

Le traitement du langage naturel par transformers illustré par un exemple pour la classification de texte

Cerisara Nathan, MPI

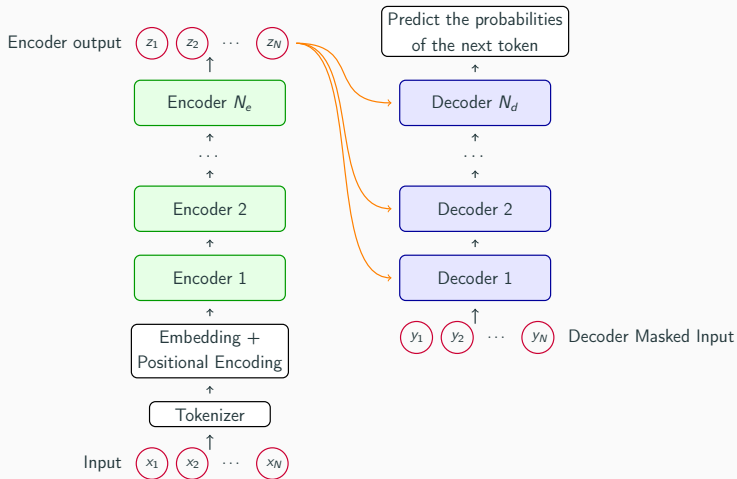
26 mai 2023

Sommaire

- Architecture Transformer
 - Vectorisation du texte
 - La partie Encodeur de l'architecture
 - Les matrices d'Attention
 - Le réseau Feed Forward
- Application personnelle
 - Objectif / Rapport à la ville
 - Le modèle BERT
 - La structure du réseau de neurone utilisée
 - Les données et l'apprentissage
 - Les résultats
- Annexes

L'architecture Transformer

Schéma de l'architecture dans le cas de la génération :



Vectorisation du texte : Tokenisation du texte

Tokenizer (bert-base-uncased)

Ex1 :

SENTENCE : "Neural Networks are so cool!"

TOKENS :

[101, 15756, 6125, 2024, 2061, 4658, 999, 102, 0, ..., 0]

[CLS] "neural" "networks" "are" "so" "cool" "!" [SEP]

Ex2 :

SENTENCE : "Bonjour le monde!"

TOKENS :

[101, 14753, 23099, 2099, 3393, 23117, 999, 102, 0, ..., 0]

[CLS] "bon" "##jou" "##r" "le" "monde" "!" [SEP]

Vectorisation du texte : Embeddings & Positional Encoding

Positional Encoding

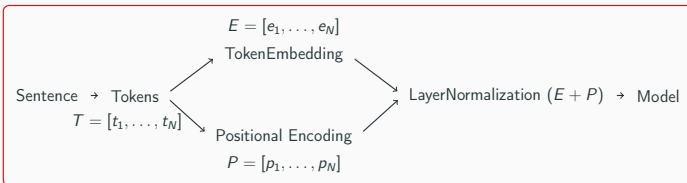
$$p_k = (s_0, c_1, \dots, s_{2i}, c_{2i+1}, \dots, s_N)$$

$$s_{2i} = \sin \left(k \cdot 10000^{-\frac{2i}{d_E}} \right)$$

$$c_{2i+1} = \cos \left(k \cdot 10000^{-\frac{2i+1}{d_E}} \right)$$

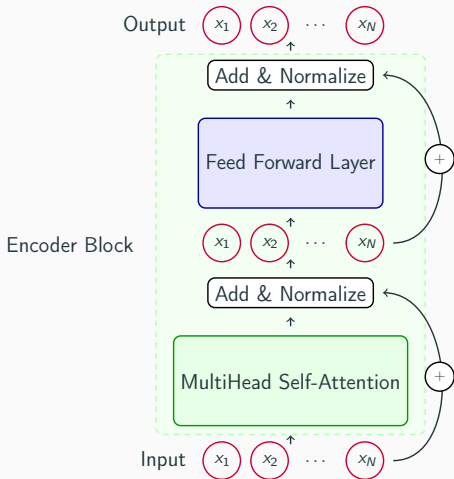
Token Embedding

$$t_k \longrightarrow \underbrace{(e_{k,0}, \dots, e_{k,d_E})}_{\text{dimension } d_E}$$

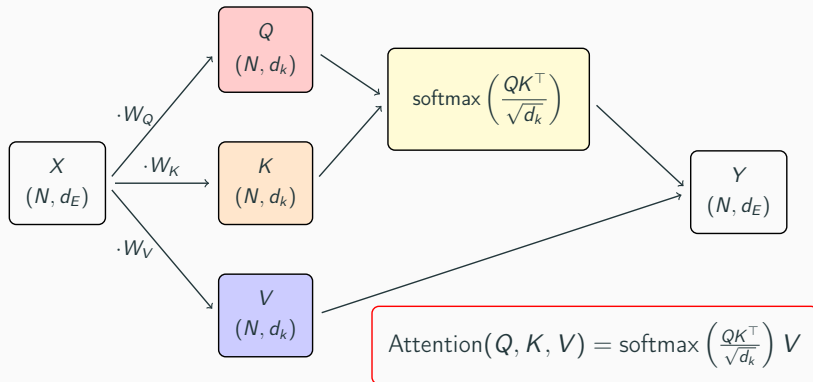


La partie Encodeur

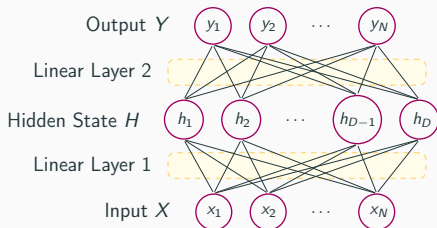
Schéma d'un block de la partie Encodeur de l'architecture Transformer :



Matrice d'attention



Le réseau Feed Forward



Couche linéaire :

$$X \mapsto X \cdot W^T + B$$

Application Personnelle : Objectifs & Rapport à la ville

- Objectif :
 - Classification de texte
 - Sentiment : Négatif \longleftrightarrow Positif
- Rapport à la ville :
 - Analyser les sentiments des habitants sur différents sujets

Le modèle BERT

- Architecture Transformer
- Plusieurs tailles de BERT (base, large, ...)
- BERT base : $12 \times$ blocks encoder \rightarrow 112M paramètres
- BooksCorpus (800M words) and English Wikipedia (2,500M words)
- Publié vers fin 2018 par des chercheurs de Google

La structure du réseau de neurone utilisée



Les données et l'apprentissage

- **Données** : Stanford Sentiment Dataset
- Train : 8544, Test : 2210
- **Temps d'entraînement** : $\approx 15\text{min}$
- **Algorithme d'apprentissage** : Adam optimizer
(extension of the stochastic gradient descent)
- **Fonction de Loss** : MinSquaredError

Les résultats

Ce que l'on a vu :

- Architecture Transformer
 - Le block d'encodeur
 - Les matrices d'attention
 - Les réseaux Feed Forward
- Application Personnelle

Annexes / Ouvertures :

- Détails du code python
- Adam optimizer
- La fonction de loss
- La Normalisation par couche (LayerNorm)
- La partie décodeur
- Comparaison avec le modèle GPT
- Le théorème d'approximation universel
- Bibliographie

Annexes : Code python

```
import torch.nn as nn

class Net(nn.Module):
    def __init__(self, dim_in, dim_out):
        super().__init__()
        #
        self.layer = nn.Linear(dim_in, dim_out)

    def forward(self, x):
        return self.layer(x)
```

Annexes : Adam Optimizer

Annexes : Fonction de loss

Annexes : Comparaison avec le modèle GPT

Annexes : Le théorème d'approximation universel