Generative AI, GPT & BERT Models
Secret Of the Dark knowledge

https://github.com/mohamedYoussfi/nlp-sentiment-analysis-bert

Mohamed Youssfi, Lab. Informatique, Intelligence Artificielle et Cyber Securité, ENSET Mohammedia, Université Hassan II de Casablanca

Generative AI

- Generative AI refers to a type of Artificial Intelligence that is capable of generating new data, such as images, music, or text, based on a set of training data.
- Generative AI systems use machine learning techniques, such as neural networks, to analyze and learn patterns in the data, and then generate new data that follows those patterns
- Reasons that have led to a significant acceleration in the pace of research and development in generative AI.
  - Powerful Language Models (**GPT-3, BERT**) : **Machine translation, Text summarisation, Sentiment analysis, Conversational AI**
  - Generative Adversarial Networks (**GANs**)
  - **Abundant Data:**
    - In 2022, users sent around **650 million Tweets** per **day**.
    - In 2022, **333.2 billion emails** were sent every **day**.
  - Hardware Advancements : **GPU & TPU , Cloud Computing**

Generative Adversarial Networks (GANs)

- GANs : Approach to generative modeling using deep learning methods, such as convolutional neural networks.
- Initially developed by Ian Goodfellow and his colleagues in **June 2014**
- GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models:
  - The **Generator** model that we train to generate new random samples
  - The **Discriminator** model that tries to classify if generated fake samples are either real (from the domain) or fake (generated).
  - The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

Generative Adversarial Networks (GANs)

https://thispersondoesnotexist.com/

Which of these faces are real? And which of these are fake?

They're all fake!

The key innovation behind this technology, Generative Adversarial Networks (GANs) were invented in **2014**.

Transformers Attention Is All You Need

https://arxiv.org/pdf/1706.03762.pdf

arXiv:1706.03762v5 [cs.CL] 6 Dec 2017

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention

My cat can not move the bag because it is tired

0.95

0.03

Attention Is All You Need

Multi-Head Attention

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Figure 1: The Transformer - model architecture.

Attention Is All You Need

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V$$

arXiv:1706.03762v5 [cs.CL] 6 Dec 2017

Scaled Dot-Product Attention

Dot Product

$$\text{Score} = Q \bullet K = \begin{bmatrix} q1.k1 & q1.k2 \\ q2.k1 & q2.k2 \end{bmatrix} \Big/ \sqrt{d_k} = \begin{bmatrix} DP11 & DP12 \\ DP21 & DP22 \end{bmatrix}$$

$$SM = \text{SoftMax}\left(\begin{bmatrix} DP11 & DP12 \\ DP21 & DP22 \end{bmatrix}\right) = \begin{bmatrix} 0.88 & 0.12 \\ 0.11 & 0.89 \end{bmatrix}$$

$$Z = SM \,.\, V = \begin{bmatrix} 0.88 & 0.12 \\ 0.11 & 0.89 \end{bmatrix} \times \begin{bmatrix} \; & \; & \; \\ \; & \; & \; \end{bmatrix} = \begin{bmatrix} 0.88 & 0.12 \\ 0.11 & 0.89 \end{bmatrix}$$

Transformers

- A Transformer is a deep learning model, using **Self-Attention Mechanism**,
- Require less training time compared to older **RNN**, **LSTM** & **GRU** models thanks to the to **Parallelization**
- The original transformer model used an **Encoder/Decoder** architecture.
- It has led to the development of pretrained systems, such as
  - **GPT** (**G**enerative **P**re-**T**rained transformer) and
  - **BERT** (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers).

Models Behind Chat GPT : Dark Knowledge

- ChatGPT est une IA basée sur une architecture combinant des modèles de deep learning de type **Transformers** et des **modèles d'apprentissage par renforcement**
- Chat GPT exploite l'émergence du **Dark Knowledge** par référence à la matière noire en physique
- **Dark knowledge** est l'ensemble des connaissances cachées que le modèle de l'IA a pu acquérir de l'ensemble des données qu'il perçu
- Ce **Dark knowledge** représente une connaissance acquise dans un domaine X et transférable vers un autre domaine Y grâce au **Transfert Learning**
- Le fait d'entrainer un modèle avec des données d'un domaine X, permet de faire des prédiction dans le domaine X, mais le modèle cache d'autres connaissances sous exploitée dans le domaine X.

Models Behind Chat GPT : Embodiment = > Reinforcement Learning

Qu'est ce qui ferait que dans l'apprentissage de ChatGPT qu'on aie la possibilité d'exploiter ce Dark Knowledge?

- l'**Embodiment**
- **L'Embodiment** en IA est un principe qui suppose que l'IA ne peut émerger que si l'on donne un corps (Virtuel ou physique) à une modèle d'IA (**Agent = MIND + BODY**)
- Le corps permet d'induire le fait que :
  - L'agent se trouve dans un environnement
  - l'agent peut agir sur l'environnement
  - L'agent reçoie des récompenses de l'environnement
  - Ce qui provoque le changement de l'état de l'agent
  - L'agent décide de l'action optimale à prendre pour maximiser les récompenses en utilisant une table d'apprentissage construite pendant le processus d'entrainement selon un objectif à atteindre
  - Ce principe d'Embodiment est utilisé dans l'apprentissage par renforcement

Training Steps of Chat GPT

- **Etape 1 : Modèle A**
  - **Apprentissage Auto-supervisé**
    - Le fait d'entrainer un modèle avec du texte en ligne non labelisé en utilisant les transformers
    - Apprendre à générer la suite d'un texte qui est **statistiquement plausible** (Text-Completion)
- **Etape 2 : Modèle B**
  - Prendre un réplica du modèle A
  - On spécialise ce modèle B à une tache précise : **Apprentissage supervisé**
  - Dans cette étape, on fait appel à **des humains pour annoter ces conversations** pour indiquer au modèle si la réponse d'une étape de cette conversation est :
    - Acceptable
    - Politiquement correcte
    - Moralement et éthiquement acceptable
  - Ce modèle B a pour objectif **d'apprendre quels sont les patterns dans une conversation qui font qu'une réponse est correcte ou non**
  - C'est donc les institutions qui entrainent ces modèles qui nous dictent l'éthique que l'agent doit respecter.

Training Steps of Chat GPT

- **Etape 3 : Embodiment**
  - On prend le modèle A et y ajoute le principe de l'Embodiment => Agent A
  - Dans cet agent on y introduit un algorithme d'apprentissage par renforcement **Proximal Policy Optimization (PPO)**
  - L'objectif de l'agent A est de prendre le modèle A qui est capable de générer du texte selon le contexte de la conversation (Text-Completion)
  - Faire de sorte que le modèle A puisse générer **des réponses statistiquement plausibles et qui satisfassent le modèle B**. Ce dernier permet de juger la pertinence de la réponse du modèle A en attribuant une note à la réponse du modèle A
  - Cet agent A va essayer de générer des phrases qui respectent les Guide-lines du modèle B
  - L'interaction entre l'agent A et le modèle B est importante car elle permet de réutiliser un concept qui s'appelle « **Energy Based Model** »

BERT Transformer

- BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) is a family of language models introduced in 2018 by researchers at Google.
- BERT has become a ubiquitous baseline in Natural Language Processing (NLP)
- BERT was originally implemented in the English language at two model sizes:
  - BERTBASE: **12** encoders with **12** bidirectional self-heads-attention
  - **BERTLARGE: 24** encoders with **16** bidirectional self-heads-attention.
- Both models were pre-trained on the Toronto BookCorpus (**800M words**) and English Wikipedia (**2,500M words**).

https://tfhub.dev/google/collections/bert/1

BERT Transformer

- BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) is a family of language models introduced in 2018 by researchers at Google.
- BERT has become a ubiquitous baseline in Natural Language Processing (NLP)
- BERT was originally implemented in the English language at two model sizes:
  - BERTBASE: **12** encoders with **12** bidirectional self-heads-attention
  - **BERTLARGE: 24** encoders with **16** bidirectional self-heads-attention.
- Both models were pre-trained on the Toronto BookCorpus (**800M words**) and English Wikipedia (**2,500M words**).

https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4

https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3

Use Case : Sentiment Analysis

I would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. Forget pretty pictures painted for mainstream audiences, forget charm, forget romance...OZ doesn't mess around. The first episode I ever saw struck me as so nasty it was surreal, I couldn't say I was ready for it, but as I watched more, I developed a taste for Oz, and got accustomed to the high levels of graphic violence. Not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) Watching Oz, you may become comfortable with what is uncomfortable viewing....thats if you can get in touch with your darker side."

Use Case : Sentiment Analysis

```python
import pandas as pd
import matplotlib as plt
import seaborn as sns
import tensorflow_hub as hub
import tensorflow_text as text
import tensorflow as tf
```

```python
bert_encoder_url="https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4"
bert_preprocess_url="https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3"
```

```python
bert_preprocess_model = hub.KerasLayer(bert_preprocess_url)
bert_encoder_model = hub.KerasLayer(bert_encoder_url)
```

```
systemMemory: 32.00 GB
maxCacheSize: 10.67 GB
```

```python
review_test=["Hello my name is mohamed","today we are talking about transformers"]
```

```python
preprocessed_text=bert_preprocess_model(review_test)
encoded_text=bert_encoder_model(preprocessed_text)
```

```python
preprocessed_text.keys()
```

```
dict_keys(['input_type_ids', 'input_mask', 'input_word_ids'])
```

Text Preprocessor Bert Model

```
preprocessed_text=bert_preprocess_model(review_test)
```

```
preprocessed_text['input_word_ids'][0]
```

CLS Hello my name is mohamed SEP

Array(128)

```
[ 101, 7592, 2026, 2171, 2003, 14467, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
dtype=int32)>
```

```
preprocessed_text['input_mask'][0]
```

```
<tf.Tensor: shape=(128,), dtype=int32, numpy= array(
[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
dtype=int32)>
```

Bert Encoder Model

```
bert_preprocess_model = hub.KerasLayer(bert_preprocess_url)
bert_encoder_model = hub.KerasLayer(bert_encoder_url)
```

```
encoded_text.keys()
```

```
dict_keys(['pooled_output', 'sequence_output', 'encoder_outputs', 'default'])
```

```
encoded_text['sequence_output'][0][0]
```

```
# Contexttualized word embeeding : Premier mot (Hello) de la première phrase
# « Hello my name is mohamed » > Vecteur de dimension : 768
```

```
<tf.Tensor: shape=(768,), dtype=float32, numpy= array(
[ 1.13881275e-01, 3.06966960e-01, -1.34479269e-01, -
2.24087924e-01, -1.73023298e-01, -2.19552577e-01,
1.43109560e-01, 4.75077331e-01, -1.47830561e-01, -
3.31553817e-01, 1.86126515e-01, -1.79994583e-01,
3.72473225e-02, 4.16999549e-01, 1.83585156e-02, -
1.04494572e-01, -1.78568110e-01, 2.28355229e-01,
2.19092175e-01, 1.42559320e-01, 1.66799128e-01, -
1.49366185e-01, 1.29990935e-01, 1.39638230e-01, -
5.73812909e-02, 5.24293445e-03, -5.11699766e-02, -
4.11996961e-01, -2.71968937e+00, -,...
-8.07741356e+00, 1.17559068e-01, -2.58015692e-02, -
4.04933751e-01, -1.26915157e-01, -5.01500010e-01, -
4.46696579e-03, -4.09875900e-01, -7.49762058e-02, -
1.99619442e-01, 1.64636046e-01, 7.12240636e-02, -
2.95594245e-01, -5.64841405e-02, -3.65161300e-02,
6.25804842e-01], dtype=float32)>
```

Bert Encoder Model

```
# Contexttualized Sentence embeeding : Première phrase
# « Hello my name is mohamed » => Vecteur de dimension : 768
```

```
encoded_text['pooled_output'][0]
```

```
<tf.Tensor: shape=(768,), dtype=float32, numpy= array(
[-6.71814084e-01, -2.08563313e-01, 5.40385187e-01,
2.97822684e-01, -3.23103756e-01, 3.30618620e-02,
6.52756333e-01, 8.87683183e-02, 5.41845977e-01, -
9.98861015e-01, 3.81980479e-01, 5.95086738e-02,
9.74099338e-01, -3.59646559e-01, 8.89524221e-01, -
3.76179129e-01, 1.72169223e-01, -4.86407042e-01,
1.90916777e-01, -7.51642808e-02, 3.96276712e-01,
9.01618063e-01, 6.90044582e-01, 1.66912705e-01,
1.73067614e-01, 3.13725206e-03, -2.56693304e-01, -
8.25042874e-02, -4.69835550e-01, 3.85511100e-01, -
3.51046324e-02, 3.48850898e-02,...
-9.00455594e-01, -1.12172790e-01, -4.05894453e-03, -
3.92533422e-01, -3.18686455e-01, 2.28766233e-01,
3.88182372e-01, 7.88155138e-01, -7.47817308e-02,
7.66605973e-01, 2.26663485e-01, 7.24977136e-01,
4.53520954e-01, 1.44146308e-01, -5.51417172e-01,
8.12596560e-01]
, dtype=float32)>
```

Bert Encoder Model

```
# Contexttualized Sentence embeeding : Première phrase
# « Hello my name is mohamed » => Vecteur de dimension : 768
```

```
encoded_text['pooled_output'][0]
```

```
<tf.Tensor: shape=(768,), dtype=float32, numpy= array(
[-6.71814084e-01, -2.08563313e-01, 5.40385187e-01,
2.97822684e-01, -3.23103756e-01, 3.30618620e-02,
6.52756333e-01, 8.87683183e-02, 5.41845977e-01, -
9.98861015e-01, 3.81980479e-01, 5.95086738e-02,
9.74099338e-01, -3.59646559e-01, 8.89524221e-01, -
3.76179129e-01, 1.72169223e-01, -4.86407042e-01,
1.90916777e-01, -7.51642808e-02, 3.96276712e-01,
9.01618063e-01, 6.90044582e-01, 1.66912705e-01,
1.73067614e-01, 3.13725206e-03, -2.56693304e-01, -
8.25042874e-02, -4.69835550e-01, 3.85511100e-01, -
3.51046324e-02, 3.48850898e-02,...
-9.00455594e-01, -1.12172790e-01, -4.05894453e-03, -
3.92533422e-01, -3.18686455e-01, 2.28766233e-01,
3.88182372e-01, 7.88155138e-01, -7.47817308e-02,
7.66605973e-01, 2.26663485e-01, 7.24977136e-01,
4.53520954e-01, 1.44146308e-01, -5.51417172e-01,
8.12596560e-01]
, dtype=float32)>
```

Bert Encoder Model

```
# Contexttualized WORDe embeeding in a specific Encoder Bloc
```

```
len(encoded_text['encoder_outputs'])
```

```
Output : 12
```

```
encoded_text['encoder_outputs'][0]
```

```
<tf.Tensor: shape=(2, 128, 768), dtype=float32, numpy=
array(
[[[ 1.6370770e-01, -1.2917660e-02, -8.5544161e-02, ...,
-5.5640571e-02, 7.7056810e-02, -4.0374212e-03], [
6.8490756e-01, 2.2906061e-01, 4.2973149e-01, ..., -
3.2194525e-01, -3.4143475e-01, 1.2136697e-01], [
4.2342395e-02, -1.5809081e-01, 4.4588542e-01, ...,
6.3759273e-01, -6.4465159e-01, 7.2044395e-02]]],
dtype=float32)>
```

```
encoded_text['encoder_outputs'][-1][0]==encoded_text['sequence_output'][0]
```

```
<tf.Tensor: shape=(128, 768), dtype=bool, numpy=
array([[ True, True, True, ..., True, True, True], [
True, True, True, ..., True, True, True], [ True, True,
True, ..., True, True, True], ..., [ True, True, True,
..., True, True, True], [ True, True, True, ..., True,
True, True], [ True, True, True, ..., True, True,
True]])>
```

Sentiment Analysis Use Case

```
df=pd.read_csv('dataset.csv')
```

```
df.sample(10)
```

	review	sentiment
40828	As of this writing John Carpenter's 'Halloween...	positive
28510	The subject is banal, even in 1942. It is war ...	positive
45581	For a feature film, the plot closely follows h...	positive
34166	At first I thought that this one was supposed ...	negative
7973	When I first looked at the back of the cover o...	negative
25863	Wow. So my boyfriend and I went to the movie s...	negative
11270	First of all, i have nothing against Christian...	negative
2746	Father and son communicate very little. IN fac...	positive
33588	Certainly not a bad little low budget film, th...	negative
49445	This movie has been advertised for over three ...	negative

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

```
df.review.isna().sum()
```

```
Output : 0
```

```
df.sentiment.isna().sum()
```

```
Output : 0
```

```
df['sentiment'].value_counts()
```

```
sentiment
  positive 25000
  negative 25000
Name: count, dtype: int64
```

```
sns.countplot(x=df['sentiment'])
```

```
<Axes: xlabel='sentiment', ylabel='count'>
```

Sentiment Analysis Use Case

```
df.groupby('sentiment').describe()
```

	review			
	count	unique	top	freq
sentiment				
negative	25000	24698	This show comes up with interesting locations ...	3
positive	25000	24884	Loved today's show!!! It was a variety and not...	5

```
df['output']=df['sentiment'].apply(lambda x:1 if x=='positive' else 0)
```

```
df.sample(10)
```

	review	sentiment	output
28043	I was fortunate enough to catch a midnight scr...	positive	1
4866	The only reason I haven't given this film an a...	negative	0
24819	ultra cheezy soundtrack. vinnie tries really h...	negative	0
43838	So they hyped the violence and it's been brand...	negative	0
20984	Here's a horror version of PRISCILLA: QUEEN OF...	negative	0
36301	I don't know why all of the critics say this w...	positive	1
2883	The Beguiled was one of the few early Eastwood...	positive	1
49611	I caught this film late on a sat night/ Sunday...	positive	1
4296	Calling this a "Sunday School" movie might be ...	negative	0
11812	Sometimes I just want to laugh. Don't you? No ...	positive	1

Sentiment Analysis Model :

```
input_layer = tf.keras.layers.Input(shape=(), dtype=tf.dtypes.string, name="input_layer")

preprocess_layer = bert_preprocess_model(input_layer)

encoder_layer = bert_encoder_model(preprocess_layer)

dropout_layer = tf.keras.layers.Dropout(rate=0.1,name="dropout_layer")(encoder_layer['pooled_output’])

output_layer=tf.keras.layers.Dense(units=1,activation="sigmoid", name="output_layer")(dropout_layer)

model=tf.keras.Model(inputs=[input_layer],outputs=[output_layer])
```

```
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
 text (InputLayer)               [(None,)]            0           []

 keras_layer_1 (KerasLayer)      {'input_type_ids':   0           ['text[0][0]']
                                 (None, 128),
                                  'input_mask': (Non
                                 e, 128),
                                  'input_word_ids':
                                 (None, 128)}

 keras_layer_2 (KerasLayer)      {'pooled_output': (  109482241   ['keras_layer_1[0][0]',
                                 None, 768),                       'keras_layer_1[0][1]',
                                  'sequence_output':               'keras_layer_1[0][2]']
                                  (None, 128, 768),
                                  'encoder_outputs':
                                  [(None, 128, 768),
                                  (None, 128, 768),
                                  (None, 128, 768),
                                  (None, 128, 768),
                                  (None, 128, 768),
                                  (None, 128, 768),
                                  (None, 128, 768),
...
```

```
Total params: 109,483,010
Trainable params: 769
Non-trainable params: 109,482,241
```

Sentiment Analysis Model :

```
METRICS = [
 tf.keras.metrics.BinaryAccuracy("accuracy"),
 tf.keras.metrics.Recall(),
 tf.keras.metrics.Precision()
]
model.compile(optimizer = "adam", metrics = METRICS, loss = "binary_crossentropy")
```

Sentiment Analysis Model :

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(
   df['review'],
   df['output'],
   stratify=df['output'],
   test_size=0.2
)
```

```
print("x_train shape => ", x_train.shape)
print("y_train shape => ", y_train.shape)
print("x_test shape => ",  x_test.shape)
print("y_test shape => ",  y_test.shape)
```

```
x_train shape => (40000,)
y_train shape => (40000,)
x_test shape => (10000,)
y_test shape => (10000,)
```

Sentiment Analysis Model :

```
history = model.fit(x_train,y_train, epochs=10, validation_split=0.2)
```

```
Epoch 1/10 1172/1172 [==============================] - 2441s 2s/step - loss: 0.6024 - accuracy: 0.6787 -
precision: 0.6825 - recall: 0.6682
Epoch 2/10 1172/1172 [==============================] - 25493s 22s/step - loss: 0.5400 - accuracy: 0.7353
- precision: 0.7400 - recall: 0.7254
Epoch 3/10 1172/1172 [==============================] - 12301s 11s/step - loss: 0.5189 - accuracy: 0.7503
- precision: 0.7548 - recall: 0.7414
Epoch 4/10 1172/1172 [==============================] - 5072s 4s/step - loss: 0.5062 - accuracy: 0.7561 -
precision: 0.7607 - recall: 0.7475
Epoch 5/10 1172/1172 [==============================] - 3900s 3s/step - loss: 0.4992 - accuracy: 0.7602 -
precision: 0.7651 - recall: 0.7509
Epoch 6/10 1172/1172 [==============================] - 14002s 12s/step - loss: 0.4917 - accuracy: 0.7658
- precision: 0.7722 - recall: 0.7539
Epoch 7/10 1172/1172 [==============================] - 4760s 4s/step - loss: 0.4871 - accuracy: 0.7687 -
precision: 0.7746 - recall: 0.7581
Epoch 8/10 1172/1172 [==============================] - 2858s 2s/step - loss: 0.4827 - accuracy: 0.7697 -
precision: 0.7750 - recall: 0.7602
Epoch 9/10 1172/1172 [==============================] - 3761s 3s/step - loss: 0.4806 - accuracy: 0.7727 -
precision: 0.7774 - recall: 0.7642
Epoch 10/10 1172/1172 [==============================] - 3089s 3s/step - loss: 0.4772 - accuracy: 0.7730
- precision: 0.7777 - recall: 0.7645
```

Durée d'entrainement : 873 Minute => 14,55 heures

```
model.save("./sentiment-analysis")
```

Sentiment Analysis Model :

```
model.evaluate(x_test,y_test)
```

```
[==============================] - 687s 2s/step
-   loss: 0.4757
-   accuracy: 0.7757
-   recall: 0.7140
-   precision: 0.8145
```

```
y_predicted = model.predict(x_test)
```

```
import numpy as np
import matplotlib.pyplot as plt

y_predicted = np.where(y_predicted>0.5,1,0)
```

```
array([[1], [0], [0], ..., [1], [0], [0]])
```

Sentiment Analysis Model :

```
from sklearn.metrics import confusion_matrix, classification_report
```

```
print(classification_report(y_test,y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.75      0.84      0.79      5000
           1       0.81      0.71      0.76      5000

    accuracy                           0.78     10000
   macro avg       0.78      0.78      0.77     10000
weighted avg       0.78      0.78      0.77     10000
```

Sentiment Analysis Model :

```
cm = confusion_matrix (y_test,y_predicted)
cm
```

```
array([
        [ 4187 , 813  ],
        [ 1430 , 3570 ]
     ])
```

```
sns.heatmap(cm,annot=True,fmt='d')
plt.xlabel("Predicted")
plt.ylabel("Truth")
```

Sentiment Analysis Model :

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

Sentiment Analysis Model :

```
reviews=['Hello, i do not like', "i like", "bad experience","good look"]
predicted=model.predict(reviews)
```

```
predicted=np.where(predicted>0.5,1,0)
```

```
predicted
```

```
array([[0], [1], [0], [1]])
```

Model Deployment : REST API based on Django

```
$ pip install django
$ pip install djangorestframework
$ django-admin startproject bert_sa .
$ python manage.py runserver
```

./bert_sa/urls.py

```python
from django.contrib import admin
from django.urls import path
from bert_sa import api

urlpatterns = [
  path("admin/", admin.site.urls),
  path("test/", api.test)
]
```

localhost:8000/test/?review=j%27ai%20vécu%20une%20très%20bonne%20expérience


```
{
    "name": "sentiment analysis",
    "review": "j'ai vécu une très bonne expérience",
    "review_trans": "I had a very good experience",
    "result": "1"
}
```


./bert_sa/api.py

```python
import tensorflow as tf
from django.http import JsonResponse
from rest_framework.decorators import api_view
import tensorflow_text as text
import numpy as np
from deep_translator import GoogleTranslator

translator = GoogleTranslator()

model = tf.keras.models.load_model("sentiment-analysis")


@api_view(["GET"])
def test(request):
   review = request.query_params.get("review")
   review_translated = translator.translate(review)
   result = model.predict([review_translated])
   result = np.where(result > 0.5, 1, 0)
   data = {
     "name": "sentiment analysis",
     "review": review,
     "review_trans": review_translated,
     "result": str(result[0][0]),
    }
return JsonResponse(data)
```

Model Deployment : REST API based on Django

```
(base) $ /Users/mohamedyoussfi/miniconda3/bin/python3 manage.py runserver
Watching for file changes with StatReloader
Performing system che

Metal device set to: Ap

systemMemory: 32.00
maxCacheSize: 10.67 G

2023-07-05 08:05:37.8
System check identifie

You have 18 unapplied                                                    s.
Run 'python manage.p
July 05, 2023 - 08:05:3
Django version 4.2.2, using settings 'bert_sa.settings'
Starting
Quit the

1/1 [===
[05/Jul/
1/1 [===
[05/Jul/
1/1 [===
[05/Jul/
1/1 [===
[05/Jul/
```

localhost:8000/test/?review=j%27ai%20vécu%20une%20très%20bonne%20expérience


```
{
    "name": "sentiment analysis",
    "review": "j'ai vécu une très bonne expérience",
    "review_trans": "I had a very good experience",
    "result": "1"
}
```


localhost:8000/test/?review=je%20suis%20navré%20de%20vous%20de%20dire%20que%20je%20suis%20en%20collère


```
{
    "name": "sentiment analysis",
    "review": "je suis navré de vous de dire que je suis en collère",
    "review_trans": "I'm sorry for you to say that I'm angry",
    "result": "0"
}
```

Conclusion

- Generative AI
- Transformers
- Self Attention, Encoders, Decoders
- Transformers => NLP, Computer Vision
- Chat GPT :
  - Transformers,
  - Dark knowledge,
  - Embodiment

Références

- Attention is all you need : https://arxiv.org/pdf/1706.03762.pdf
- https://jalammar.github.io/illustrated-transformer/
- https://www.youtube.com/@ThibaultNeveu
- Great Learning – MIT : https://www.mygreatlearning.com/mit-data-science-and-machine-learning-program