https://doi.org/10.18653/v1/P18-1007
45. Le, H.S., Oparin, I., Allauzen, A., Gauvain, J.L., Yvon, F.: Structured output layer neural network language model. In: Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing, Prague, Czech Republic, pp. 5524–5527 (2011)
46. Le, H., et al.: FlauBERT: unsupervised language model pre-training for french. In: LREC. Marseille, France (2020)
47. Lee, J.S., Hsiang, J.: Patent classification by fine-tuning BERT language model. World Patent Inf. **61**, 101965 (2020). https://doi.org/10.1016/j.wpi.2020.101965
48. Lee, J., et al.: BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinform. (Oxford Engl.) **36**(4), 1234–1240 (2020). https://doi.org/10.1093/bioinformatics/btz682
49. Lewis, M., et al.: BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of the 58th

Annual Meeting of the Association for Computational Linguistics, pp. 7871–7880. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.acl-main.703
50. Linzen, T., Dupoux, E., Goldberg, Y.: Assessing the ability of LSTMs to learn syntax-sensitive dependencies. Trans. Assoc. Comput. Linguist. **4**, 521–535 (2016)
51. Liu, P.J., et al.: Generating Wikipedia by summarizing long sequences. CoRR abs/1801.10198 (2018)
52. Liu, Y., et al.: RoBERTa: a robustly optimized BERT pretraining approach. CoRR abs/1907.11692 (2019)
53. Lu, J., Batra, D., Parikh, D., Lee, S.: ViLBERT: pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019)
54. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge (1999)
55. Martin, L., et al.: CamemBERT: a tasty french language model. In: ACL 2020–58th Annual Meeting of the Association for Computational Linguistics. Seattle/Virtual, United States (2020). https://doi.org/10.18653/v1/2020.acl-main.645
56. Maruf, S., Saleh, F., Haffari, G.: A survey on document-level neural machine translation: Methods and evaluation. ACM Comput. Surv. **54**(2) (2021). https://doi.org/10.1145/3441691
57. Merrill, W., Weiss, G., Goldberg, Y., Schwartz, R., Smith, N.A., Yahav, E.: A formal hierarchy of RNN architectures. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 443–459. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.acl-main.43
58. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of the International Conference on Representation Learning. ICLR (2013)
59. Mikolov, T., Karafiát, M., Burget, L., Černocký, J., Khudanpur, S.: Recurrent neural network based language model. In: Proceedings of the 11th Annual Conference of the International Speech Communication Association (Interspeech 2010), pp. 1045–1048. International Speech Communication Association (2010)
60. Peters, M., et al.: 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), pp. 2227–2237. Association for Computational Linguistics, New Orleans (2018)
61. Peters, M.E., et al.: Knowledge enhanced contextual word representations. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 43–54. Association for Computational Linguistics, Hong Kong (2019). https://doi.org/10.18653/v1/D19-1005
62. Petroni, F., et al.: Language models as knowledge bases? In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 2463–2473. Association for Computational Linguistics, Hong Kong (2019). https://doi.org/10.18653/v1/D19-1250
63. Qi, D., Su, L., Song, J., Cui, E., Bharti, T., Sacheti, A.: ImageBERT: cross-modal pre-training with large-scale weak-supervised image-text data (2020)
64. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. Technical report, OpenAI (2019)

65. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)
66. Rajbhandari, S., Rasley, J., Ruwase, O., He, Y.: ZeRO: memory optimizations toward training trillion parameter models. In: SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–16 (2020). https://doi.org/10.1109/SC41405.2020.00024
67. Rogers, A., Kovaleva, O., Rumshisky, A.: A primer in BERTology: what we know about how BERT works. Trans. Assoc. Comput. Linguist. **8**, 842–866 (2020)
68. Rosenfeld, R.: Two decades of statistical language modeling: where do we go from here?? Proc. IEEE **88**(8), 1270–1278 (2000). https://doi.org/10.1109/5.880083
69. Schwenk, H.: Continuous space language models. Comput. Speech Lang. **21**(3), 492–518 (2007)
70. Sennrich, R., Haddow, B., Birch, A.: 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), Berlin, Germany, pp. 1715–1725 (2016). https://doi.org/10.18653/v1/P16-1162
71. Smith, N.A.: Contextual word representations: putting words into computers. Commun. ACM **63**(6), 66–74 (2020). https://doi.org/10.1145/3347145
72. Søgaard, A., Vulic, I., Ruder, S., Faruqui, M.: Cross-lingual word embeddings. In: Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers (2019). https://doi.org/10.2200/S00920ED2V01Y201904HLT042
73. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in NLP. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 3645–3650. Association for Computational Linguistics, Florence (2019). https://doi.org/10.18653/v1/P19-1355
74. Sun, C., Myers, A., Vondrick, C., Murphy, K., Schmid, C.: VideoBERT: a joint model for video and language representation learning. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 7463–7472 (2019). https://doi.org/10.1109/ICCV.2019.00756
75. Tay, Y., Dehghani, M., Bahri, D., Metzler, D.: Efficient transformers: a survey (2020)
76. Tiedemann, J.: Parallel data, tools and interfaces in opus. In: Choukri, K., et al. (eds.) Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012). European Language Resources Association (ELRA), Istanbul (2012)
77. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 5998–6008. Curran Associates, Inc. (2017)
78. de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., Nissim, M.: Bertje: a dutch BERT model. CoRR abs/1912.09582 (2019)
79. Wolf, T., et al.: Transformers: state-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.emnlp-demos.6