ENSET

BUMS
BIBLIOTHÈQUE UNIVERSITAIRE MOHAMED SEKKAT
UNIVERSITÉ HASSAN II DE CASABLANCA

Generative AI
Prompt Engineering

NLP
Transformers
LLM
Chat GPT

Large Language Models

Computer Vision, Transformers, Stable Diffusion, DALL-E

Prompt

Create an image of cat, having fun in a party, wearing a black tuxedo with tea in one hand

Créez une image de chat, s'amusant lors d'une fête, portant un smoking noir avec du thé dans une main

Generative AI Models
(Dall-E, Stable Diffusion, Midjourney)

Prompt

Create a short story about a girl who lost her parents in an accident. The story should not exceed 1000 words.

Generative AI Models
(GPT3, BERT)

In a small town nestled between rolling hills and whispering trees, lived a young girl named Lily. She had a heart as wild and free as the meadows that surrounded her family's modest cottage. Lily's world revolved around the laughter of her parents, their warm embraces, and the stories they told her under the starlit sky.

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

Generative AI & Prompt Engineering

- Notre interaction avec l'Intelligence Artificielle(IA) est devenue une science qu'il faudrait enseigner à tout le monde.
- Une compétence qui fera sans doute des différences dans notre métier et notre business.
- Dans le domaine de l'éducation, les pratiques pédagogiques ont beaucoup innové et la réussite des apprenants est tributaire du développement de leurs capacités métacognitives.
- Grâce au mécanisme d'attention, l'IA a pu développer aussi sa capacité métacognitive de manière inattendue et incroyable.
- La réalité est que les LLMs sont devenus de véritables assistants pour la pratique de tous les métiers.
- Concevoir et évaluer les performances des prompts et déployer ces prompts via des APIs, est devenu une science qui va générer de nouvelles opportunités de business avec un fort potentiel d'embauche.
- Ne cherchez pas à minimiser les capacités d'un Large Language Models (LLM) comme ChatGPT.
- Si vous n'êtes pas satisfait de sa réponse, c'est que vous n'avez pas su lui poser la bonne question et que vous n'avez pas su le mettre dans le vrai contexte de votre problématique. Prompt Engineering vise à tirer un maximum de profits des LLMs (Large Language Models). À suivre …

Computer Vision, Transformers, Stable Diffusion, DALL-E

NLP, Transformers, LLM, ChatGPT

Outline

1. • Artificial Intelligence
2. • Generative AI : State Of Art, Discriminator AI vs Generative AI
3. • NLP, Transformers, LLM
- • How LLM (Chat GPT) is Trained
4. • Few Shot Learning vs Fine Tuning
5. • Prompt Engineering at scale, Zero Shot Prompt, Few Shot Prompt (Instructional, Reasoning, Induction, Paraphrasing)
6. • Retrieval-Augmented Generation (RAG)
7. • Use Cases : Patients Sentiment Analysis
   - ❑ **Prompt Engineering : 1. Define Objectives and Metrics, 2. Assemble Data, 3. Derive Prompt, 4. Evaluate Prompt, 5. Deploy Prompt**
   - ❑ **Fine Tuning**
   - ❑ RAG

Artificial Intelligence

- L'**intelligence artificielle** : cherche à doter les systèmes informatiques de capacités intellectuelles comparables à celles des êtres humains »
- **Machine Learning :** Une branche de l'IA qui consiste à entrainer des modèles de deux façon **:**
  - **Apprentissage par les données**
    - Apprentissage supervisé (Données étiquetés)
    - Apprentissage non supervisé (Clustering : données non étiquetés, structurées)
    - Apprentissage auto supervisé (Données non étiquetés non structurée)
  - **Apprentissage par l'expérience :** Apprentissage par renforcement

Apprentissage Supervisé

Embodiment

Apprentissage Par Renforcement

Generative AI (State Of Art)

https://this-person-does-not-exist.com/en

- 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.
- Reasons that have led to a significant acceleration in the pace of research and development in generative AI.
  - Powerful Language Models :
    - For NLP : (GPT-3, BERT) : Machine translation, Text summarization, Sentiment analysis, Conversational AI
    - For Computer Vision : GANs, Stable Diffusion, DALL-E, Midjourney
  - 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**

2014
- Generative Adversarial Networks (GANs) (Ian Goodfellow)

2017
- Break through in NLP : **Transformers** (google)

  Attention is All You need (https://arxiv.org/pdf/1706.03762.pdf)

2018
- Advent of Large Models :
  - **BERT** (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) , Google
  - **GPT** (**G**enerative **P**re-trained **T**ransformers **) (Open AI)**

2019
- Advanced GANs – Style GANs

2020
- **GPT-3** (Premier LLM)(Open AI)

2021
- Vision Transformers – CLIP, **DALL-E** : Images (OpenAI)

2022
- Chat GPT : **GPT – 3.5 (Text)**, **DALL-E 2(Images)**

2023
- **BARD** (Google), **GPT- 4** (Open AI)
- **Stable Diffusion** (Images)

Discriminative AI Vs Generative AI

Input Data

Train the model

Learn Patterns

Labels

Input Data

Discriminative AI Models

(CNN)

Classification Task

Cat

Prompt

Create an image of cat, having fun in a party, wearing a black tuxedo with tea in one hand

Créez une image de chat, s'amusant lors d'une fête, portant un smoking noir avec du thé dans une main

Generative AI Models

(Dall-E, Stable Diffusion, Midjourney)

Prompt

Create a short story about a girl who lost her parents in an accident. The story should not exceed 1000 words.

Generative AI Models

(GPT3, BERT)

In a small town nestled between rolling hills and whispering trees, lived a young girl named Lily. She had a heart as wild and free as the meadows that surrounded her family's modest cottage. Lily's world revolved around the laughter of her parents, their warm embraces, and the stories they told her under the starlit sky.

https://this-person-does-not-exist.com/en

NLP, Transformers, LLM

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

- **NLP (Natural Language processing) aims** to equip computers the ability to analyze, understand and generate human language.
- **Before 2017**, NLP use Deep Learning Techniques based on Neural Network Models such **RNN**, **LSTM** or **GRU**
- In 2017, The paper "**Attention is All you need**" introduced **transformers** which are based on **attention mechanism** to brings a significant breakthrough that improve significantly the performance of NLP models
- **BERT** (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) , Google
- **GPT** (**Generative Pre-trained Transformers), Open AI**
- **Large Language Models (LLMs)**, based on transformers are trained using a large corpus of data and **HPC** resources from cloud computing
- LLMs exhibit state-of-the-art performance on NLP tasks that fall into two broad buckets:
  - **Text To Label generation (classification)**
    - **Sentiment Analysis**
  - **Text To Text generation**
    - **Text Translation**
    - **Text Summarization**

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

Attention Is All You Need

Transformers

Figure 1: The Transformer - model architecture.

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

How LLMs are trained?

- **Etape 1 : Modèle A**
  - **Apprentissage Auto-supervisé**
- **Etape 2 : Modèle B**
  - Prendre un réplica du modèle A
  - On spécialise ce modèle B à une tache précise : **Apprentissage supervisé**
- **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

Few Shot Learning vs Fine Tuning

- Pour tirer profit des LLMs, on peut procéder de deux façons :
  - Few Shot Learning
  - Fine Tuning
- Few Shot Learning consiste à en interroger le LLM Généraliste en lui envoyant des promptes bien conçus pour contextualiser la problématique en question. Le prompt contient 3 parties :
  - System Message : **Le Contexte de la problématique en question**
  - Few Shot Examples : **Quelques Exemples spécifiques du contexte avec réponse attendues**
  - Use Query : **La question de l'utilisateur**
- Fine Tuning consiste à :
  - Entrainer un LLM Généraliste avec un dataset relatif à un domaine spécifique
  - Le nouveau LLM spécifique obtenu peut être interrogé avec des prompt sans préciser d'exemples spécifiques dans le prompt.

Prompt Engineering

Length of input + output

More temperature = More randomness in response

More Top P = More tokens selected for completion ; nucleus sampling échantillonnage du noyau

More FP = Less chance of tokens repeating

- **Prompt** = Specific set of instructions sent to a LLM to accomplish a task
- **Engineering** = Process of designing, evaluating and deploying the prompt for specific tasks :
  1. **Task definition** and **metrics**,
  2. **Prepare Data** (Dataset, Examples and Gold Examples),
  3. **Design** and **Evaluate prompt**
  4. **Integrate prompt** (APIs, data bases, Applications)
  5. **Deploy and monitor** (APIs, Applications in production)

Interact with ChatGPT using POSTMAN

https://api.openai.com/v1/chat/completions

https://api.openai.com/v1/chat/completions

POST https://api.openai.com/v1/chat/completions Send

Params | Authorization | Headers (11) | Body | Pre-request Script | Tests | Settings

Headers 9 hidden

API KEY

Key	Value
Content-Type	application/json
Authorization	Bearer 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
Key	Value

Params | Authorization | Headers (11) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | JSON

```
{
    "model": "gpt-3.5-turbo",
    "messages": [{"role": "user", "content": "Say this is a test!"}],
    "temperature": 0.7
}
```


```
{
    "id": "chatcmpl-8lcxSStSzdQOC9sBulqfZk8l1G8Aa",
    "object": "chat.completion",
    "created": 1706361862,
    "model": "gpt-3.5-turbo-0613",
    "choices": [
        {
            "index": 0,
            "message": {
                "role": "assistant",
                "content": "This is a test!"
            },
            "logprobs": null,
            "finish_reason": "stop"
        }
    ],
    "usage": {
        "prompt_tokens": 13,
        "completion_tokens": 5,
        "total_tokens": 18
    },
    "system_fingerprint": null
}
```


https://platform.openai.com/docs/api-reference/introduction

How to Get Open AI API KEY

platform.openai.com/api-keys

- Playground
- Assistants
- Fine-tuning
- API keys
- Files
- Usage
- Settings

API keys

https://platform.openai.com/api-keys

Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically disable any API key that we've found has leaked publicly.

Enable tracking to see usage per API key on the Usage page.

NAME	SECRET KEY	TRACKING	CREATED	LAST USED	PERMISSIONS
k4	sk-...xIxa	Enabled	23 janv. 2024	27 janv. 2024	All

+ Create new secret key

Default organization

If you belong to multiple organizations, this setting controls which organization is used by default when making requests with the API keys above.

ENSET Hassan II Uni...

Note: You can also specify which organization to use for each API request. See Authentication to learn more.

Interact with ChatGPT using POSTMAN

https://api.openai.com/v1/chat/completions

https://api.openai.com/v1/chat/completions

POST https://api.openai.com/v1/chat/completions Send

Params | Authorization | Headers (11) | Body | Pre-request Script | Tests | Settings

Headers 9 hidden

API KEY

	Key	Value
☑	Content-Type	application/json
☑	Authorization	Bearer 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
	Key	Value

```
"choices": [
{
 "index": 0,
 "message": {
   "role": "assistant",
   "content": "Positive sentiment"
}
]
```

```
{
"model": "gpt-3.5-turbo",
"messages": [
  { "role": "system", "content": "Perform sentiment analysis on laptop reviews.presented in the input delimited by triple backticks, that is, ```." },
  { "role": "user", "content": "```  Laptop performance is the best than i have never seen ```" }
],
"temperature": 0
}
```

https://platform.openai.com/docs/api-reference/introduction

Python Example 1

```
!pip install openai==0.28.0
```

```
import openai
```

```
openai.api_key="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"
```

```
prompt = [
 {"role": "system", "content": "You are a helpful assistant."},
 {"role": "user", "content": "What's the weather like?"},
 {"role": "assistant", "content": "It's raining."},
 {"role": "user", "content": "What should I take with me?"}
]
```

```
response = openai.ChatCompletion.create(
   model="gpt-4",
   messages= prompt,
   n=2,
   temperature = 0.8
)
```

```
print(response)
```

```
print(response['choices'][0]['message']['content'])
print(response['choices'][1]['message']['content'])
```

- Considering the weather, it would be advisable to take an umbrella or a raincoat with you. You might also want to wear waterproof shoes.
- Considering the weather, you should take an umbrella or raincoat with you. It would also be a good idea to wear waterproof shoes.


```
{
 "id": "chatcmpl-8grBJ7O8kdOPIFbzlFAoQF7gweLcc",
 "object": "chat.completion",
 "created": 1705224897,
 "model": "gpt-4-0613",
 "choices": [
  {
   "index": 0,
   "message": {
    "role": "assistant",
    "content": "Considering the weather, it would be advisable to take an umbrella or a raincoat with you. You might also want to wear waterproof shoes."
   },
   "logprobs": null,
   "finish_reason": "stop"
  },
  {
   "index": 1,
   "message": {
    "role": "assistant",
    "content": "Considering the weather, you should take an umbrella or raincoat with you. It would also be a good idea to wear waterproof shoes."
   },
   "logprobs": null,
   "finish_reason": "stop"
  }
 ],
 "usage": {
  "prompt_tokens": 42,
  "completion_tokens": 55,
  "total_tokens": 97
 },
 "system_fingerprint": null
}
```

Python Example 2 : Aspect Based Sentiment Analysis

```
system_message = """
Perform aspect based sentiment analysis on laptop reviews presented in the input delimited by triple backticks, that is, ```.
In each review there might be one or more of the following aspects: screen, keyboard, and mousepad.
For each review presented as input:
- Identify if there are any of the 3 aspects (screen, keyboard, mousepad) present in the review.
- Assign a sentiment polarity (positive, negative or neutral) for each aspect
Arrange your response a JSON object with the following headers:
- category:[list of aspects]
- polarity:[list of corresponding polarities for each aspect]
"""
```

```
prompt = [
  {"role": "system", "content": system_message},
  {"role": "user", "content": """
       ```The screen is good. The keyboard is bad and the mousepad is quite```
       """},
  {"role": "assistant", "content": """
      {'category':['screen','keyboard','mousepad'],'polarity':['positive','negative','neutral']}
      """},
  {"role": "user", "content": """
       ```I am impressed by the quality of the screen. disappointed with the keyboard,
          but the mousepad is of very good quality```
       """}
]
```

```
response = openai.ChatCompletion.create(
   model="gpt-4",
   messages= prompt,
   n=2,
   temperature = 0,
   max_tokens = 100
)
```

```
print(response['choices'][0]['message']['content'])
print(response['choices'][1]['message']['content'])
```

```
{'category':['screen','keyboard','mousepad'],'polarity':['positive','negative','positive']}
{'category':['screen','keyboard','mousepad'],'polarity':['positive','negative','positive']}
```

Max Tokens

Max Tokens Limit : 4096

Prompt Max Tokens : 800

Model	Input	Output	Max tokens	Training data
gpt-3.5-turbo	$0.0015 / 1K tokens	$0.002 / 1K tokens	4,096 tokens	Up to Sep 2021
gpt-3.5-turbo-16k	$0.003 / 1K tokens	$0.004 / 1K tokens	16,384 tokens	Up to Sep 2021

Model	Input	Output	Max tokens	Training data
gpt-4	$0.03 / 1K tokens	$0.06 / 1K tokens	8,192 tokens	Up to Sep 2021
gpt-4-32k	$0.06 / 1K tokens	$0.12 / 1K tokens	32,768 tokens	Up to Sep 2021

https://platform.openai.com/tokenizer

GPT-3.5 & GPT-4 GPT-3 (Legacy)

```
You are an objective assistant, part of multinational computer, phone,
laptop, and hardware manufacturer, to summarize unstructured customer
reviews in simple words.
```

Tokens 29 Characters 165

```
You are an objective assistant, part of multinational computer, phone,
 laptop, and hardware manufacturer, to summarize unstructured customer
 reviews in simple words.
```

```
[2675, 527, 459, 16945, 18328, 11, 961, 315, 69026, 6500, 11, 4641, 11,
21288, 11, 323, 12035, 14290, 11, 311, 63179, 653, 52243, 6130, 8544,
304, 4382, 4339, 13]
```

Prompt Tokens Count with tiktoken

```
import tiktoken
```

```
def num_tokens_from_messages(messages):

  encoding = tiktoken.encoding_for_model("gpt-3.5-turbo")
    # Each message is sandwiched with <|start|>role and <|end|>
    # Hence, messages look like: <|start|>system or user or assistant{message}<|end|>
  tokens_per_message = 3 # token1:<|start|>, token2:system(or user or assistant), token3:<|end|>
  num_tokens = 0
  for message in messages:
     num_tokens += tokens_per_message
     for key, value in message.items():
        num_tokens += len(encoding.encode(value))
  num_tokens += 3 # every reply is primed with <|start|>assistant<|message|>

  return num_tokens
```

```
prompt = [
 {"role": "system", "content": "You are a helpful assistant."},
 {"role": "user", "content": "What's the weather like?"},
 {"role": "assistant", "content": "It's raining."},
 {"role": "user", "content": "What should I take with me?"}
]
```

```
print(num_tokens_from_messages(prompt))
```

Prompt Tokens Count in java : jtokit

```java
public static int numberOfToken(List<ChatMessage> messages){
   EncodingRegistry registry = Encodings.newDefaultEncodingRegistry();
   Encoding enc = registry.getEncodingForModel(ModelType.GPT_3_5_TURBO);
   // Each message is sandwiched with <|start|>role and <|end|>
   // Hence, messages look like: <|start|>system or user or assistant{message}<|end|>
   int tokensPerMessage = 3; // token1:<|start|>, token2:system(or user or assistant), token3:<|end|>
   int numberOfTokens = 0;
   for (ChatMessage message : messages){
      numberOfTokens += tokensPerMessage;
      int messageNumTokens = enc.encode(message.getContent()).size() + enc.encode(message.getRole()).size();
      numberOfTokens += messageNumTokens;
   }
   numberOfTokens += 3;
  return numberOfTokens ;
}
```

```xml
<dependency>
   <groupId>com.knuddels</groupId>
   <artifactId>jtokkit</artifactId>
   <version>0.6.1</version>
</dependency>
```

```java
String promptText = """
      [
       {"role":"system", "content": "You are a helpful assistant." },
       {"role":"user", "content": "What's the weather like?" },
       {"role":"assistant", "content":"It's raining." },
       {"role":"user", "content":"What should I take with me?" }
      ]
      """;
```

```java
GsonBuilder builder = new GsonBuilder().setPrettyPrinting();
Gson gson = builder.create();
ChatMessage[] chatMessagesArray = gson.fromJson(promptText,
ChatMessage[].class);
List<ChatMessage> prompt = Arrays.asList(chatMessagesArray);
```

```java
System.out.println(numberOfToken(prompt));
```

Rate limits

- Rate limits are restrictions that our API imposes on the number of times a user or client can access our services within a specified period of time.
- Why do we have rate limits?
  - They help protect against abuse or misuse of the API
  - Rate limits help ensure that everyone has fair access to the API
  - Rate limits can help to manage the aggregate load on its infrastructure
- Rate limits are measured in five ways:
  - **RPM** (requests per minute),
  - **RPD** (requests per day),
  - **TPM** (tokens per minute),
  - **TPD** (tokens per day),
  - **IPM** (images per minute).

Usage tier 1

MODEL	TOKEN LIMITS	REQUEST AND OTHER LIMITS
gpt-3.5-turbo	60 000 TPM	500 RPM 10 000 RPD
gpt-4	10 000 TPM	500 RPM 10 000 RPD
gpt-4-1106-preview	150 000 TPM 500 000 TPD	500 RPM 10 000 RPD
text-embedding-ada-002	1 000 000 TPM	3 000 RPM
dall-e-3		5 images per minute
tts-1		50 RPM
whisper-1		50 RPM

Temperature

- Parameter between 0 and 1 (Default Value 0.5)
- This parameter controls the variability in the response. Using a higher temperature results in potentially lower probability tokens being chosen.
- With higher temperatures, we could prompt the model to make multiple completions for the same prompt.
- With a temperature of 0, only the highest probability tokens are chosen so multiple completions should return the same response.
- Examples :
  - To ask the LLM to generate a story, we need to use high value of temperature
  - To ask the model to analyze and summarize a text, we need to a lower value of temperature

```
prompt = [
 {"role": "system", "content": "You are a helpful assistant."},
 {"role": "user", "content": "What's the weather like?"},
 {"role": "assistant", "content": "It's raining."},
 {"role": "user", "content": "What should I take with me?"}
]
```

```
response = openai.ChatCompletion.create(
   model="gpt-4",
   messages= prompt,
   n=2,
   temperature = 0.8
)
```

- Considering the weather, it would be advisable to take an umbrella or a raincoat with you. You might also want to wear waterproof shoes.
- Considering the weather, you should take an umbrella or raincoat with you. It would also be a good idea to wear waterproof shoes.

```
response = openai.ChatCompletion.create(
   model="gpt-4",
   messages= prompt,
   n=2,
   temperature = 0
)
```

- Considering the weather, it would be a good idea to take an umbrella or a raincoat, and possibly some waterproof footwear to keep your feet dry.
- Considering the weather, it would be a good idea to take an umbrella or a raincoat, and possibly some waterproof footwear to keep your feet dry.

Prompt

- Specific set of instructions sent to a LLM to accomplish a task
- Two types of prompts :
  - Zero Shot Prompt
  - Few Shot Prompt

System Message :
Clear instructions explaining the task that the LLM should accomplish. Should include the expected format of user input and output and a chain of thought to accomplish the task

Few Shot Examples :
Input - Output pairs delineating the expected response for exemplar inputs. Outputs need not be golden

User Input
Input presented in the format mentioned in the system message

→ GPT → Prediction

Prompt

Zero Shot Prompt

System Message

User Input

Prompt Parameters

```
{
 "model": "gpt-3.5-turbo",
 "messages": [
   {
    "role": "system",
    "content": "SYSTEM MESSAGE"
   },
   {
    "role": "user",
    "content": "USER INPUT"
   }
 ],
 "temperature": 0.5,
 "max_tokens":500,
 "top_p":0.95,
 "frequency_penalty":0
}
```

Few Shot Prompt

System Message

Few Shot Examples

User Input

Parameters

```
{
 "model": "gpt-3.5-turbo",
 "messages": [
   {
    "role": "system",
    "content": "SYSTEM MESSAGE"
   },
   {
    "role": "user",
    "content": "USER INPUT 1"
   },
   {
    "role": "assistant",
    "content": "RESPONSE 1"
   },
   {
    "role": "user",
    "content": "USER INPUT 2 "
   },
   {
    "role": "assistant",
    "content": "RESPONSE 2"
   },
   {
    "role": "user",
    "content": "USER INPUT"
   }
 ],
 "temperature": 0.5
}
```

There are four common patterns of prompts :

- **Instruction prompt**
- **Reasoning prompt**
- **Induction prompt**
- **Paraphrasing prompt**
- **Chain-of-Thought Promp** (Cot prompt)

Instructional prompt (Prompt pédagogique)

- Les invites pédagogiques sont un partenaire précieux dans les tâches créatives où les idées initiales sont difficiles à trouver.
- Par exemple, imaginez que vous soyez chargé de proposer une description qui figurera sur la page produit du [Clavier de jeu Razer Ornata V3 X] sur Amazon.
- Voici une invite qui accomplit cette tâche ; varier la température générera une grande variété de descriptions de produits facilitant le processus créatif.

```
instuction_prompt = """

Vous êtes marketeur pour la société de jeux Razer.
Vous trouverez ci-dessous les métadonnées sur le clavier de jeu Razer Ornata V3 X :
- Marque : Razer
- Série : Ornata V3 X
- Numéro de modèle de l'article : RZ03-04470200-R3U1.
- Plateforme matérielle : PC
- Système d'exploitation : Microsoft Windows
- Poids de l'article : 2,97 livres
- Dimensions du produit : 17,46 x 5,68 x 1,23 pouces
- Dimensions de l'article LxlxH : 17,46 x 5,68 x 1,23 pouces
- Couleur : noir classique.
- Fabricant : Razer
- Langue: français
- ASIN : B09X6GJ691
- Caractéristiques spéciales : touches à profil bas, résistantes aux éclaboussures,
repose-poignet ergonomique, éclairage Chroma RVB, interrupteurs à membrane
silencieux, options de routage des câbles

Avec ces informations, rédigez une description élégante « À propos de cet article »
qui sera utilisée sur sa page produit Amazon.

Utilisez des puces pour délimiter les principales fonctionnalités mentionnées dans
la description.
"""
```

Instructional prompt

```python
def get_completion(prompt, model_id='gpt-4', max_tokens=10, tempe=0.9):
    completion_messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
      model=model_id,
      messages= completion_messages,
      max_tokens=max_tokens,
      temperature = temp
    )
return response.choices[0].message["content"]
```

Rédigez une description Élégante à propos de cet article pour Amazone

```python
response = get_completion(
    prmpt=instuction_prompt,
    model_id='gpt-4',
    max_tokens=800,
    temp=0.8
)
```

```python
print(response)
```

```
Introducing the Razer Ornata V3 X gaming keyboard, an essential
tool for the devoted gamer. Elegantly crafted and engineered to
bring your gaming experience to the next level, this keyboard
features the following:
- **Brand**: Trust in the quality and innovation that comes
  with the Razer name, a leading brand in the gaming industry.
  The Ornata V3 X is part of their premium series.
- **Hardware Platform**: Designed for the PC platform, ensuring
  compatibility and seamless integration with your existing
  gaming setup. - **Operating System**: Fully compatible with
  Microsoft Windows, ensuring zero lag or delay in your gaming
  commands.
- **Lightweight**: Weighing just 2.97 pounds, this keyboard is
  lightweight and portable, ideal for transport to gaming
  events or simply moving around your gaming space.
- **Dimensions**: With dimensions measuring 17.46 x 5.68 x 1.23
  inches, this keyboard provides a perfect balance of size and
  functionality, ensuring your gaming desk remains clutter-free
  without compromising on gaming experience.
- **Color**: Presented in a Classic Black color, this keyboard
  exudes a sleek and professional look, perfect for any gaming
  setup.
- **Special Features**: The Razer Ornata V3 X is packed with
  unique features designed to improve your gaming experience:
      - Low-Profile Keys: Fast and efficient keystroke
        registration with minimal noise.
      - Spill Resistant: Anxious about accidental spills? This
        keyboard has you covered, designed to resist damage
        from liquids.
      - Ergonomic Wrist Rest: Comfort is paramount during long
        gaming sessions. The included ergonomic wrist rest
        ensures just that.
      - Chroma RGB Lighting: Customize your gaming experience
        with immersive RGB lighting options.
      - Silent Membrane Switches: Quiet keystrokes for those
        who game in shared spaces or simply prefer silent
        gaming.
      - Cable Routing Options: Keep your desk clutter-free with
        flexible cable routing options.
Elevate your gaming with the Razer Ornata V3 X Keyboard.
Designed for performance. Built for durability. Perfected for
comfort.
```

Instructional prompt

Using the same context to write a Twitter post for the product.

```
prompt = """
You are a marketer for the gaming company Razer.
Below is the metadata about the Razer Ornata V3 X
gaming keyboard:
- Brand: Razer
- Series: Ornata V3 X
- Item model number: RZ03-04470200-R3U1
- Hardware Platform: PC
- Operating System: Microsoft Windows
- Item Weight: 2.97 pounds
- Product Dimensions: 17.46 x 5.68 x 1.23 inches
- Item Dimensions LxWxH: 17.46 x 5.68 x 1.23 inches
- Color: Classic Black
- Manufacturer: Razer
- Language: English
- ASIN: B09X6GJ691
- Special Features: Low-Profile Keys, Spill
Resistant, Ergonomic Wrist Rest, Chroma RGB
Lighting, Silent Membrane Switches, Cable Routing
Options

Write an exciting Twitter post (280 characters)
announcing the features of this product.
"""
```

🎮 Elevate your gaming experience with the new Razer Ornata V3 X keyboard! Enjoy low-profile keys, spill resistance, an ergonomic wrist rest, and vibrant Chroma RGB lighting. Plus, silent membrane switches and cable routing options! Game on in classic black! #RazerGaming ⌨️🔥🎉

Reasoning Prompt : Prompt de raisonnement

- GPT is a great reasoning agent.
- This capability is best utilized by nudging the model to think in specific directions.

ABC Ecommerce est un détaillant en ligne ayant beaucoup de succès, spécialisé dans la vente de gadgets électroniques.
Ils ont ont une clientèle solide et proposent une large gamme de produits.
Cependant, l'entreprise est actuellement confrontée à une pénurie de fournisseurs sur l'une de ses principales gammes de produits; les smartphones.
La chaîne logistique impliquée dans la livraison des smartphones connaît de multiples contraintes, entraînant des retards et des ruptures de stock.
Cette situation pose un défi important à ABC Ecommerce, car elle a un impact sur ses revenus et sur la satisfaction de ses clients.
Pour répondre à cette problématique, l'entreprise a identifié trois grandes pistes d'action :
1. Diversifier les fournisseurs et les produits.
2. Améliorer la visibilité de la chaîne d'approvisionnement et la collaboration .
3. Explorez les capacités de fabrication internes ou les opportunités de partenariat.
Parmi les mesures ci-dessus, lesquelles recommanderiez-vous à ABC Ecommerce et pourquoi ?

Je recommanderais une combinaison des trois approches, car elles abordent chacune différents aspects du problème :
1. Diversifier les fournisseurs et les produits : cela peut atténuer le risque de dépendre d'un seul fournisseur ou d'une seule gamme de produits. En se diversifiant, ABC Ecommerce peut garantir un approvisionnement constant en produits même si l'un des fournisseurs rencontre des problèmes. Cela permet également à l'entreprise d'offrir une gamme plus large de produits à ses clients, augmentant ainsi potentiellement ses ventes.
2. Améliorer la visibilité et la collaboration de la chaîne d'approvisionnement : cela peut aider ABC Ecommerce à identifier plus tôt les problèmes dans la chaîne d'approvisionnement, leur permettant ainsi de prendre des mesures correctives avant que cela n'affecte la disponibilité du produit. Cela encourage également la collaboration entre les différentes parties impliquées dans la chaîne d'approvisionnement, ce qui peut améliorer l'efficacité et conduire à des solutions innovantes à des problèmes communs.
3. Explorer les capacités de fabrication internes ou les opportunités de partenariat : cela peut donner à ABC Ecommerce plus de contrôle sur son approvisionnement en produits. En fabriquant des produits en interne ou dans le cadre d'un partenariat, l'entreprise ne serait pas aussi dépendante des fournisseurs extérieurs. Cependant, cette approche peut être coûteuse et prendre du temps et doit donc être soigneusement étudiée.

En conclusion, même si chaque approche présente ses avantages, une combinaison des trois donnerait probablement les meilleurs résultats pour ABC Ecommerce. Une chaîne d'approvisionnement diversifiée et transparente, complétée par des capacités de fabrication internes potentielles, peut contribuer à garantir un approvisionnement régulier en produits, à améliorer l'efficacité et à réduire la dépendance à l'égard de fournisseurs uniques.

Reasoning Prompt with ReAct Framework

- Une meilleure méthode pour utiliser les capacités de raisonnement de GPT consiste à utiliser le Framework **ReAct** (**Reasoning and Action**).
- Avec ce Framework, nous considérons GPT comme un agent intelligent et nous codifions explicitement les actions disponibles pour le modèle.
- Cela incite le modèle à peser sur les alternatives disponibles avant d'agir.

```
prompt = """
ABC Ecommerce ……….
1. Diversifier les fournisseurs et les produits.
2. Améliorer la visibilité de la chaîne d'approvisionnement et la collaboration .
3. Explorez les capacités de fabrication internes ou les opportunités de
partenariat.

Recommander un plan d'action à l'entreprise et créer un plan étape par
étape que l'entreprise devra suivre dans cette situation.
Présentez votre réponse sous le format suivant :
{
  "question": <la question d'entrée à laquelle il faut répondre>,
  "reflection": <réfléchissez à vos options et au plan d'action pour
chaque option>,
  "action": <l'action à entreprendre parmi les trois options
présentées">,
  "raison": <le raisonnement derrière votre réponse>
  "étapes" : <plan étape par étape mettant en œuvre l'action>
}
"""
```


{
"question": "Quel plan d'action ABC Ecommerce devrait-il prendre pour surmonter le problème de pénurie de fournisseurs sur sa principale gamme de produits, les smartphones ?",
"reflection ": "En considérant les options disponibles, la diversification des fournisseurs pourrait contribuer à réduire la dépendance à l'égard d'un fournisseur unique et réduire le risque de pénurie, mais cela pourrait également accroître la complexité opérationnelle. Améliorer la visibilité et la collaboration de la chaîne d'approvisionnement pourrait aider à mieux planifier et gérer les stocks, mais cela pourrait ne pas résoudre le problème de pénurie d'approvisionnement. Explorer les opportunités de fabrication en interne ou de partenariat. pourrait également être une option, mais cela pourrait nécessiter des investissements et du temps importants.",
"action": "Diversifier les fournisseurs et les produits.",
"raison": "La diversification des fournisseurs contribuerait à atténuer le risque de pénurie de fournisseurs. Dans le cas où un fournisseur est confrontée à des problèmes, l'entreprise ne serait pas entièrement dépendante de ce fournisseur et pourrait obtenir le même produit auprès d'autres fournisseurs. Cela lui donnerait également un meilleur pouvoir de négociation. La diversification des produits contribuerait à réduire la dépendance à l'égard d'une seule gamme de produits et pourrait ouvrir de nouvelles sources de revenus..
"étapes": [
"Étape 1 : Réaliser une étude de marché pour identifier de nouveaux fournisseurs potentiels capables de répondre aux exigences de qualité et de quantité de l'entreprise.",
"Étape 2 : Contacter les fournisseurs potentiels et négocier les conditions.",
"Étape 3 : Changer progressivement certaines des commandes aux nouveaux fournisseurs pour tester leur fiabilité.",
"Étape 4 : Réaliser une étude de marché pour identifier les nouveaux produits potentiels qui sont en demande et s'aligner sur le portefeuille de produits existant de l'entreprise.",
"Étape 5 : Ajouter les nouveaux produits au portefeuille de l'entreprise et les promouvoir auprès des clients.",
"Étape 6 : Surveiller les performances des nouveaux fournisseurs et produits et ajuster la stratégie si nécessaire."] }

Induction Prompt

- Une approche pour générer des prompts pour un modèle de langage consiste à :
  - Présenter des paires d'entrées-sorties
  - et à inviter le modèle à générer une prompt qui obtient la sortie donnée à partir des entrées fournies.
- Cette technique peut servir de base d'idées d'incitation, contribuant ainsi à relancer le processus de génération de prompts.

```
Vous êtes un assistant qui aide l'équipe marketing d'une entreprise
d'électronique à mieux comprendre les avis de ses clients.
Les exemples d'entrées-sorties suivants ont été collectés à titre de
suggestions pour que vous puissiez en tirer des leçons.
- Entrée : Gris commandé qui annonce un éclairage vert, quand on opte pour
une esthétique cheap, c'est bouleversant. La -souris fonctionne bien.
- Sortie : Souris
- Entrée : j'en ai acheté un pour les jeux sur PC. J'ai adoré, puis j'en ai
acheté une autre pour le travail. Cette souris n'est pas à la hauteur des
souris haut de gamme comme la série Logitech MX Master, mais à 1/5-/8ème du
prix, je ne m'attendais pas à ce niveau de qualité. Il fonctionne bien, la
molette de la souris semble lourde, les boutons latéraux sont bien placés
avec différentes textures pour que vous puissiez les distinguer. La souris
semble plutôt plastique et bon marché, mais pour le prix, c'est à peu près ce
à quoi je m'attendais. J'aime une souris filaire pour éviter que le
pointeur/le jeu ne saute à cause de la latence. Fil long également, ce qui
minimise les problèmes d'accrochage. Très bon rapport qualité/prix dans
l'ensemble.
- Sortie : souris, Logitech MX Master, boutons DPI, molette de la souris, fil

Créez pour vous-même un prompt pour extraire la sortie requise des entrées,
comme décrit dans les exemples ci-dessus.

Créez l'invite pour qu'elle contienne des exemples généralisés à partir de
ceux présentés ci-dessus.
N'oubliez pas que l'invite doit contenir des instructions que vous pouvez
comprendre et générer le résultat attendu compte tenu de l'entrée.
"""
```

System Message
Input, Output
Créez pour vous-même un prompt pour extraire la sortie requise des entrées, comme décrit dans les exemples ci-dessus.

Prompt:
Lisez attentivement l'avis du client et identifiez le principal produit électronique en question. Notez également toutes les caractéristiques ou composants spécifiques du produit mentionnés, ainsi que tout autre produit électronique mentionné à des fins de comparaison. Le résultat doit être une liste de ces éléments.

Induction Prompt

- One approach to generating prompts for a language model is to present input-output pairs and prompt the model to generate a prompt that elicits the given output from the provided inputs.
- This technique can serve as a seed set of prompting ideas, helping to kickstart the prompt generation process.

```
induction_prompt = """
You are an assistant that helps the marketing team of an electronics company to
better understand customer reviews.
The following input - output examples were collected as suggestions for you to
learn from.

Input: Ordered grey which advertises green lighting, when you're going for a
cheap aesthetic, it's upsetting. Mouse works fine.
Output: Mouse

Input: I bought one of these for PC gaming. Loved it, then bought another for
work.This mouse is not on par with high end mouses from like the Logitech MX
Master series, but at 1/5-/8th the price, I didn't expect that level of
quality. It does perform well, mouse wheel feels weighty, side buttons are well
place with different textures so you can tell them apart. DPI buttons are handy
for adjusting between games, work jobs, etc. The mouse does feel rather
plasticky and cheap, but for the money, it about what I expected. I like a
wired mouse to avoid the pointer/game jumping around due to latency.
Long wire too, so snagging issues are minimized. Great value overall.
Output: Mouse, Logitech MX Master, DPI Buttons, Mouse Wheel, Wire

Create a prompt for yourself to extract the required output from the inputs as
described in the above examples.
Craft the prompt to contain examples generalized from those presented above.
Remember that the prompt should contain instructions that you can understand
and generate the expected output given the input.
"""
```

```
Prompt:
Read the customer review carefully and
identify the main electronic product being
discussed. Also, note any specific features
or components of the product that are
mentioned, as well as any other electronic
products that are referred to for
comparison. The output should be a list of
these items.
```

Paraphrasing Prompt

- Le Prompte de paraphrase est une technique qui consiste à générer un ensemble de promptes à l'aide d'un LLM à partir d'un Prompt de départ,
- Puis à évaluer les performances de chaque prompte sur un ensemble de test.
- Choisir le prompt le plus performant

```
paraphrase_prompt = """

Un promprt de départ vous sera présentée,
délimitée par trois backticks, c'est-à-dire
```.
Ce prompt de départ sera présentée à un grand
modèle de langage qui génère une sortie
requise spécifique.

Veuillez générer 5 variantes du prompt de
départ en gardant intacte l'intention de
l'invite de départ.

```
Veuillez analyser les avis clients suivants
et identifier les produits et fonctionnalités
qui sont mentionnés.
Fournissez une liste de produits et de
fonctionnalités mentionnés dans chaque avis.
```
"""
```

```
``` Analysez les commentaires des clients fournis ci-dessous et identifiez les produits et
les fonctionnalités respectives qui sont discutés. Compilez une liste de ces produits et de
leurs fonctionnalités correspondantes mentionnées dans chaque avis. ```

``` Examinez les témoignages de clients fournis et déterminez les produits spécifiques ainsi
que leurs caractéristiques auxquels il est fait référence. Créez une liste détaillant le
produit et ses fonctionnalités associées mentionnées dans chaque avis. ```

``` Veuillez examiner les évaluations des clients ci-dessous et identifier les produits
explicites et leurs fonctionnalités qui sont en discussion. Créez une liste des produits et
fonctionnalités mentionnés dans chaque avis individuel. ```

``` Pouvez-vous analyser les retours clients suivants et reconnaître les produits et les
caractéristiques sur lesquels ils mettent l'accent ? Établissez une liste comprenant les
produits et les fonctionnalités mentionnés dans chacun des avis. ```

``` Inspectez les évaluations des clients qui s'ensuivent et discernez les produits et leurs
attributs qui sont mis en avant. Compilez une liste détaillant les produits et les
fonctionnalités respectives mentionnées dans chaque avis. ```
```

CoT Prompt : Chain-of-Thought Prompting :

Chain-of-Thought Prompting : CoT Prompt

- Pour l'invite CoT, nous ajoutons des instructions détaillées étape par étape au message système demandant au modèle de réfléchir attentivement avant de décider du résultat à générer
- En dehors de cet ajout, il n'y a aucun autre changement par rapport à prompt standard

```
cot_system_message = """
Classify the sentiment of movie reviews presented
in the input as 'positive' or 'negative'.
Movie reviews will be delimited by triple backticks
in the input.
Answer only 'positive' or 'negative'. Do not
explain your answer.

Instructions:
1. Carefully read the text of the review and think
through the options for sentiment provided
2. Consider the overall sentiment of the review and
estimate the probability of the review being
positive

To reiterate, your answer should strictly only
contain the label: positive or negative.
"""
```

Standard Prompt

- Q : Ahmed has 5 tennis balls, he buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls doas he have now?
- A : **The answer is 11**
- Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more. How many apples do they have?

The cafeteria now has 9 apples.

CoT Prompt

- Q : Ahmed has 5 tennis balls, he buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls doas he have now?
- A : **Ahmed started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5+6=11. The answer is 11**
- Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more. How many apples do they have?

The cafeteria started with 23 apples. They used 20, so they had 23-20=3 apples left. Then they bought 6 more, so they now have 3+6=9 apples.

Retrieval-Augmented Generation **(RAG)**

- Considérons que nous souhaitons interroger le LLM pour connaître le contenu d'un rapport annuel.
- Bien que le modèle soit capable de comprendre les instructions contenues dans le prompt, il ignore parfaitement ce qui se passe après la date de son entrainement.
- Concrètement, il est fort probable que le rapport annuel n'ait pas été ingéré lors de l'entrainement.
- Dans ce cas, le résultat de LLM serait, au mieux, inutile et, au pire, incorrect (appelé « **Hallucination** »).
- Pour corriger cette sérieuse limitation, nous devrons assembler et injecter un contexte généré dynamiquement en fonction des entrées de l'utilisateur.
- Un exemple typique est celui où les données stockées par les organisations sont utilisées comme source de contexte pour répondre aux questions posées par les utilisateurs.
- Cette idée de récupération et d'injection d'un contexte pertinent pour la requête dans un prompt pour augmenter la génération de réponse est appelée **Retrieval Augmented Generation (RAG)**

Text to Label

Text to Text

The building blocks of RAG

Step 1 : Ingestion des sources de données

- Ingestion des différentes sources de données dans une base de données pour former une couche de contexte (PDF, Audio, Vidéos, Excel, Word, etc.).
- Nous devrons sélectionner soigneusement les documents liés à la saisie de l'utilisateur afin que le LLM puisse être chargé de répondre uniquement en fonction du contexte.

The building blocks of RAG

Step 2 : Split to Chunks

- Transformer les données brutes en morceaux plus petits avant de les transmettre à un modèle d'intégration
- Deux raisons importantes pour lesquelles cela est nécessaire :
  1. Les modèles d'intégration sont eux-mêmes des modèles de transformateur et ont donc des contraintes de longueur d'entrée.
  2. Le découpage des données en morceaux facilite un contrôle précis des informations spécifiques qui peuvent être injectées comme contexte. Cela aide le modèle à se concentrer uniquement sur les informations les plus pertinentes pour la requête.

The building blocks of RAG

Step 3 : Content Embedding

- Transformer les données brutes des chunks en représentations vectorielles.
- Ceci permet de construire un moteur de recherche sémantique avec une base de donnée locale
- De cette façon, nous pouvons comparer la requête (convertie en vecteur) avec les vecteurs de documents et choisir uniquement les documents similaires à la requête.
- Cette recherche est basée sur une mesure de similarité comme **Cosine Similarity**
- Ce processus de création d'une représentation vectorielle à partir du texte d'entrée est appelé **intégration** et est réalisé à l'aide de **modèles d'intégration** (par exemple, Open AI `**ada**`).

https://huggingface.co/spaces/mteb/leaderboard

The building blocks of RAG

Step 4 : Store in Vector Database (VDBMS)

- Une fois les intégrations créées, elles peuvent être stockées pour une récupération efficace dans des magasins de données spécialisés tels que :
  - [Pinecone](https://www.pinecone.io/)
  - [ChromaDB](https://www.trychroma.com/).

The building blocks of RAG

Step 5 : RAG App

- Développer une application qui permet de :
  1. Saisir la requête utilisateur
  2. Transformer la requête en représentation vectorielle
  3. Interroger le VDBMS (Pinecone, ChromaDB, PostGreSQL) pour faire une recherche sémantique.
  4. Retrouver les documents pertinents lié au contexte de la requête
  5. Injecter les contenu contextuel retrouvé dans le prompt
  6. Présenter ce prompt au LLM pour l'inciter à réponse à la requête utilisateur en utilisant les données contextuelle les plus pertinentes.
  7. Récupérer la réponse du LLM
  8. Envoyer à l'utilisateur la réponse

Use case 1 : Patients Reviews Summarization Analysis - Healthcare Provider

- **Objectif**:
  - Vous êtes un Data Scientist/Data Analyst travaillant avec un prestataire de santé.
  - Dernièrement, l'organisation a reçu de nombreuses plaintes et témoignages d'insatisfaction à l'égard de la consultation fournie
  - On souhaite analyser toutes les données pour effectuer une analyse des causes profondes afin d'identifier la source du problème.
  - On dispose d'une grande quantité de données non structurées contenant des avis de patients qui ont été donnés après consultation des médecins sous contrat.
  - Votre tâche consiste à **concevoir un prompt** efficace pour extraire :
    - **Le nom du patient**,
    - **le médecin consultant et ses références,**
    - **les scores,**
    - **un résumé de l'examen,**
    - **Si le patient était satisfait ou non de la consultation**
    - **Les tags liés à une consultation insatisfaite.**
  - Les échantillons sont à votre disposition dans un fichier csv appelé : **patient_review.csv**

	patient_review_id	patient_review_text
0	1	John Smith - I recently had the privilege of c...
1	2	Sarah Johnson - I recently had a disappointing...

John Smith – I recently had the privilege of consulting with Dr. Emily Roberts, a board-certified Family Physician, and I must say it was an exceptional experience. From start to finish, everything about my visit was top-notch, and I couldn't be more satisfied with the care I received. Dr. Roberts' credentials and professionalism were immediately evident. Her warm and welcoming demeanor put me at ease right away. She took the time to listen attentively to my concerns, showing genuine empathy and understanding. This made me feel like more than just a patient; I felt like a valued individual whose health truly mattered. What stood out to me the most was Dr. Roberts' comprehensive knowledge and expertise. She asked detailed questions to get a full understanding of my medical history, and her diagnosis and treatment recommendations were both thorough and easy to understand. It was evident that she had a deep understanding of the latest medical advancements and treatment options. Throughout the consultation, Dr. Roberts maintained excellent communication. She explained each step of the diagnostic process and the rationale behind her recommendations. I appreciated how she encouraged questions and took the time to address all of my concerns, ensuring that I was well-informed and actively involved in my healthcare decisions. Moreover, the clinic itself was well-organized and clean, which further enhanced the overall experience. The support staff were courteous and efficient, contributing to a seamless visit. In the end, my experience with Dr. Emily Roberts was nothing short of outstanding. Her compassionate approach to patient care, coupled with her extensive medical knowledge, left me feeling confident in the treatment plan she devised. Thanks to her, I am now on the path to recovery, and I couldn't be more grateful. I wholeheartedly recommend Dr. Emily Roberts to anyone seeking exceptional medical care. Her professionalism, expertise, and genuine concern for her patients truly set her apart as an outstanding healthcare provider. My rating for this consultation is a resounding five out of five stars. Thank you, Dr. Roberts, for your exemplary service and commitment to patient well-being.

patient_name	consulting_doctor	review_rating	review_description	satisfaction	issue_tags
John Smith	Dr. Emily Roberts, MD	5	Exceptional experience with Dr. Emily Roberts....	True	None
Sarah Johnson	Dr. David Smith	2	Sarah Johnson had a disappointing consultation...	False	Bedside Manner, Diagnostic Skills, Communication

Use case 1 : Patients Review Analysis

```python
import pandas as pd
import openai
import json
from sklearn.model_selection import train_test_split
```

```python
df = pd.read_csv("patient_reviews.csv")
```

```python
df.head()
```

	patient_review_id	patient_review_text
0	1	John Smith - I recently had the privilege of c...
1	2	Sarah Johnson - I recently had a disappointing...

```python
def extract_review_details(prompt_template, text, max_tokens=800,
temp=0):
    prompt = prompt_template.format(text=text)
    completion_messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
       model='gpt-3.5-turbo',
       messages= completion_messages,
       max_tokens=max_tokens,
       temperature = temp
    )
    json_response = response.choices[0].message["content"]
    data = json.loads(json_response)
    return pd.DataFrame(data, index=[0])
```

```python
prompt_template = """
You are an assistant that supports a healthcare provider
in analyzing patient reviews

Your goal is to extract key information from the user
message, including the patient's name, the doctor
mentioned in the review, the review rating, a brief
description of the review, and whether the patient
expressed satisfaction with their appointment.

Go through the user's feedback step by step, and generate
a structured output for further analysis by the
healthcare provider in the below format as a json

"patient_name": <extract the patient’s first and last
name from the corpus>,
"consulting_doctor": <extract the doctor’s first and last
name and credentials from the corpus>,
"review_rating": <this has to be a number out of 5 points
– if you cannot find a rating, output NULL>,
"review_description": <summarize the review at most in 50
words>
"satisfaction": <this has to be a TRUE or FALSE value –
arrive at this conclusion using your own judgment>,
"issue_tags": <in the case of a negative review or
dissatisfaction, add tags which specify the area of
dissatisfaction as comma separated values>

Below is the review:
```{text}```
"""
```
```

Use case : Patients Review Analysis

	patient_review_id	patient_review_text
0	1	John Smith - I recently had the privilege of c...
1	2	Sarah Johnson - I recently had a disappointing...

Patient_review_text [0]

Patient_review_text [1]

LLM

Structured Data in JSON

```
review_details_df = pd.DataFrame()
for text in df['patient_review_text']:
    df_response = extract_review_details(prompt_template,text, max_tokens=800)
    review_details_df = pd.concat([review_details_df, df_response])
```


```
{
  "patient_name": "John Smith",
  "consulting_doctor": "Dr. Emily Roberts, a board-certified Family Physician",
  "review_rating": 5,
  "review_description": "Exceptional experience with Dr. Emily Roberts. Comprehensive knowledge and expertise, excellent communication, and compassionate approach to patient care. The clinic was well-organized and clean. Highly recommend!",
  "satisfaction": true,
  "issue_tags": ""
}
```


```
review_details_df
```

```
review_details_df['satisfaction'].value_counts()
```

```
satisfaction
True     3
False    2
Name: count, dtype: int64
```

```
import seaborn as sns
sns.displot(data=review_details_df, x= 'satisfaction')
```

	patient_name	consulting_doctor	review_rating	review_description	satisfaction	issue_tags
0	John Smith	Dr. Emily Roberts, MD	5	Exceptional experience with Dr. Emily Roberts....	True	None
0	Sarah Johnson	Dr. David Smith	2	Sarah Johnson had a disappointing consultation...	False	Bedside Manner, Diagnostic Skills, Communication

Use case 2 : Patients Sentiment Analysis

STEP 1 - Define Objectives and Metrics

- In sentiment analysis, we assign one of two labels - **positive or negative** (in some cases one of three labels - **positive**, **negative** or **neutral**) to the text presented as input.
- As use case, we will assign sentiment to customer reviews for products listed in the apparel category on Amazon.
- Since our prediction will be one of two labels - **positive or negative**, we could use the following metrics to measure the performance of a prompt.
  - **Accuracy score**: calculated as the number of correct predictions (i.e., the number of times the prediction matches the ground truth)
  - **Micro-F1 score**: calculated by computing the True Positives (TP), False Negatives (FN), and False Positives (FP) across all classes. This metric is used when we suspect class imbalance.
- In practise, it is preferable to use a nuanced metric such as micro-F1 over plain accuracy.

- To illustrate the computation of micro F1 score for a two-class scenario, we can use a confusion matrix.
- For a binary classification problem, A confusion matrix is a table that shows
  - The number of True Positives (TP),
  - The number of False Positives (FP),
  - The number of True negatives (TN),
  - The number of False Negatives (FN)
- Here is an example of a confusion matrix for a two-class scenario:

Predicted/Actual	Positive	Negative
Positive	50 (TP)	10 (FP)
Negative	5 (FN)	100 (TN)

- **Micro-precision (MP)** = TP / (TP + FP) = 50 / (50 + 10) = **0.833**
- **Micro-recall (MR)** = TP / (TP + FN) = 50 / (50 + 5) = **0.909**
- **Micro-F1 score** = 2 * (**MP** * **MR**) / (**MP** + **MR**)
- = 2 * (0.833 * 0.909) / (0.833 + 0.909) = **0.870**

Use case : Patients Sentiment Analysis

STEP 2 - Assemble Data

- We need to have a representative, annotated corpus of examples illustrating the particular task we are trying to solve.
- Since the number of examples required is not high, our focus is on the quality of the examples and its impact on model performance.

```
imdb_reviews_corpus = load_dataset("imdb")
```

```
train_df = imdb_reviews_corpus['train'].to_pandas()
test_df = imdb_reviews_corpus['test'].to_pandas()
```

```
train_df.sample(6)
```

	text	label
12329	I don't remember seeing another murder/mystery...	0
9719	Man, what the hell were the people who made th...	0
15794	This is a top finnish film this year,although ...	1
17262	I was young film student in 1979 when the Unio...	1
24610	Nina Foch delivers a surprisingly strong perfo...	1
15894	The world of the 1973 sci-fi drama SOYLENT GRE...	1

- **IMDB Data Set** :
  - IMDB dataset having 50K movie reviews for natural language processing or Text analytics.
  - This is a dataset for binary sentiment classification.
  - Provide a set of 25,000 highly polar movie reviews for training and 25,000 for testing.
  - So, predict the number of positive and negative reviews using either classification or deep learning algorithms.

```
train_df['sentiment'].hist()
```

```
train_df.sample(5)
```

	text	label	sentiment
17019	Seriously, can you imagine such a spread of ta...	1	positive
7627	Sure it takes place in the west, but the title...	0	negative
18545	A really sweet movie that has some similaritie...	1	positive
17399	This movie is really genuine and random. It's ...	1	positive
3934	Alicia Silverstone (pre-"Clueless") plays a mo...	0	negative

```
train_df['sentiment']=train_df['label'].apply(lambda x:'positive' if x==1 else 'negative')
```

Task 1: Sentiment analysis

STEP 2 - Assemble Data

- Split the data into two segments :
  - one segment that gives us a pool to draw few-shot examples
  - another segment that gives us a pool of gold examples.

```
examples_df, gold_examples_df = train_test_split(
    train_df,
    test_size=0.2,
    random_state=42
)
```

```
(examples_df.shape, gold_examples_df.shape)
```

```
((20000, 3), (5000, 3))
```

```
columns_to_select = ['text', 'sentiment']
gold_examples = gold_examples_df.loc[:, columns_to_select]
gold_examples = gold_examples.sample(50, random_state=42) .to_json(orient='records')
```

Task 1: Sentiment analysis

STEP 3 – Derive Prompt

```
user_message_template = """```{movie_review}```"""
```

Zero Shot Prompt :

```
zero_shot_system_message = """
Classify the sentiment of movie reviews presented in the input as 'positive' or 'negative'.
Movie reviews will be delimited by triple backticks in the input.
Answer only 'positive' or 'negative'. Do not explain your answer.
"""
```

```
zero_shot_prompt = [{'role':'system', 'content': zero_shot_system_message}]
```

Task 1: Sentiment analysis

STEP 3 – Derive Prompt

Few Shot Prompt :

```
few_shot_system_message = """
Classify the sentiment of movie reviews presented in the input as 'positive' or 'negative'.
Movie reviews will be delimited by triple backticks in the input.
Answer only 'positive' or 'negative'. Do not explain your answer.
"""
```

```
positive_reviews = (examples_df.sentiment == 'positive')
negative_reviews = (examples_df.sentiment == 'negative')
```

```
columns_to_select = ['text', 'sentiment']
positive_examples = examples_df.loc[positive_reviews, columns_to_select].sample(4)
negative_examples = examples_df.loc[negative_reviews, columns_to_select].sample(4)
```

```
positive_examples
```

	text	sentiment
17228	I think that people are under estimating this ...	positive
16707	The title is onomatopoeic, the sound of a stre...	positive
18468	After seeing the terrible, terrible, terrible ...	positive
23543	This is my favorite show. I think it is utterl...	positive

	text	sentiment
5054	Very outdated film with awful, cliché-ridden a...	negative
7025	I got this in the DVD 10 pack CURSE OF THE DEA...	negative
2679	Dear me... Peter Sellers was one of the most o...	negative
779	New York, I Love You, or rather should-be-titl...	negative

Task 1: Sentiment analysis

STEP 3 – Derive Prompt

Examples

Text	Sentiment
Review 1	positive
Review 2	positive
Review 3	positive
Review 4	positive
Review 5	negative
Review 6	negative
Review 7	Negative
Review 8	Negative

positive reviews

Text	Sentiment
Review 1	positive
Review 2	positive
Review 3	positive
Review 4	positive

negative reviews

Text	Sentiment
Review 5	negative
Review 6	negative
Review 7	Negative
Review 8	Negative

Text	Sentiment
Review 1	positive
Review 2	positive

Random Subsets

Text	Sentiment
Review 5	negative
Review 6	negative

Few Shot Examples Arranged in random order

Text	Sentiment
Review 1	positive
Review 5	negative
Review 2	positive
Review 6	negative

Task 1: Sentiment analysis

STEP 3 – Derive Prompt

Few Shot Examples
Arranged in random order

Text	Sentiment
Review 1	positive
Review 5	negative
Review 2	positive
Review 6	negative

```
few_shot_system_message =
"""
Classify the sentiment of
movie reviews presented in
the input as 'positive' or
'negative'.
Movie reviews will be
delimited by triple
backticks in the input.
Answer only 'positive' or
'negative'. Do not explain
your answer.
"""
```

```
def create_prompt( 1 , 2 , 3 )

    Create few_shot_examples using the
    user message template.

  return system_message +
         few_shot_examples
```

```
user_message_template = """```{movie_review}```"""
```

Task 1: Sentiment analysis

STEP 3 – Derive Prompt

Few Shot Prompt

```
def create_examples(dataset, n=4):
    positive_reviews = (dataset.sentiment == 'positive')
    negative_reviews = (dataset.sentiment == 'negative')
    columns_to_select = ['text', 'sentiment']
    positive_examples = dataset.loc[positive_reviews,columns_to_select].sample(n)
    negative_examples = dataset.loc[negative_reviews, columns_to_select].sample(n)
    examples = pd.concat([positive_examples, negative_examples])
    randomized_examples = examples.sample(2*n, replace=False)
    return randomized_examples.to_json(orient='records')
```

```
examples = create_examples(train_df, n=5)
```

Few Shot Examples
Arranged in random order

Text	Sentiment
Review 1	positive
Review 5	negative
Review 2	positive
Review 6	negative

Task 1: Sentiment analysis

```
user_message_template = """```{movie_review}```"""
```

STEP 3 – Derive Prompt

Few Shot Prompt

```
def create_prompt(system_message, examples, user_message_template):
   few_shot_prompt = [{'role':'system', 'content': system_message}]
   for example in json.loads(examples):
      example_review = example['text’]
      example_sentiment = example['sentiment’]
      few_shot_prompt.append( {
         'role': 'user’,
         'content': user_message_template.format(movie_review=example_review)
       } )
      few_shot_prompt.append({
         'role': 'assistant’,
         'content': f"{example_sentiment}"
       } )
   return few_shot_prompt
```

```
few_shot_prompt = create_prompt(
    few_shot_system_message,
    examples,
    user_message_template
)
```

Few Shot Prompt

System Message

```
[
  {
   "role": "system",
   "content": "SYSTEM MESSAGE"
  },
```

Few Shot Examples

```
  {
   "role": "user",
   "content": "```Review 1```"
  },
  {
   "role": "assistant",
   "content": "positive"
  },
  {
   "role": "user",
   "content": " ```Review 2``` "
  },
  {
   "role": "assistant",
   "content": "negative"
  },

]
```

User Input

```
{
   "role": "user",
   "content": "```User Review```"
}
```

Task 1: Sentiment analysis

STEP 3 – Derive Prompt

```
print(json.dumps(few_shot_prompt, indent=3))
```

```
[
 {
  "role": "system",
  "content": "\nClassify the sentiment ......\n"
 },
 {
  "role": "user",
  "content": "```please don't rent or even .....```"
 },
 {
  "role": "assistant", "content": "negative"
 },
 {
  "role": "user",
  "content": "```When I heard that Adrian Pasdar ......```"
 },
 {
  "role": "assistant", "content": "positive"
 },
```

Few Shot Prompt

```
 {
  "role": "user",
  "content": "```The acting was bad, the script was bad and .....```"
 },
 {
  "role": "assistant",
  "content": "negative"
 },
 {
  "role": "user",
  "content": "```I've just returned from a ......\"```"
 },
 {
  "role": "assistant",
  "content": "positive"
 }
]
```

```
num_tokens_from_messages(few_shot_prompt)
```

1048

Task 1: Sentiment analysis

STEP 3 – Derive Prompt

Chain-of-Thought Prompting : CoT Prompt

- For the CoT prompt, we add detailed step-by-step instructions to the few shot system message instructing the model to carefully ponder before assigning sentiment.
- Apart from this addition, there are no further changes from the few-shot prompt.

```
cot_system_message = """
Classify the sentiment of movie reviews presented
in the input as 'positive' or 'negative'.
Movie reviews will be delimited by triple backticks
in the input.
Answer only 'positive' or 'negative'. Do not
explain your answer.

Instructions:
1. Carefully read the text of the review and think
through the options for sentiment provided
2. Consider the overall sentiment of the review and
estimate the probability of the review being
positive

To reiterate, your answer should strictly only
contain the label: positive or negative.
"""
```

Standard Prompt

- Q : Ahmed has 5 tennis balls, he buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls doas he have now?
- A : The answer is 11
- Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more. How many apples do they have?

The cafeteria now has 9 apples.

CoT Prompt

- Q : Ahmed has 5 tennis balls, he buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls doas he have now?
- A : Ahmed started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5+6=11. The answer is 11
- Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more. How many apples do they have?

The cafeteria started with 23 apples. They used 20, so they had 23-20=3 apples left. Then they bought 6 more, so they now have 3+6=9 apples.

```
cot_few_shot_prompt = create_prompt(
   cot_system_message,
   examples,
   user_message_template
)
```

Task 1: Sentiment analysis

STEP 4 : Evaluate Prompt

```
def evaluate_prompt(prompt, gold_examples, user_message_template):
   model_predictions, ground_truths = [], []
   for example in json.loads(gold_examples):
      gold_input = example['text’]
      user_input = [
        {
         'role':'user’,
         'content': user_message_template.format(movie_review=gold_input)
        }
       ]
      try:
         response = openai.ChatCompletion.create(
           model="gpt-3.5-turbo",
           messages= prompt+user_input,
           temperature = 0,
           max_tokens=2
         )
         prediction = response['choices'][0]['message']['content’]
         model_predictions.append(prediction.strip().lower())
         ground_truths.append(example['sentiment’])
      except Exception as e:
         print (e)
         continue
      micro_f1_score = f1_score(ground_truths, model_predictions,average="micro")

      return micro_f1_score
```

```
evaluate_prompt(
  zero_shot_prompt,
  gold_examples,
  user_message_template
)
```

0.94

```
evaluate_prompt(
  few_shot_prompt,
  gold_examples,
  user_message_template
)
```

0.96

```
evaluate_prompt(
   cot_few_shot_prompt,
   gold_examples,
   user_message_template
)
```

0.98

Task 1: Sentiment analysis

STEP 4 : Evaluate Prompt

- We will need to run these evaluations with multiple choices of examples to get a sense of variability in F1 score for the few-shot prompt and the CoT prompt.

```
from tqdm import tqdm

num_eval_runs = 5
few_shot_performance, cot_few_shot_performance = [], []

for _ in tqdm(range(num_eval_runs)):
   examples = create_examples(examples_df, n=2)
   few_shot_prompt =   create_prompt(few_shot_system_message, examples, user_message_template)
   cot_few_shot_prompt = create_prompt(cot_system_message, examples, user_message_template)
   few_shot_micro_f1 = evaluate_prompt(few_shot_prompt, gold_examples, user_message_template)
   cot_few_shot_micro_f1 = evaluate_prompt(cot_few_shot_prompt, gold_examples, user_message_template)
   few_shot_performance.append(few_shot_micro_f1)
   cot_few_shot_performance.append(cot_few_shot_micro_f1)
   time.sleep(1)
```

```
import numpy as np
np.array(few_shot_performance).mean(), np.array(few_shot_performance).std()

(0.99, 0.019999999999999976)
```

Fine-tuning

- Fine-tuning lets you get more out of the models available through the API by providing:
  - Higher quality results than prompting
  - Ability to train on more examples than can fit in a prompt
  - Token savings due to shorter prompts
  - Lower latency requests
- OpenAI's text generation models have been pre-trained on a vast amount of text.
- Fine-tuning improves on **few-shot learning** by training on many more examples than can fit in the prompt, letting you achieve better results on a wide number of tasks.
- **Once a model has been fine-tuned, you won't need to provide as many examples in the prompt.**
- This saves costs and enables lower-latency requests.
- At a high level, fine-tuning involves the following steps:
  1. **Prepare and upload training data**
  2. **Train a new fine-tuned model**
  3. **Evaluate results and go back to step 1 if needed**
  4. **Use your fine-tuned model**

Fine_tuning_train_data.jsonl

```
{"messages": [{"role": "system", "content": ".."},{"role": "user", "content":  ".."}, {"role": "assistant", "content": ".."}]}
{"messages": [{"role": "system", "content": ".."},{"role": "user", "content":  ".."}, {"role": "assistant", "content": ".."}]}
{"messages": [{"role": "system", "content": ".."},{"role": "user", "content":  ".."}, {"role": "assistant", "content": ".."}]}
{"messages": [{"role": "system", "content": ".."},{"role": "user", "content":  ".."}, {"role": "assistant", "content": ".."}]}
```

Fine-tuning

```
fine_tuning_train_data =[];
data= json.loads(create_examples(train_df, n=10))
for example in data :
   row = {
          "messages" : [
              {"role":"system", "content" : cot_system_message},
              {"role":"user", "content": user_message_template.format(movie_review=example ['text'])},
              {"role":"assistant", "content": example ['sentiment’]}
           ]
         }
   fine_tuning_train_data.append(row)
```

```
f = open('fine_tuning_file.jsonl', 'w')
for row in fine_tuning_train_data:
   json.dump(row,f)
   f.write("\n")
```

fine_tuning_file.jsonl


```
{"messages": [{"role": "system", "content": "\nClassify… "}, {"role": "user", "content": "```Checking…```"}, {"role": "assistant", "content": "negative"}]}
{"messages": [{"role": "system", "content": "\nClassify… "}, {"role": "user", "content": "```Checking…```"}, {"role": "assistant", "content": "positive"}]}
{"messages": [{"role": "system", "content": "\nClassify… "}, {"role": "user", "content": "```Checking…```"}, {"role": "assistant", "content": "negative"}]}
{"messages": [{"role": "system", "content": "\nClassify… "}, {"role": "user", "content": "```Checking…```"}, {"role": "assistant", "content": "positive"}]}
{"messages": [{"role": "system", "content": "\nClassify… "}, {"role": "user", "content": "```Checking…```"}, {"role": "assistant", "content": "negative"}]}
```

Fine-tuning

platform.openai.com/finetune
Playground
Assistants
Fine-tuning
API keys
Files
Usage
Settings
Documentation
Fine-tuning
All
Successful
Failed
ft:gpt-3.5-turbo-1106:enset-hassan-ii-university-of-casablanca::8kUh6FL2
gpt-3.5-turbo-1106
Failed
Create a fine-tuned model
Base model
gpt-3.5-turbo-1106
Training data
Add a jsonl file to use for training.
Upload new
Select existing
Upload a file or drag and drop here
(.jsonl)
fine_tuning_file.jsonl
fine_tuning_file.jsonl
Upload and Select
Validation data
Add a jsonl file to use for validation metrics.
Upload new
Select existing
None
Learn about fine-tuning
Cancel
Create

Fine-tuning

platform.openai.com/finetune/ftjob-yWBcD4zsEMNVCFboYN2XJNqg
Playground
Assistants
Fine-tuning
API keys
Files
Usage
Settings
Documentation
Help
All products
ENSET Hassan II Uni...
Fine-tuning
All
Successful
Failed
Learn more
Create
ft:gpt-3.5-turbo-1106:enset-hassan-ii-university-of-casablanca::8kUh6FL2
24/01/2024 10:59
gpt-3.5-turbo-1106
Failed
24/01/2024 10:55
MODEL
ft:gpt-3.5-turbo-1106:enset-hassan-ii-university-of-casablanca::8kUh6FL2
Succeeded
Job ID
ftjob-yWBcD4zsEMNVCFboYN2XJNqg
Base model
gpt-3.5-turbo-1106
Created at
24 janv. 2024, 10:59
Trained tokens
38390
Epochs
5
Files
Training
fine_tuning_file.jsonl
Validation
-
Training loss
0.0000
6
4
2
0
1
26
51
85
Messages
Metrics
11:22:52
The job has successfully completed
11:22:49
New fine-tuned model created: ft:gpt-3.5-turbo-1106:enset-hassan-ii-university-of-casablanca::8kUh6FL2
11:17:04
Fine-tuning job started
11:00:01
Files validated, moving job to queued state

Fine-tuning

```
cot_prompt = [
    {
      "role" :"system",
      "content":cot_system_message
    },
    {
      "role":"user",
      "content":train_df ['text'][0]
    }
  ]
response = openai.ChatCompletion.create(
  model="ft:gpt-3.5-turbo-1106:enset-hassan-ii-university-of-casablanca::8kUh6FL2",
  messages= cot_prompt,
  temperature = 0
)
print(response['choices'][0]['message']['content'])
```

```
positive
```

The Next Episode

Classification tasks

- **Objectives**
  - Present LLMs as an effective tool to tackle classification problems.
  - Illustrate prompt engineering for sentiment analysis (including aspect based sentiment analysis) using Azure Open AI.
  - Build and evaluate LLMs on broader classification tasks; identify and correct for bias.

- A central problem that we tackle in NLP is to equip computers the ability to analyze, understand and generate human language.
- LLMs exhibit state-of-the-art performance on NLP tasks that fall into two broad buckets:
  - **Text To Label generation (classification)**
  - **Text To Text generation**

Text To Label Generation

- A Text To label task involves the categorization of a given input text into a defined set of categories.
- There are three types of classification tasks:
  1. **Binary classification**: Input text is classified into one of two classes (typically negations). Spam detection & sentiment analysis are examples of binary classification.
  2. **Multi-class classification**: Input text is classified into one of more than two classes. Classifying a customer complaint as one of **critical**, **major** or **minor** is an example of multi-class classification.
  3. **Multi-label classification**: Input text is classified into one or more labels. Assigning themes to legal documents, such as Intellectual Property Law, Tax Law, and Employment Law, is an example of multi-label classification (note that each input can be assigned multiple classes).

Text To Text Generation

- A text To text generation task involves the generation of a relevant response to the input depending on the task at hand.
- Examples include:
  - **Text Summarization** (abstractive or extractive). For example, LLMs are used to summarize key themes in the analyst presentation about a company of interest.
  - **Question Answering** (including multiple choice). For example, LLMs are used to power rule-based support agents in the financial sector based on the question asked by a customer (e.g., account-related, investment-related).
  - **Matching a premise to a hypothesis.** This is a general problem where two pieces of text are checked for agreement (i.e., an entailment) or disagreement (i.e., a contradiction).

Operationalizing Generative AI

Java Example

```java
String promptText = """
    [
     {"role":"system", "content": "You are a helpful assistant." },
     {"role":"user", "content": "What's the weather like?" },
     {"role":"assistant", "content":"It's raining." },
     {"role":"user", "content":"What should I take with me?" }
    ]
    """;
GsonBuilder builder = new GsonBuilder().setPrettyPrinting();
Gson gson = builder.create();
ChatMessage[] chatMessagesArray = gson.fromJson(promptText, ChatMessage[].class);
List<ChatMessage> prompt = Arrays.asList(chatMessagesArray);
OpenAiService service = new OpenAiService("sk-bwDGZ2............");
ChatCompletionRequest completionRequest = ChatCompletionRequest.builder()
    .messages(prompt)
    .temperature(0.5)
    .model("gpt-4")
    .build();
ChatCompletionResult response = service.createChatCompletion(completionRequest);
System.out.println(response.getChoices().get(0).getMessage().getContent());
```

```xml
<dependency>
  <groupId>com.theokanning.openai-gpt3-java</groupId>
  <artifactId>service</artifactId>
  <version>0.18.2</version>
</dependency>
<dependency>
  <groupId>com.theokanning.openai-gpt3-java</groupId>
  <artifactId>api</artifactId>
  <version>0.18.2</version>
</dependency>
<dependency>
  <groupId>com.theokanning.openai-gpt3-java</groupId>
  <artifactId>client</artifactId>
  <version>0.18.2</version>
</dependency>
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.10.1</version>
</dependency>
```

- Considering that it's raining, it'd be a good idea to take an umbrella or a raincoat, along with waterproof shoes if you have them. Also, don't forget to carry any important documents or electronics in a waterproof bag.
- You should take an umbrella and possibly a raincoat. Also, it might be a good idea to wear waterproof shoes to keep your feet dry.

Few Shot Prompt Example

Aspect Based Sentiment Analysis On Laptop reviews

System Message

Few Shot Examples

User Input

```
{
 "model": "gpt-3.5-turbo",
 "messages": [
  {
    "role": "system",
    "content": "Perform aspect based sentiment analysis on laptop reviews presented in the input delimited by triple backticks, that is, ```.
In each review there might be one or more of the following aspects: screen, keyboard, and mousepad.
For each review presented as input:
-   Identify if there are any of the 3 aspects (screen, keyboard, mousepad) present in the review.
-   Assign a sentiment polarity (positive, negative or neutral) for each aspect
Arrange your response a JSON object with the following headers:
-   category:[list of aspects]
-   polarity:[list of corresponding polarities for each aspect]}"
      },
      {
        "role": "user",
        "content": "```The RAM is good. The design is decent.```\n"
      },
      {
        "role": "assistant",
        "content": {"category":["ram","design"],"polarity":["positive","neutral"] }
      },{
        "role": "user",
        "content": "```The screen is amazing. The design is impressive. The mousepad is bad.```\n"
      },
      {
        "role": "assistant",
        "content": {"category":["screen","design","mousepad"],"polarity":["positive","positive","negative"] }
      },
  {
    "role": "user",
    "content": "```The screen is amazing. The design is impressive. The mousepad is bad.```\n"
  }
 ],
 "temperature": 0.5
}
```

Zero Shot Prompt Example

Aspect Based Sentiment Analysis On Laptop reviews

System Message

```
{
 "model": "gpt-3.5-turbo",
 "messages": [
  {
    "role": "system",
    "content": "Perform aspect based sentiment analysis on laptop reviews presented in the input delimited by triple backticks, that is, ```.
In each review there might be one or more of the following aspects: screen, keyboard, and mousepad.
For each review presented as input:
-   Identify if there are any of the 3 aspects (screen, keyboard, mousepad) present in the review.
-   Assign a sentiment polarity (positive, negative or neutral) for each aspect
Arrange your response a JSON object with the following headers:
-   category:[list of aspects]
-   polarity:[list of corresponding polarities for each aspect]}"
      },
```

User Input

```
  {
    "role": "user",
    "content": "```The screen is amazing. The design is impressive. The mousepad is bad.```\n"
   }

 ],
 "temperature": 0.5
}
```

LLM Response

Response Content


```
{
  "id": "chatcmpl-8gXlWvFTDsBgZkh6S7jDEmo4jY3ok",
  "object": "chat.completion",
  "created": 1705150262,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": " {"category":["screen","design","mousepad"],"polarity":["positive","positive","negative"]} "
      },
      "finishReason": "stop"
    }
  ],
  "usage": {
    "promptTokens": 598,
    "completionTokens": 19,
    "totalTokens": 617
  }
}
```

Zero Shot Prompt Example

Product Review Summarisation and Sentiment Analysis

System Message

```
You are an objective assistant, part of multinational computer, phone, laptop, and hardware manufacturer,
to summarize unstructured customer reviews in simple words.

I want to generate a structured response, which can be then used to take action in a manner which optimizes
for the overall customer experience of your product. I want to capture the date of the review, product /
service in question, the rating, a short summary (upto 100 words) of the feedback for the product /
service, list of actions items which can improve the product / service, any mention of competitors - if
yes, what was better in their experience, the overall sentiment (positive, negative, or neutral). Please
generate the output in JSON format. Use this format:

{
"review_date":<If not exist, take a list of two values: The first item is it's been number and the second
item is purchased date >
"product": <extract the name of the product and its type>,
"review rating": <the rating. this has to be a number out of 5 points>,
"review_summary": <summarize the review at most in 100 words>,
"improvement_areas": <provide improvements as a list>,
"competitors": <provide a list of potential competitors>,
"overall_sentiment": <the overall sentiment (positive, negative, or neutral) >
```

Zero Shot Prompt Example

Product Review Summarization and Sentiment Analysis

LLM Response


```
{
   "review_date": "2023-08-24",
   "product": "ZephyrBook Pro 2023 Laptop",
   "review_rating": 2,
   "review_summary": "The ZephyrBook Pro 2023 has a severe heating issue that makes it uncomfortable to use for an extended period. The customer support experience was equally terrible. In comparison, the StarLite 2023 laptop is sleek, efficient, and stays cool as a cucumber.",
   "improvement_areas": ["Invest in better hardware to manage the heating issue", " Improve customer support"],
  "competitors": ["StarLite 2023 Laptop"],
  "overall_sentiment": "Negative"
}
```

Zero Shot Prompt Example

Product Review Summarisation and Sentiment Analysis

Uner Input

```
[2023-08-24] - rating 2/5 - I bought the ZephyrBook Pro 2023 with high hopes, but
it has been nothing but a disappointment. From the moment I powered it on, it felt
like a terrible investment. The heating issue is absurd; this laptop heats up
faster than a toaster. It's like having a portable oven on my lap. I can't even use
it for more than 30 minutes without feeling like my thighs are on fire. The
customer support experience has been equally horrendous. When I reached out to them
about the heating problem, they brushed me off, saying it's normal for a laptop to
get warm. Warm? No, this is scalding hot! It's like they expect me to carry a fire
extinguisher with me while using it. Now, let me tell you about the competition. My
friend owns the sleek and efficient StarLite 2023 laptop, and it's miles ahead. Not
only does it stay cool as a cucumber, but it also runs like a dream. The ZephyrBook
Pro 2023 can't hold a candle to it. My advice to the company is simple: Invest in
better hardware to manage the heating issue, and please, for the love of
technology, improve your customer support. I regret not going for the StarLite
2023, and I wish I had made a wiser choice. Don't make the same mistake I did.
```

Pricing

Pricing is per 1000 tokens in the prompt + completion

https://platform.openai.com/tokenizer

Overview Documentation API reference

tokens for the same input text.

GPT-3.5 & GPT-4 GPT-3 (Legacy)

Merci de me générer une histoire qui

Clear Show example

Tokens	Characters
13	51

Model	Input	Output
gpt-3.5-turbo-1106	$0.0010 / 1K tokens	$0.0020 / 1K tokens
gpt-3.5-turbo-instruct	$0.0015 / 1K tokens	$0.0020 / 1K tokens

Model	Input	Output
gpt-4-1106-preview	$0.01 / 1K tokens	$0.03 / 1K tokens
gpt-4-1106-vision-preview	$0.01 / 1K tokens	$0.03 / 1K tokens

Brief History of Generative AI

2014 Generated realistic images - GANs - Ian Goodfellow

2017 Breakthrough in NLP - Transformer Models

2018 Advent of large language models - BERT, GPT

2019 Advanced GANs - style GANs

2020 OpenAI released GPT-3 - this was state-of-the-art

2021 Vision Transformers - CLIP, DALL-E

2022 OpenAI released ChatGPT (GPT-3.5), DALL-E 2

2023 GPT-4, Stable Diffusion Model - market is flooded with GenAI apps

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

Multi-Head Attention

My cat can not move the bag because it is tired

0.95

0.03

Figure 1: The Transformer - model architecture.

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

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 prédictions dans le domaine X, mais le modèle cache d'autres connaissances sous exploitées 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