

# 5 – LLMs

IASD / MASH – LLMs course

Florian Le Bronnec

November 8, 2023

# Table of Contents

## ① Reminders

- Attention
- Transformer layer
- Subsequent questions

## ② Pre-training

- Generalities on pre-training
- Pre-training for classification
- Word2Vec
- BERT

## ③ Generation

- Conditional language modeling
- Other architectures
- Inference

## ④ Conclusion

# Table of Contents

## ① Reminders

- Attention
- Transformer layer
- Subsequent questions

## ② Pre-training

- Generalities on pre-training
- Pre-training for classification
- Word2Vec
- BERT

## ③ Generation

- Conditional language modeling
- Other architectures
- Inference

## ④ Conclusion

# Transformer

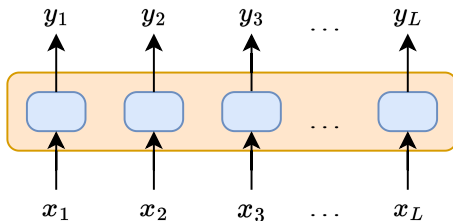


Figure 1: Base transformer layer.  $\mathbf{x}_1, \dots, \mathbf{x}_L \in \mathbb{R}^d$  and  $\mathbf{y}_1, \dots, \mathbf{y}_L \in \mathbb{R}^d$ .

A **transformer** is a sequence processing architecture. It takes a sequence as input and outputs another sequence, using **attention**.

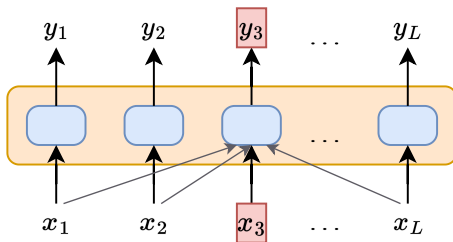


Figure 2: Each input is contextualized by the mean of **attention**.

## Attention

$$\begin{cases} \mathbf{q}_i &= \mathbf{Q}\mathbf{x}_i \in \mathbb{R}^d, \\ \mathbf{v}_j &= \mathbf{V}\mathbf{x}_j \in \mathbb{R}^d, \\ \mathbf{k}_j &= \mathbf{K}\mathbf{x}_j \in \mathbb{R}^d, \end{cases} \quad \begin{cases} s_{ij} &= \mathbf{q}_i^T \mathbf{k}_j \in \mathbb{R}, \quad 1 \leq j \leq L, \\ \alpha_i &= \text{Softmax}(\mathbf{s}_i) \in \mathbb{R}^L, \\ \mathbf{y}_i &= \sum_{j=1}^L \alpha_{ij} \mathbf{v}_j \in \mathbb{R}^d. \end{cases} \quad (1)$$

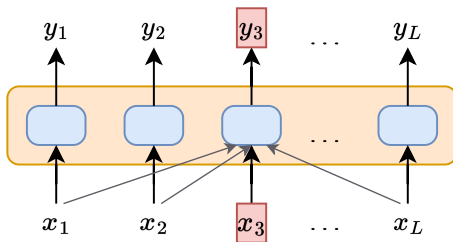


Figure 2: Each input is contextualized by the mean of **attention**.

## Attention

$$\begin{cases} \mathbf{q}_i &= \mathbf{Q}\mathbf{x}_i \in \mathbb{R}^d, \\ \mathbf{v}_j &= \mathbf{V}\mathbf{x}_j \in \mathbb{R}^d, \\ \mathbf{k}_j &= \mathbf{K}\mathbf{x}_j \in \mathbb{R}^d, \end{cases} \quad \begin{cases} s_{ij} &= \mathbf{q}_i^T \mathbf{k}_j \in \mathbb{R}, \quad 1 \leq j \leq L, \\ \alpha_i &= \text{Softmax}(\mathbf{s}_i) \in \mathbb{R}^L, \\ \mathbf{y}_i &= \sum_{j=1}^L \alpha_{ij} \mathbf{v}_j \in \mathbb{R}^d. \end{cases} \quad (1)$$

$\Rightarrow$  The output  $\mathbf{y}_i$  is simply a **non-negative weighted sum** of the input.

# Standard transformer model

Then we can build **deep networks** around this attention block.

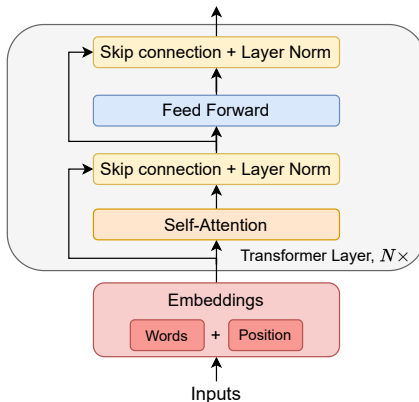


Figure 3: Standard stack of transformer layers.

# How should we use it in practice?

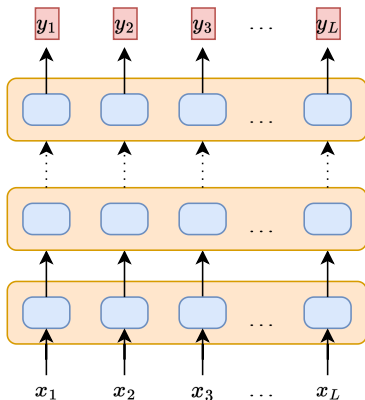


Figure 4:  $y_1, \dots, y_L \in \mathbb{R}^d$  are **deep contextualized words embeddings**.

**Challenge:** make these embeddings as expressive as possible.



# Subsequent questions

How did we obtain such impressive performances in NLP?

## Observations

- ① LLMs can perform well on almost all classical NLP tasks, despite **not having being trained for it**.
- ② LLMs **perform better than human annotators** on some tasks [6].
- ③ LLMs are **big**.

# Subsequent questions

How did we obtain such impressive performances in NLP?

## Observations

- ① LLMs can perform well on almost all classical NLP tasks, despite **not having being trained for it**.
- ② LLMs **perform better than human annotators** on some tasks [6].
- ③ LLMs are **big**.

## Conclusion

- 1. and 2. implies that the model did not use annotated data.
- 3. means that the model used a lot of data.

# Subsequent questions

How did we obtain such impressive performances in NLP?

## Observations

- ① LLMs can perform well on almost all classical NLP tasks, despite **not having being trained for it**.
- ② LLMs **perform better than human annotators** on some tasks [6].
- ③ LLMs are **big**.

## Conclusion

- 1. and 2. implies that the model did not use annotated data.
  - 3. means that the model used a lot of data.
- ⇒ LLMs work because they perform a lot of **self-supervised** learning.

# Table of Contents

## ① Reminders

- Attention
- Transformer layer
- Subsequent questions

## ② Pre-training

- Generalities on pre-training
- Pre-training for classification
- Word2Vec
- BERT

## ③ Generation

- Conditional language modeling
- Other architectures
- Inference

## ④ Conclusion

# Why pre-training?

**Huge expressive power** of transformers [10].  $\implies$  Leverage lots of data.

# Why pre-training?

**Huge expressive power** of transformers [10].  $\implies$  Leverage lots of data.

## Good news

There are lots of text data.

## Bad news

Few annotated data.

# Why pre-training?

**Huge expressive power** of transformers [10].  $\Rightarrow$  Leverage lots of data.

## Good news

There are lots of text data.

- Common Crawl [11] **250 billions web pages**.
- The Pile [5] **825GiB** of texts from diverse sources (web, books, professional resources, etc).

## Bad news

Few annotated data.

- GLUE (a very popular benchmark on text classification) is made of datasets between and **780 and 400K documents**.
- TriviaQA (answering questions about a given text) is made of **650K documents**.

# What is a good-pretraining?

We have a lot of un-annotated data  $\implies$  Self-supervised learning.



# What is a good-pretraining?

We have a lot of un-annotated data  $\implies$  Self-supervised learning.

**Challenge:** find a pre-training *close-enough* to the target tasks.

## Pre-training formulation

$$\mathcal{D} = \{x_i\}_i \xrightarrow{\mathcal{T}} \hat{\mathcal{D}} = \{\tilde{x}_j, \tilde{y}_j\}_j,$$

where  $\mathcal{T}$  is any transformation over a document.

The model is trained on a loss:

$$\min_{\theta} \mathcal{L}(\text{LLM}_{\theta}(\tilde{x}), \tilde{y}).$$

# Different pre-training

Since there are several tasks in NLP, there exists **different pre-training**. We will focus on the main ones:

- Classification,
- Generation.

⇒ All the most used models derive from one of those pre-trainings.

# Pre-training for classification

## Goal of text classification

Infer the **global meaning** of texts, through **words in their context**.

# Pre-training for classification

## Goal of text classification

Infer the **global meaning** of texts, through **words in their context**.

⇒ Emphasize words contextualization during pre-training!

# Pre-training for classification

## Goal of text classification

Infer the **global meaning** of texts, through **words in their context**.

⇒ Emphasize words contextualization during pre-training!

**You have already done it!**

# Word2Vec as a pre-training

You constructed  $\tilde{D}$  by extracting positive and negative context.

- $\tilde{x} = \begin{cases} (w, C^+) \\ \text{or} \\ (w, C^-). \end{cases}$
- $\tilde{y} = \begin{cases} 1 & \text{if } C^+, \\ 0 & \text{otherwise.} \end{cases}$

# Word2Vec as a pre-training

You constructed  $\tilde{D}$  by extracting positive and negative context.

- $\tilde{x} = \begin{cases} (w, C^+) \\ \text{or} \\ (w, C^-). \end{cases}$
- $\tilde{y} = \begin{cases} 1 & \text{if } C^+, \\ 0 & \text{otherwise.} \end{cases}$

Let's see how this can be pushed further with transformers.

# BERT – Introduction

Let's now dive into bigger experiments: **BERT** [1].

- BERT was an important milestone.
- Impressive performance that yields to the **massive adoption of transformers**.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Avg -
Pre-GPT SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
ELMO++	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT <sub>base</sub>	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
<b>BERT<sub>large</sub></b>	<b>86.7/85.9</b>	<b>72.1</b>	<b>92.7</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>82.1</b>

**Table 1:** Results on GLUE dataset, 9% of relative amelioration on average on an extremely competitive dataset.



# BERT – Introduction

Let's now dive into bigger experiments: **BERT** [1].

- BERT was an important milestone.
- Impressive performance that yields to the **massive adoption of transformers**.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Avg -
Pre-GPT SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
ELMO++	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT <sub>base</sub>	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
<b>BERT<sub>large</sub></b>	<b>86.7/85.9</b>	<b>72.1</b>	<b>92.7</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>82.1</b>

**Table 1:** Results on GLUE dataset, 9% of relative amelioration on average on an extremely competitive dataset.

⇒ Let's go through the **technical details**.

# BERT's pre-training

## BERT's Masked Language Modeling (MLM)

**Goal (Word2Vec++):** predict the *word given the context*.

# BERT's pre-training

## BERT's Masked Language Modeling (MLM)

**Goal (Word2Vec++):** predict the *word given the context*.

⇒ This is intuitively much harder than Word2Vec objective.

# BERT's pre-training

## BERT's Masked Language Modeling (MLM)

**Goal (Word2Vec++):** predict the *word given the context*.

⇒ This is intuitively much harder than Word2Vec objective.

- Delete randomly 15% of tokens in  $x$ .
- Predict the deleted tokens.

# BERT MLM illustration

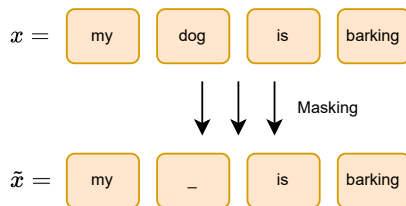


Figure 5: BERT masked language modeling.

Then we train the model to predict the right word:

$$\begin{aligned}\mathcal{L} &= -\log P_{\theta}(\text{dog} \mid \tilde{x}), \\ &= \sum_{\substack{w \in x \\ w \text{ is masked}}} -\log P_{\theta}(w \mid \tilde{x})\end{aligned}$$

## Other objectives?

That's it for the MLM pre-training objective.

## Other objectives?

That's it for the MLM pre-training objective.

**Why stop there?** We can stack several objectives!

## Other objectives?

That's it for the MLM pre-training objective.

**Why stop there?** We can stack several objectives!

Next sentence prediction.

**Intuition:** MLM operates at a token scale.

⇒ Enhance the model's global understanding with **next sentence prediction**.



# Next sentence prediction

- Extract a sentence  $s_1$  from a document  $x \in \mathcal{D}$ .
- With 50% chance, take  $s_2$  the sentence following  $s_1$ .
- With 50% chance, take  $s_2$  a random sequence from  $\mathcal{D}$ .

Simply penalize the model:

$$\mathcal{L} = -\mathbf{1}_{s_2 \text{ follows } s_1} \log P_{\theta}(s_1, s_2) - \mathbf{1}_{\text{random } s_2} \log(1 - P_{\theta}(s_1, s_2)) \quad (2)$$

# BERT in practice

Several questions arise in practice:

- How do we actually format the input?
- How do we indicate the model we deleted a word?
- How do we indicate a model what is the first sentence?

Let's visualize everything.

# BERT – Input embeddings MLM

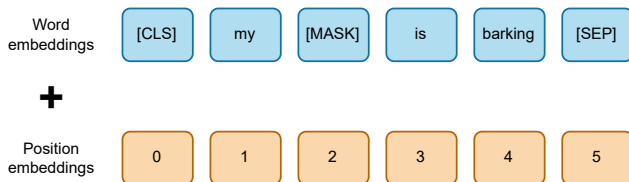


Figure 6: BERT input embeddings for MLM.

BERT uses **words and position embeddings**. There are **3 special tokens**.

- A special [MASK] token.
- A special token [CLS] that should retain sentence-level information.
- A special token indicating the end of the sequence [SEP].

# BERT – Next sentence prediction embeddings

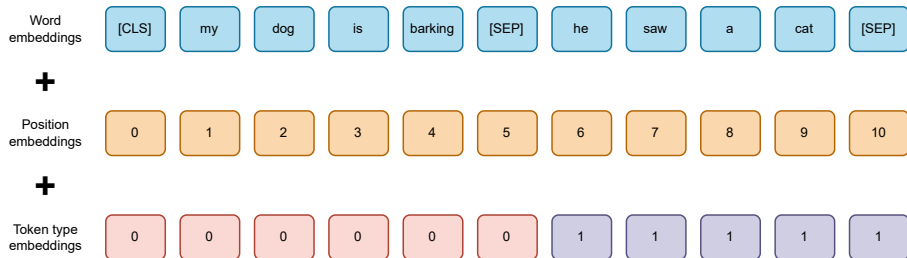


Figure 7: BERT input embeddings for next sentence prediction.

We use additional **token type** embeddings.

# Online demo

On Google Colab.

# Fine-tuning

Finetuning leverages internal representation as a backbone for classification.

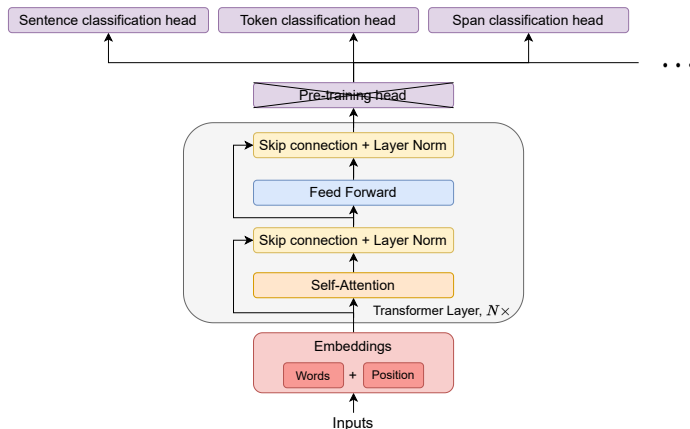


Figure 8: Switching from pretraining to finetuning.

Break!

# Table of Contents

## ① Reminders

- Attention
- Transformer layer
- Subsequent questions

## ② Pre-training

- Generalities on pre-training
- Pre-training for classification
- Word2Vec
- BERT

## ③ Generation

- Conditional language modeling
- Other architectures
- Inference

## ④ Conclusion



# Reminders

Generative models seek to maximize the log-likelihood over the dataset.

$$\max_{\theta} \sum_{x \in \mathcal{D}} \log P_{\theta}(x).$$

But, since we are dealing with sequences of words, defining  $P_{\theta}$  **should range over the whole set of sequences**, which is of cardinal  $|\mathcal{V}|^L$ .

When  $\mathcal{V} \approx 30K$  and  $L \approx 2K$ , this is **highly unfeasible**.

# Reminders

Generative models seek to maximize the log-likelihood over the dataset.

$$\max_{\theta} \sum_{x \in \mathcal{D}} \log P_{\theta}(x).$$

But, since we are dealing with sequences of words, defining  $P_{\theta}$  **should range over the whole set of sequences**, which is of cardinal  $|\mathcal{V}|^L$ .

When  $\mathcal{V} \approx 30K$  and  $L \approx 2K$ , this is **highly unfeasible**.

Instead, we choose to learn our probability distribution over the factorized form:

$$P_{\theta}(x) = \prod_{i=1}^L P_{\theta}(x_i \mid x_{<i}).$$

# Generation

$$\log P_{\theta}(x) = \sum_{i=1}^L \log P_{\theta}(x_i \mid x_{<i}).$$

⇒ The model learns to **predict the next tokens** given the previous ones.

This is clearly an **unsupervised** pre-training objective.

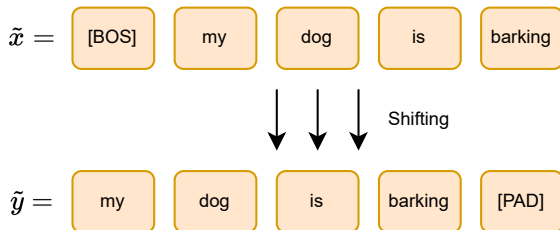


Figure 9: Generative models pre-training.

# Architecture for generation

Can we still use the same bidirectional architecture?

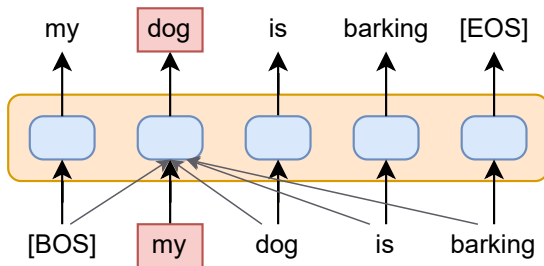


Figure 10: Bidirectional architecture.

# Architecture for generation

**Answer:** No!

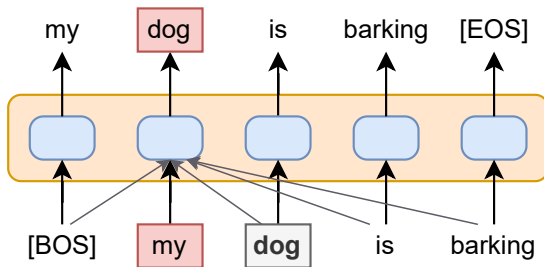


Figure 11: The transformer has access to dog to predict dog.

⇒ We need another architecture.

# Decoder

## Attention in BERT

$$\begin{cases} s_{ij} &= \mathbf{q}_i^T \mathbf{k}_j \in \mathbb{R}, \quad 1 \leq j \leq L, \\ \alpha_i &= \text{Softmax}(\mathbf{s}_i) \in \mathbb{R}^L, \\ \mathbf{y}_i &= \sum_{j=1}^L \alpha_{ij} \mathbf{v}_j \in \mathbb{R}^d. \end{cases}$$

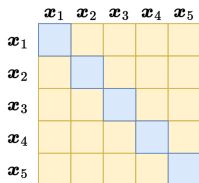


Figure 12: **Bidirectional** attention, tokens attend to every token.

## Attention for generation

$$\begin{cases} s_{ij} &= \mathbf{q}_i^T \mathbf{k}_j \in \mathbb{R}, \quad 1 \leq j \leq i, \\ \alpha_i &= \text{Softmax}(\mathbf{s}_i) \in \mathbb{R}^i, \\ \mathbf{y}_i &= \sum_{j=1}^i \alpha_{ij} \mathbf{v}_j \in \mathbb{R}^d. \end{cases}$$

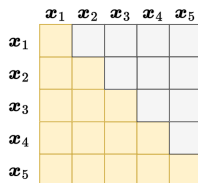


Figure 13: **Unidirectional** attention, tokens can only attend backward.

# Decoder architecture

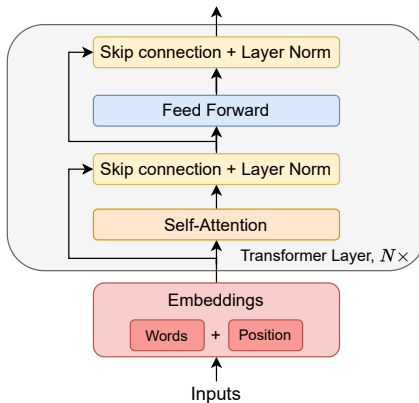


Figure 14: Decoder architecture.

# Well-known decoder architecture

- GPT2 [3], GPT3 [4]
- LLama [9], LLama-2 [8]
- etc.

⇒ Demo!



# Encoder-decoder

We saw bidirectional-**encoder** and causal **decoder**.

# Encoder-decoder

We saw bidirectional-**encoder** and causal **decoder**.

Why don't use both?

**Encoder-decoder** models.

# Encoder-decoder

We saw bidirectional-**encoder** and causal **decoder**.

Why don't use both?

**Encoder-decoder** models.

## Encoder-decoder

Architectures tailored for conditional generation tasks.

- Give full access to  $x$ .
- Causal generation for  $y$ .

$$P_{\theta}(y_i \mid y_{<i}, x).$$

# Conditional generation

## Conditional generation

Framing a problem as a condition generation task might be useful in a lot of tasks:

- translating text,
- summarizing a news article,
- answering a question over a text,
- etc.

# Conditional generation

## Conditional generation

Framing a problem as a condition generation task might be useful in a lot of tasks:

- translating text,
- summarizing a news article,
- answering a question over a text,
- etc.

⇒ It makes sense to give the model full access to a source  $x$  and generate the answer  $y$  based on this input  $x$ .

# Encoder-decoder architecture

## Encoder

$x$  is encoded through a **bidirectional** transformer (BERT-like).

## Decoder

$y$  is processed through a **causal** transformer (GPT-like).

⇒ How do the two communicate?

# Encoder-decoder illustration

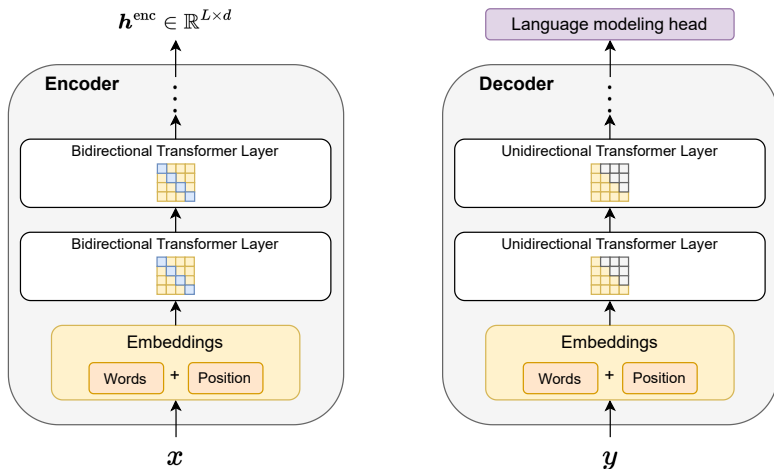


Figure 15: An encoder and a decoder.

# Encoder-decoder illustration

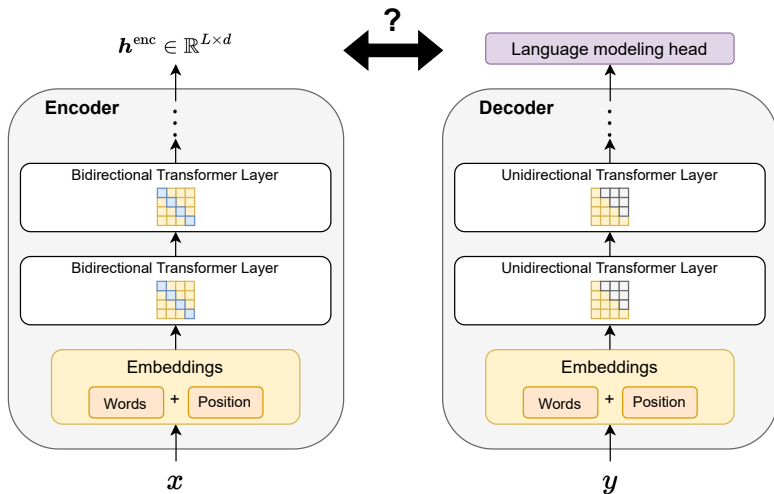


Figure 15: How can an encoder and a decoder communicate?



# Cross-attention

## Cross-attention: contextualization with the encoder output

Instead of computing the similarity of a token  $\mathbf{x}_i$  with the other tokens  $(\mathbf{x}_j)_{j \neq i}$ , we compute the **similarity with the output of the encoder**:

$$\begin{cases} \mathbf{q}_i = \mathbf{Q}\mathbf{x}_i, \\ \mathbf{v}_j = \mathbf{V}\mathbf{h}_j^{\text{enc}}, \\ \mathbf{k}_j = \boxed{\mathbf{K}\mathbf{h}_j^{\text{enc}}}, \end{cases} \quad \begin{cases} s_{ij} = \mathbf{q}_i^T \mathbf{k}_j \in \mathbb{R}, \quad 1 \leq j \leq L, \\ \alpha_i = \text{Softmax}(\mathbf{s}_i) \in \mathbb{R}^i, \\ \mathbf{y}_i = \sum_{j=1}^L \alpha_{ij} \mathbf{v}_j \in \mathbb{R}^d. \end{cases}$$

	$h_1$	$h_2$	$h_5$	$h_4$	$h_6$	$h_7$	$h_8$
$\mathbf{x}_1$							
$\mathbf{x}_2$							
$\mathbf{x}_3$							
$\mathbf{x}_4$							
$\mathbf{x}_5$							

Figure 16: Tokens in the decoder attend to tokens from the encoder output.

# Encoder-decoder with cross-attention

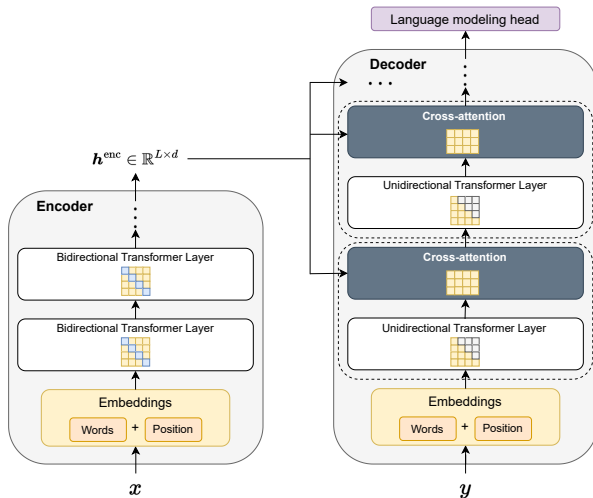


Figure 17: Encoder-decoder model with cross-attention layers.

# Decoder layer for an encoder-decoder model

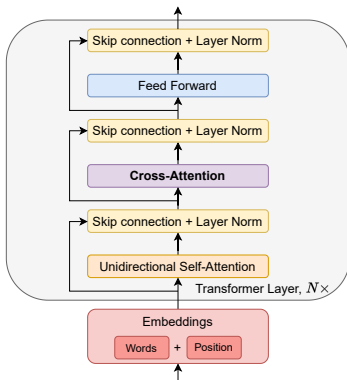


Figure 18: Decoder layer for an encoder-decoder model.

The decoder layers are extended with a **cross-attention** block. The encoder layers are not modified compared to encoder-only models (BERT).

# Pre-training for encoder-decoders

Encoder-decoders have been used broadly in:

- Machine **translation** (state-of-the-art in the domain) [10],
- Text **summarization** (state-of-the-art also according to some evaluation) [2],
- Question answering [7],
- etc.

Except for machine translation, they relied on a pre-training.

# BART

**BART** [2] is one of the most well-known encoder-decoder. For pre-training, the authors proposed a **denoising objective**.

## BART's denoising objective

Several corruptions are made on the original text, and the goal is to retrieve the original one. It can be seen as a **generalization of BERT**.

- Span masking,
- token deletion,
- sentence permutation.

# BART on tokens infilling

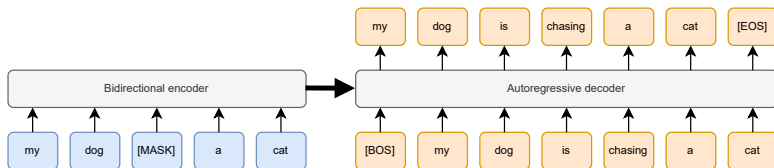


Figure 19: Denoising with BART.

# Other models

Following BERT and BART, many papers proposed new pre-training objectives. To mention some of them:

- ELECTRA,
- ROBERTA,
- DEBERTA,
- T5,
- Pegasus,
- etc.

They all have their specificities but rely on the same ideas than BERT and BART.

# Inference

Throughout the lessons we talked a lot about **training**.

But what about **inference**?

In the following part we are going to discuss the potential subtelties of inference in NLP.



# Inference in classification

## At train time

Classification models are trained with the MLE objective, i.e. they maximize  $P_{\theta}(y = \text{class} \mid x)$ .

# Inference in classification

## At train time

Classification models are trained with the MLE objective, i.e. they maximize  $P_{\theta}(y = \text{class} \mid x)$ .

## At inference time

For an input  $x$ , the model gives a probability distribution over the classes  $P_{\theta}(\cdot \mid x)$ .

Then you fix a decision rule, usually:

$$\hat{y} = \arg \max_y P_{\theta}(y \mid x). \quad (3)$$

# Inference in classification

## At train time

Classification models are trained with the MLE objective, i.e. they maximize  $P_{\theta}(y = \text{class} \mid x)$ .

## At inference time

For an input  $x$ , the model gives a probability distribution over the classes  $P_{\theta}(\cdot \mid x)$ .

Then you fix a decision rule, usually:

$$\hat{y} = \arg \max_y P_{\theta}(y \mid x). \quad (3)$$

$\implies$  For classification models, i.e., **encoders** in NLP, we do the same.

# Inference in generation

## At train time

For generative models, we maximize instead the factorized density:

$$\prod_{i=1}^L P_{\theta}(y_i = w_i \mid w_{<i}).$$

# Inference in generation

## At train time

For generative models, we maximize instead the factorized density:

$$\prod_{i=1}^L P_{\theta}(y_i = w_i \mid w_{<i}).$$

## At inference time

Is the following decision rule still a good choice?

$$\hat{y} = \arg \max_y P_{\theta}(y \mid x) = \arg \max_{y_1, \dots, y_L} \prod_{i=1}^L P_{\theta}(y_i \mid y_{<i}).$$

# Inference in generation

## At train time

For generative models, we maximize instead the factorized density:

$$\prod_{i=1}^L P_{\theta}(y_i = w_i \mid w_{<i}).$$

## At inference time

Is the following decision rule still a good choice?

$$\hat{y} = \arg \max_y P_{\theta}(y \mid x) = \arg \max_{y_1, \dots, y_L} \prod_{i=1}^L P_{\theta}(y_i \mid y_{<i}).$$

**No!** We are taking the arg max over  $|\mathcal{V}|^L$  combinations, highly **intractable**.

# Inference in generation

Workaround: approximate  $\arg \max_{y_1, \dots, y_L} \prod_{i=1}^L P_{\theta}(y_i \mid y_{<i})$  with a **greedy algorithm**.

Simply:

- $\hat{y}_1 = \arg \max_{y_1} P_{\theta}(y_1),$
- $\hat{y}_2 = \arg \max_{y_2} P_{\theta}(y_2 \mid \hat{y}_1),$
- $\vdots$
- $\hat{y}_i = \arg \max_{y_i} P_{\theta}(y_i \mid \hat{y}_{<i}).$

$\implies$  At each step, the  $\arg \max$  is only performed over  $|\mathcal{V}|$  possibilities.

# Table of Contents

## ① Reminders

- Attention
- Transformer layer
- Subsequent questions

## ② Pre-training

- Generalities on pre-training
- Pre-training for classification
- Word2Vec
- BERT

## ③ Generation

- Conditional language modeling
- Other architectures
- Inference

## ④ Conclusion



# Conclusion

Thank you!

# References I

- [1] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. [arXiv: 1810.04805](#) [cs.CL].
- [2] Mike Lewis et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. [arXiv: 1910.13461](#) [cs.CL].
- [3] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).
- [4] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. [arXiv: 2005.14165](#) [cs.CL].
- [5] Leo Gao et al. *The Pile: An 800GB Dataset of Diverse Text for Language Modeling*. 2020. [arXiv: 2101.00027](#) [cs.CL].

## References II

- [6] Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. “ChatGPT outperforms crowd workers for text-annotation tasks”. In: *Proceedings of the National Academy of Sciences* 120.30 (July 2023). DOI: [10.1073/pnas.2305016120](https://doi.org/10.1073/pnas.2305016120). URL: <https://doi.org/10.1073%2Fpnas.2305016120>.
- [7] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: 1910.10683 [cs.LG].
- [8] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].
- [9] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].
- [10] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

## References III

- [11] Common Crawl. *Common Crawl website*. URL: <https://commoncrawl.org/>.