

Modèles de langage basés sur les *Transformers*

Résumé pour projet ML CS 433

Novembre 2022

1. Introduction

Tâches typiques sur les données textuelles

- Classifier des emails, des articles, des tweets, etc.
- Résumer un ou plusieurs documents, p.ex. des articles de presse
- Répondre à une question sur la base d'un ou plusieurs documents
- Traduire d'une langue à une autre
- Décider si une phrase implique ou contredit une autre (inférence)

Modèle de langage = prédire le mot suivant

- Exemples

- ❖ La HEIG-VD est une école du canton de ____
- ❖ Jean a réussi l' ____
- ❖ La méthode de calcul ____

- Objectif mathématique

- estimer pour chaque mot du vocabulaire la probabilité qu'il soit le mot suivant
 - c'est une distribution de probabilité $P_{\text{modèle}}$ sur l'ensemble du vocabulaire
 - équivaut à estimer la probabilité d'une phrase

- Fonction de coût

- entropie croisée : $-\log(P_{\text{modèle}}(\text{mot}_{\text{correct}}))$

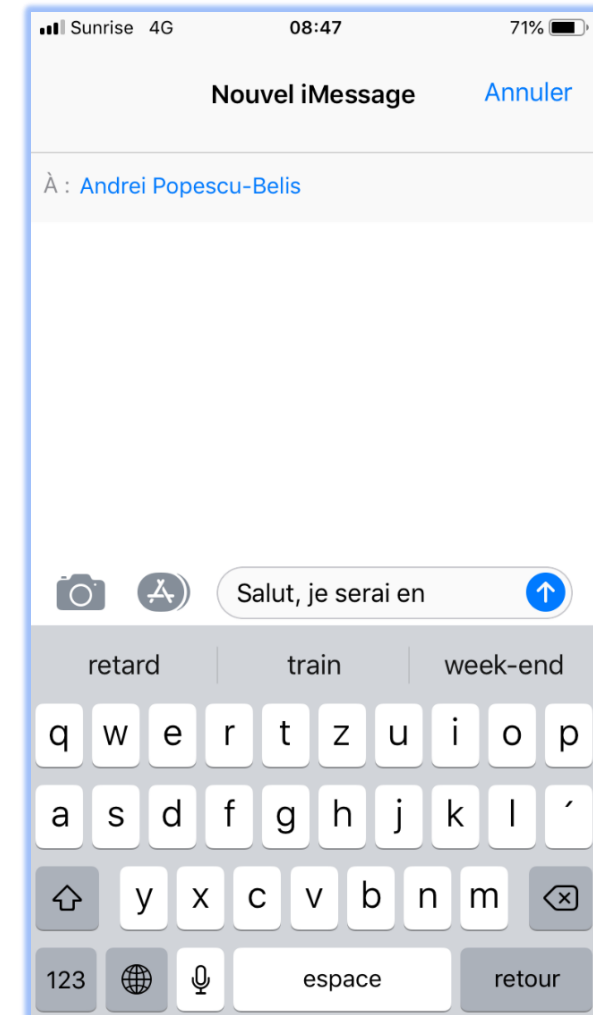
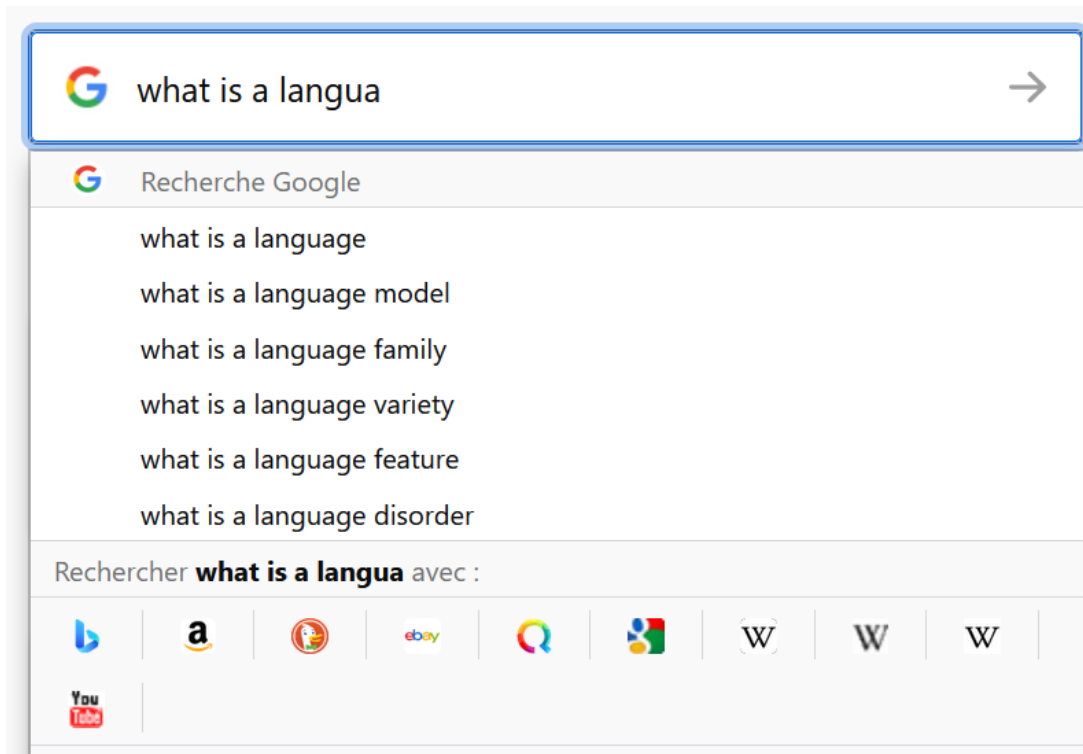
- Supervision

- textes réels : on en connaît tous les mots
- données abondantes, faciles à trouver
- aucune annotation n'est nécessaire

- Le modèle (ou système) obtenu s'appelle un **modèle de langage**

- **Utilité ?** → Saisie de mots, correction orthographique, OCR, reconnaissance vocale, ...
- **Difficulté ?** → Peut nécessiter une compréhension très approfondie.
- **Implémentation ?** → Règles linguistiques, statistiques sur les n-grammes, ou **réseaux de neurones**

Autocompletion uses language models



Goal of language models (LMs)

1. Intuitive definition

- given a word sequence, a language model predicts the **most likely next word**

2. Principled definition

- a language model computes the **probability of any sequence of words**

• Use of a language model

- subcomponent of many NLP tasks
 - rank candidates from a text generation process based on the likelihood of their word sequence
- benchmark task in its own right
 - demonstrate the level of text “understanding”

• NLP tasks where LMs are used

- predictive typing (autocompletion)
 - ✓ $P(\text{"my cat eats fish"}) > P(\text{"my cat eats dish"})$
- spelling and grammar correction
- speech recognition
 - ✓ $P(\text{"recognize speech"}) > P(\text{"wreck a nice beach"})$
- OCR and handwriting recognition
- authorship identification
- machine translation
 - ✓ $P(\text{"the red house"}) > P(\text{"the house red"})$
- summarization
- dialogue
- information retrieval

Formal definition

- LMs are models/tools that either
 - compute the probability of a sequence **or** compute the probability of next word

$$P(w_1, w_2, \dots, w_n) \text{ or } P(w_n | w_1, w_2, \dots, w_{n-1})$$

- The two formulations are equivalent because

$$P(w_n | w_1, \dots, w_{n-1}) = P(w_1, \dots, w_n) / P(w_1, \dots, w_{n-1})$$

- Therefore :

$$\begin{aligned} P(w^{(1)}, \dots, w^{(T)}) &= P(w^{(1)}) \times P(w^{(2)} | w^{(1)}) \times \dots \times P(w^{(T)} | w^{(T-1)}, \dots, w^{(1)}) \\ &= \prod_{t=1}^T P(w^{(t)} | w^{(t-1)}, \dots, w^{(1)}) \end{aligned}$$

← according to the Chain Rule

Evaluating language models

1. Indirectly: how much does a LM help a task?

- complex assessment, depends on task

2. Directly: “perplexity”

- given a sentence, compute $P_{\text{LM}}(w_1, \dots, w_n)$
➤ the higher, the better
- use cross-entropy over a sentence or text to compute average perplexity

$$\begin{aligned} H(P_{\text{LM}}) &= -\log(P_{\text{LM}}(w_1, \dots, w_n)) / n \\ &= -\sum_{i=1..n} \log(P_{\text{LM}}(w_i | w_1, \dots, w_{i-1})) / n \end{aligned}$$

- the perplexity of a LM for the text is $2^{H(P_{\text{LM}})}$
➤ the lower, the better

- Perplexity is the probability of the test set, normalized by the number of words

$$P(w^{(1)}, w^{(2)} \dots w^{(N)})^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w^{(1)}, w^{(2)} \dots w^{(N)})}}$$

$$PP(W) = \sqrt[N]{\prod_{t=1}^N \frac{1}{P(w^{(t)} | w^{(t-1)} \dots w^{(1)})}}$$

$$PP(W) = \sqrt[N]{\prod_{t=1}^N \frac{1}{P(w^{(t)} | w^{(t-1)})}}$$

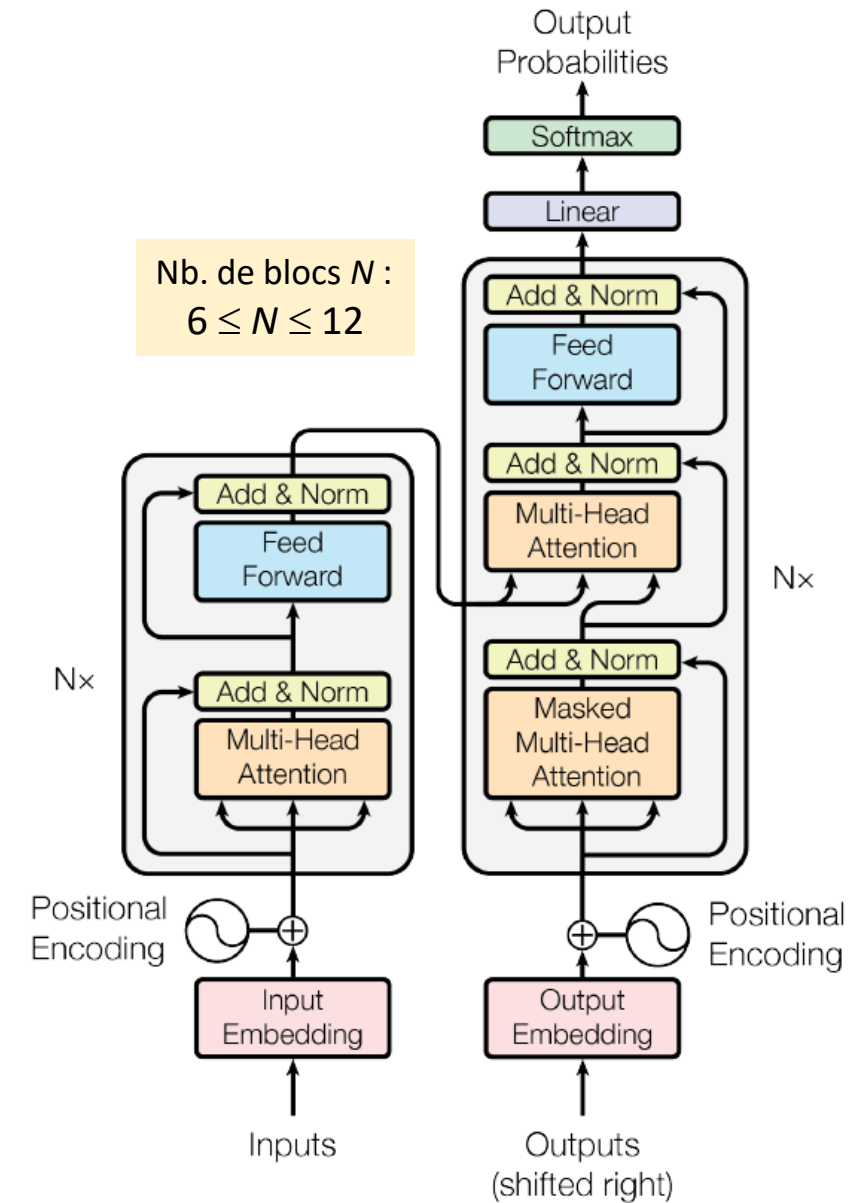
2. Le système *Transformer* et les blocs (ou couches) *Transformer*

Qu'est-ce que le *Transformer* ?

- Peut-on traiter ensemble les mots d'une phrase et obtenir pour chacun d'eux un *embedding* (vecteur en basse dimension) qui reflète son sens dans la phrase ?
 - partir des *embeddings* non-contextualisés et les transformer peu à peu en fonction des autres mots de la phrase \Rightarrow nouvelle architecture, nouvel entraînement, nouvelles tâches
- À l'origine, système complet proposé en 2017 par Google : **encodeur \rightarrow décodeur**
 - Vaswani A. et al., [*Attention is All You Need*](#) (publié en 2018) : traduction automatique
 - dans ce système, chaque partie est constituée de 6 ou 12 blocs *Transformer*
 - blocs « **décodeurs** » : construisent une phrase de gauche à droite (\sim modèle de langage)
 - blocs « **encodeurs** » : construisent une représentation d'une phrase donnée intégralement

Le système *Transformer*

- Réseau de neurones constitué d'une partie encodeur connectée à une partie décodeur
 - proposé à l'origine pour la traduction automatique
- Encodeur :
phrase à traduire → représentation abstraite
- Décodeur : *phrase traduite jusqu'à la position $i - 1$ → probabilité du mot en position i*
- Entraînement supervisé sur des phrases et leurs traductions

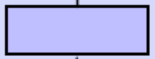


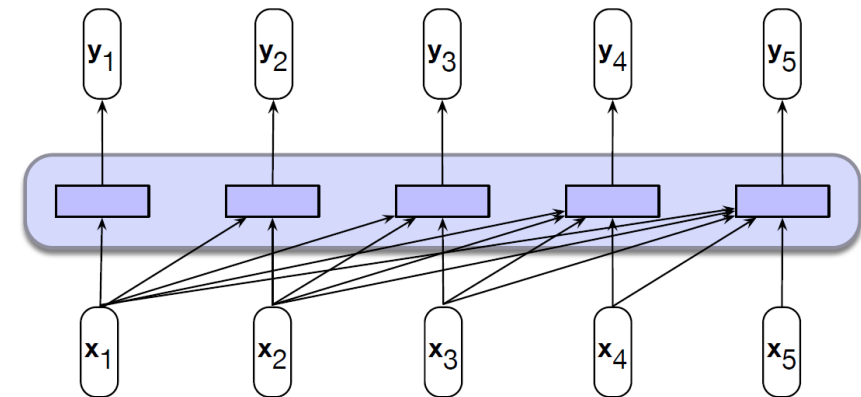
Originalité du bloc *Transformer* : la couche attentionnelle

- But : modifier la représentation (*embedding*) de chaque mot en fonction de celles des autres
- Vecteur entrant du $i^{\text{ème}}$ mot = x_i
- Couche (poids) = matrices W_q , W_k et W_v (appries pendant l'entraînement)
- Le vecteur x_i est transformé ainsi
 - vecteur requête (*query* $q_i = W_q \cdot x_i$) | vecteur clé (*key* $k_i = W_k \cdot x_i$) | vecteur valeur (*value* $v_i = W_v \cdot x_i$)
 - attention du $i^{\text{ème}}$ mot envers le $j^{\text{ème}}$ mot : $s_j = q_i \cdot k_j$ (*query* · *key*) puis pondérée & softmax $\Rightarrow a_j$
- Vecteur sortant, pour le $i^{\text{ème}}$ mot : $z_i = \sum_k a_k \cdot v_k$
- Si on regroupe tous les vecteurs dans des matrices, alors l'*output* est : $Z = \text{softmax}(Q \times K^T / \sqrt{d_k}) V$

3. Modèles de langage basés sur les décodeurs avec *Transformers*

Le bloc (couche) Transformer, version décodeur

- Le bloc Transformer est un réseau de neurones qui traite ensemble les *embeddings* des mots de la phrase (ici : x_1, x_2, \dots, x_5) et produit pour chaque mot un autre *embedding* (respectivement, ici : y_1, y_2, \dots, y_5) qui dépend du mot et des mots précédents (ici : y_3 dépend de x_1, x_2 et x_3)
 - dimensions fréquentes des *embeddings* (couche cachée) : 512, 768, 1024, etc.
- Attention !
 - les 5 **rectangles**  sont en réalité le *même* réseau Transformer
 - en réalité, la donnée d'entrée est une seule matrice dont les lignes sont les *embeddings* des mots

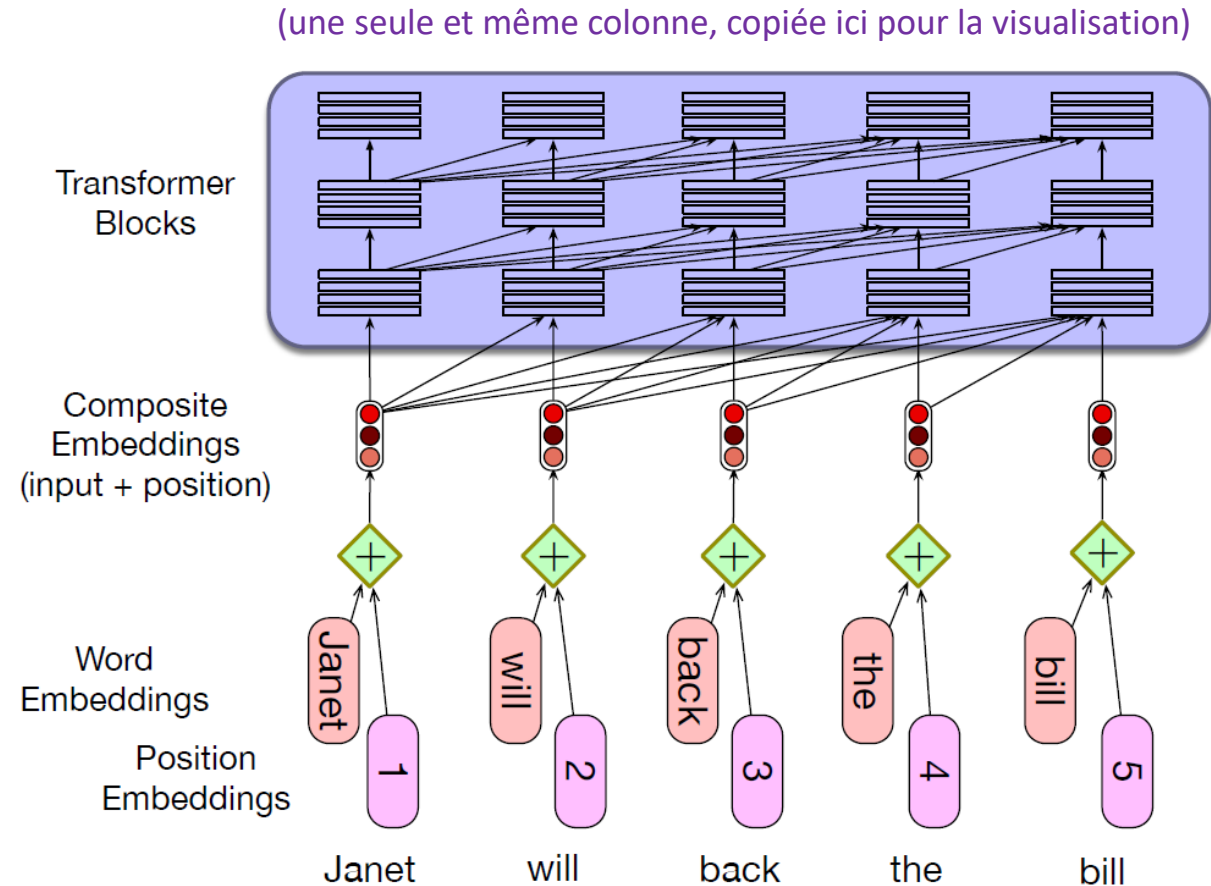


Source : [Jurafsky & Martin, SLP3](#), Ch. 9, Fig. 15

Les positions sont encodées aussi et ajoutées aux *embeddings*. Plusieurs blocs sont empilés.

- Pour transformer de manière significative les vecteurs, on empile entre 6 et 12 blocs Transformer

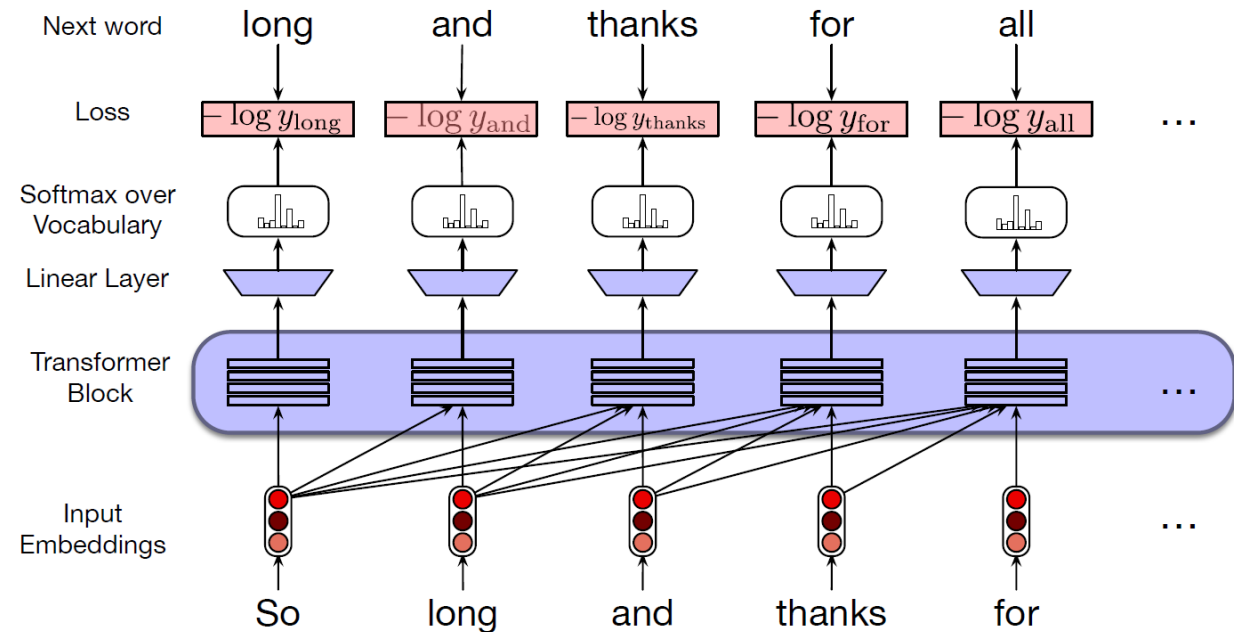
- Chaque bloc contient
 - *multi-head self-attention layer*
 - *residual connections and layer normalization*
 - *feedforward layer*
 - *residual connections and layer normalization*



(lignes différentes,
entre 6 et 12)

Entraînement du décodeur avec *Transformer* sur la tâche de prédiction de mots (auto-supervisé)

- En sortie, on retransforme chaque *embedding* en un vecteur de la taille du vocabulaire (30-50k)
- Avec *softmax*, on obtient une distribution de probabilité sur le vocabulaire
 - *testing* → c'est le mot prédit pour la suite (modèle de lg.)
 - *training* → fonction de coût = entropie croisée
= $-\log(P_{\text{modèle}}(\text{mot}_{\text{correct}}))$



⇒ Donc le « plongement contextualisé final » de chaque mot est en fait entraîné pour générer le mot suivant !

Exemples de modèles décodeurs et leurs tâches

- Modèles

- la famille GPT (*Generative Pretrained Transformer*) de l'institut OpenAI

GPT : "[Improving Language Understanding by Generative Pre-Training](#)", A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, June 2018

GPT-2 : "[Language Models are Unsupervised Multitask Learners](#)", by A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, February 2019 (et aussi "[Better Language Models](#)", blog OpenAI)

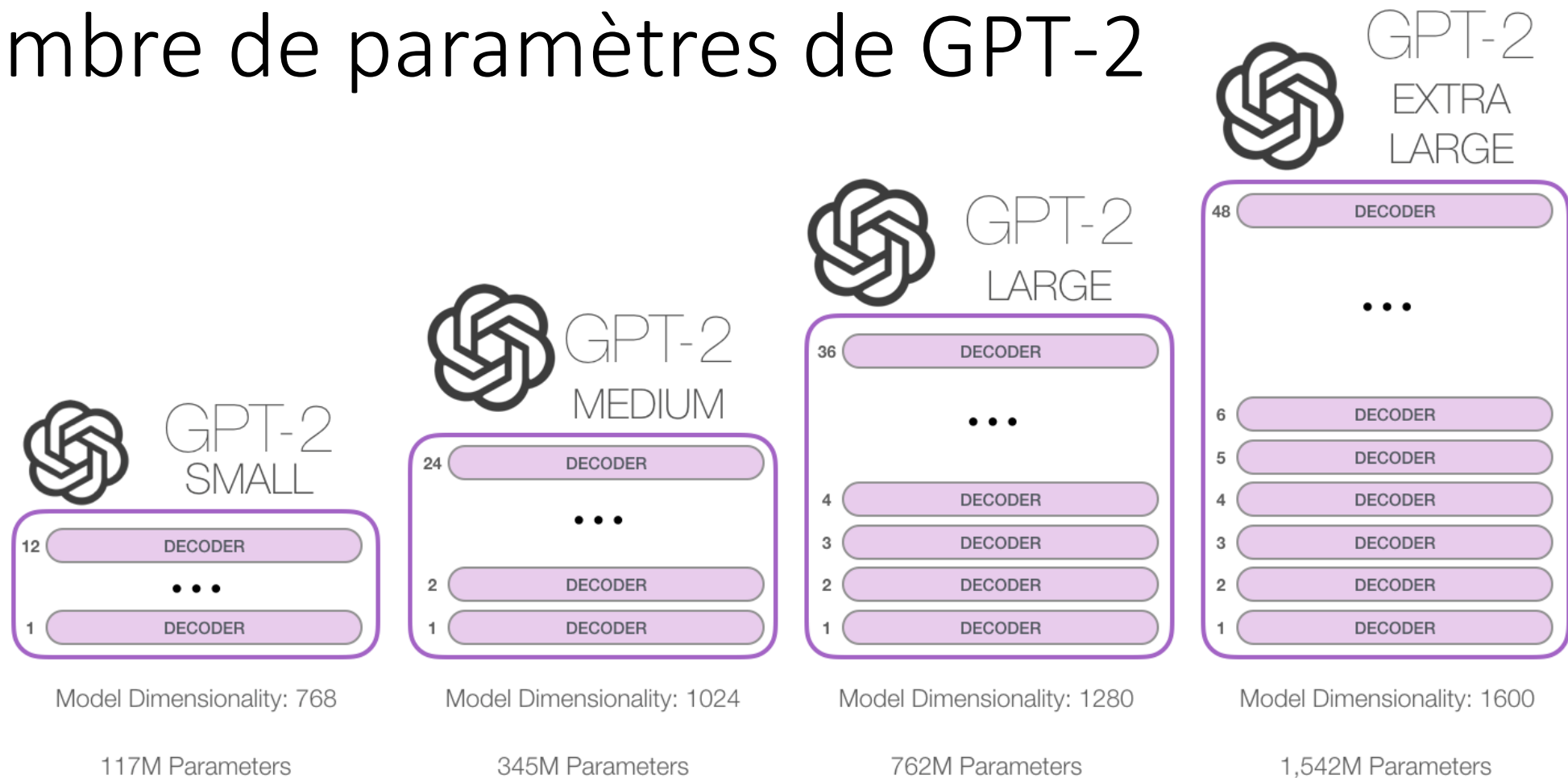
GPT-3 : "[Language Models are Few-Shot Learners](#)", by Tom B. Brown and many others (72 pages), July 2020

- Google introduit le [Transformer](#) (encodeur + décodeur) mi-2017, puis présente déjà un [modèle décodeur](#) début 2018 pour résumer des textes, mais ne poursuit pas et se concentre sur [l'encodeur pré-entraîné BERT](#) présenté fin 2018

- Tâches (*elles diffèrent quelque peu selon les modèles*)

- sans supervision = sans autre entraînement que le *pretraining*
 - génération de textes à partir d'un segment initial, réponses aux questions à partir d'un texte
 - nombreuses tâches spécifiées seulement par quelques exemples (*few-shot*), réponses aux questions (sans contexte)
 - supervision faible avec un jeu de données annotées
 - génération de résumés
 - supervision faible et ajout d'une couche linéaire + softmax sur le dernier *token* en sortie
 - classification, similarité, implication logique

Nombre de paramètres de GPT-2



- GPT-2 a été entraîné sur 40 Go de textes
 - corpus WebText construit par OpenAI à partir de 8 millions de pages web indiquées par Reddit
- Stockage
 - 'small' → ~500 Mo
 - 'extra large' → ~6.5 Go

Articles générés par GPT-3 sur la base d'un titre et d'un sous-titre donnés

Exemple difficile à distinguer d'un vrai article (12% de chances)

Title: United Methodists Agree to Historic Split
 Subtitle: Those who oppose gay marriage will form their own denomination
 Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.

The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Exemple facile à distinguer d'un vrai article (61% de chances)

Title: Star's Tux Promise Draws Megyn Kelly's Sarcasm
 Subtitle: Joaquin Phoenix pledged to not change for each awards event
 Article: A year ago, Joaquin Phoenix made headlines when he appeared on the red carpet at the Golden Globes wearing a tuxedo with a paper bag over his head that read, "I am a shape-shifter. I can't change the world. I can only change myself." It was a promise to not change to fit into the Hollywood mold: "I think that's a really special thing, to not change yourself. I think it's a really special thing to say, 'This is what's inside of me, I'm proud of it, and I'm not going to be ashamed because of the way that someone else thinks I should be.'" Now, it's the Oscars, and Phoenix is at it again. But this time, his publicist is saying he'll be wearing a tux no matter what.

Megyn Kelly was not impressed, and she let him have it on The Tonight Show. "You know, I feel like, I feel like you could have worn the tux," she says. "But you're saying you're a shape-shifter. I don't know if you can change your tux, but you can change your mind. You can change your mind. You can change your mind." Phoenix says he did, but it didn't stick. "I was like, 'Okay, I'm going to wear a tuxedo to this thing.' And then I thought, 'I don't want to wear a tuxedo to this thing.'" Kelly goes on to encourage him to change his mind again, but Phoenix says it's too late: "I'm committed to wearing this."

Voir aussi la démo de HuggingFace avec GPT-2 : <https://transformer.huggingface.co>


Andrei Popescu-Belis

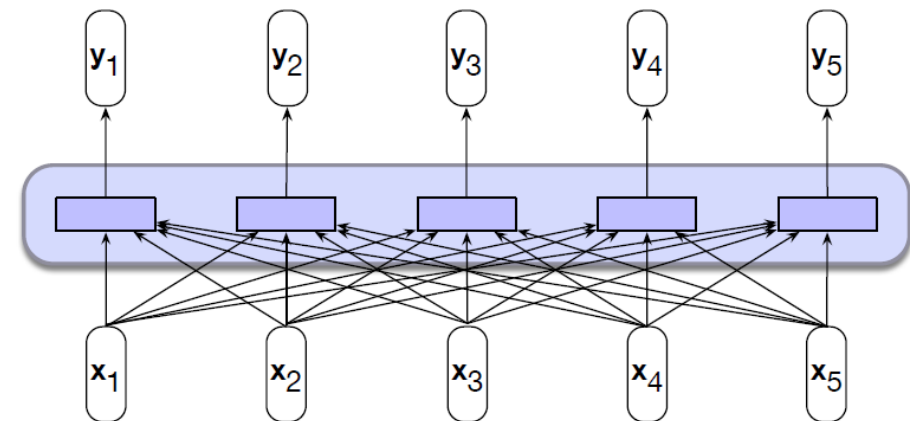


Write With Transformer
 Get a modern neural network to
 auto-complete your thoughts.

4. Modèles de langage basés sur les encodeurs avec *Transformers*

Le Transformer, version encodeur

- L'encodeur basé sur le Transformer est un réseau de neurones qui traite ensemble les *embeddings* des mots de la phrase (ici : x_1, x_2, \dots, x_5) et produit pour chaque mot un autre *embedding* (respectivement, ici : y_1, y_2, \dots, y_5) qui dépend du mot lui-même et de tous les autres mots (ici : y_3 dépend de x_1, \dots, x_5)
- Attention !
 - les 5 **rectangles**  sont en réalité le *même* réseau Transformer
 - en réalité, la donnée d'entrée est une seule matrice dont les lignes sont les *embeddings* des mots



Source : [Jurafsky & Martin, SLP3](#), Ch. 11, Fig. 2



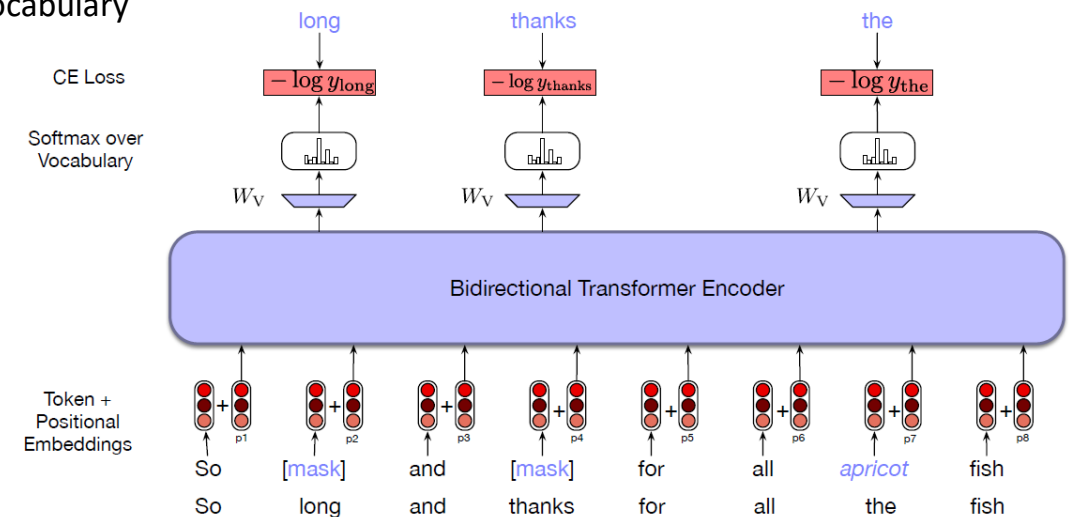
Source : [YouTube](#) via [Lil'log](#)

BERT : Bidirectional Encoder Representations from Transformers (Devlin et al. [Google] 2018)

- Comment entraîner la partie encodeur du *Transformer*?
 - **difficulté** : le réseau a accès à tous les mots, donc prédire le mot suivant est trivial
 - **solution** : essayer de prédire des mots qui sont masqués au hasard
- Originalité de l'approche BERT : séparer *pretraining* et *fine-tuning*
 1. **Pré-entraînement auto-supervisé**, avec seulement du texte
 - étape très coûteuse en calculs, mais à faire une seule fois
 2. **Adaptation supervisée à une tâche** avec des données spécifiques
 - nécessite une couche de sortie adaptée à la tâche
- Le modèle pré-entraîné construit des représentations contextuelles (*embeddings*) des mots d'une phrase, mais qui ne sont pas adaptées à une tâche
 - en pratique : **adapter des modèles publics à diverses tâches**, avec peu de calculs

Masked language modeling = 1^{ère} tâche d'entraînement auto-supervisé

- Principe : prédire des *tokens* qui ont été cachés dans une phrase d'entrée
 - remplacés par [MASK] (12%, au hasard) ou par un autre *token* du vocabulaire (1.5%)
- Méthode :
 - passer la représentation de [MASK] qui sort du réseau par une couche linéaire de dimension d'entrée $dim_{\text{embedding}}$ et de sortie $dim_{\text{vocabulary}}$
 - après application de *softmax*, on obtient une distribution de probabilité sur le vocabulaire
- Entraînement (*backpropagation*) : maximiser la probabilité du *token* caché
 - fonction de coût : $-\log(P_{\text{modèle}}(\text{token}_{\text{correct}}))$



Source : [Jurafsky & Martin, SLP3](#), Ch. 11, Fig. 5

Que peut faire un modèle BERT pré-entraîné ?

- Le modèle peut certes « deviner » des mots cachés (*Cloze test*)
 - mais ce n'est pas une tâche très utile...
- Le modèle peut fournir des représentations contextualisées (*embeddings*) des mots d'une phrase
 - mais il vaut mieux les « adapter » à une tâche = *fine-tuning*
- Comment adapter le modèle à une tâche ?
 1. Ajout d'une ou plusieurs couches en sortie
 2. Entraînement supervisé avec des données
 - le volume de données et le temps d'entraînement nécessaires sont beaucoup plus faibles que celui du pré-entraînement

Selon la tâche envisagée

RoBERTa : *Robustly optimized BERT approach* (Liu, et al. 2019)

- Méthode alternative pour entraîner le modèle BERT
 - en partant de l'observation que le modèle BERT d'origine est sous-entraîné
- Modifications
 - suppression de la tâche de détection de phrases consécutives (NSP)
 - corpus d'entraînement plus grand, taille de *batch* plus grande
 - entraînement plus long, sur des séquences de plusieurs phrases
 - *tokenisation* avec *Byte-level BPE* (comme GPT-2)
 - changement des mots masqués entre les époques
 - puisqu'on voit plusieurs fois chaque phrase à l'entraînement, il vaut mieux éviter d'avoir les mêmes [MASK] à chaque fois
 - 40 époques \Rightarrow 10 masquages différents
- Autre alternative : SpanBERT (masquer des plages de *tokens*)

Où trouver des modèles pré-entraînés ?

- BERT ou ses variants (RoBERTa, SpanBERT, etc.), monolingues (en diverses langues, p.ex. CamemBERT en français) ou multilingues (p.ex. mBERT)
 - disponibles en version pré-entraînée, ou alors déjà adaptée à une tâche (jeu de données)
- La société [HuggingFace](https://huggingface.co) fournit un *repository* très populaire pour ces modèles
 - Model Hub: <https://huggingface.co/models> (versions pour PyTorch, TensorFlow, etc.)
 - Transformers : librairie pour les utiliser <https://github.com/huggingface/transformers>
- La société a développé aussi des modèles allégés (*distilled*) : moins de paramètres (p.ex. 50%) mais scores très proches des modèles originaux



5. Conclusion

Conclusion

The Autoregressive (AR) models such as GPT and autoencoder (AE) models such as BERT are two most common ways for language modeling. However, each has their own disadvantages: AR does not learn the bidirectional context, which is needed by downstream tasks like reading comprehension, and AE assumes masked positions are independent given all other unmasked tokens, which oversimplifies the long context dependency.

<https://lilianweng.github.io/posts/2019-01-31-lm>