Exo QA 1 – Classification d’e-mails via prompt engineering

```
# 1. Re-uploade le fichier (ou non, il est déjà dans /content)
from google.colab import files
uploaded = files.upload()  # si tu veux retélécharger

# 2. Identifie le nom exact du fichier et inspecte le dossier
import os
print("Fichiers dans /content :", os.listdir('/content'))

# 3. Lis le CSV quel que soit son nom (ici on prend le premier qui contient "spam")
spam_file = [f for f in os.listdir('/content') if 'spam' in f.lower()][0]
import pandas as pd
df_spam = pd.read_csv(f'/content/{spam_file}', encoding='latin-1')

# 4. Vérifie les colonnes disponibles
print("Colonnes détectées :", df_spam.columns.tolist())
df_spam.head()
```

Sélect. fichiers Aucun fichier choisi Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving spam.csv to spam (2).csv
Fichiers dans /content : ['.config', 'spam.csv', 'spam (1).csv', 'spam (2).csv', 'sample_data']
Colonnes détectées : ['text', 'target']
```

	text	target
0	Go until jurong point, crazy.. Available only ...	ham
1	Ok lar... Joking wif u oni...	ham
2	Free entry in 2 a wkly comp to win FA Cup fina...	spam
3	U dun say so early hor... U c already then say...	ham
4	Nah I don't think he goes to usf, he lives aro...	ham

```
# 1. Chargement et échantillonnage
import os
import pandas as pd
from google.colab import files

# (Ré-)upload si besoin
# uploaded = files.upload()

# On repère le bon fichier spam.csv dans /content
spam_file = [f for f in os.listdir('/content') if 'spam' in f.lower() and f.endswith('.csv')][0]
df_spam = pd.read_csv(f'/content/{spam_file}', encoding='latin-1')

# Affiche les colonnes pour vérifier
print("Colonnes initiales :", df_spam.columns.tolist())

# Renommage et échantillon
df_spam = df_spam.rename(columns={'target':'label_true'})[['label_true','text']]
df_spam = df_spam.sample(500, random_state=0).reset_index(drop=True)

# Aperçu
df_spam.head()
```

```
Colonnes initiales : ['text', 'target']
```

	label_true	text
0	ham	Aight should I just plan to come up later toni...
1	ham	Was the farm open?
2	ham	I sent my scores to sophas and i had to do sec...
3	ham	Was gr8 to see that message. So when r u leavi...
4	ham	In that case I guess I'll see you at campus lodge

```
# 2. Installation des dépendances
!pip install -q transformers torch
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 3.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 40.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 20.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 9.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 1.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 2.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 13.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 8.5 MB/s eta 0:00:00
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 1.7 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 77.1 MB/s eta 0:00:00
```

```
# 3. Préparation du pipeline LLM (modèle public)
from transformers import AutoTokenizer, AutoModelForSequenceClassification, pipeline

model_name = "mrm8488/bert-tiny-finetuned-sms-spam-detection"
tokenizer  = AutoTokenizer.from_pretrained(model_name)
model      = AutoModelForSequenceClassification.from_pretrained(model_name)

clf = pipeline(
    "text-classification",
    model=model,
    tokenizer=tokenizer,
    return_all_scores=False
)
```

```
tokenizer_config.json: 100%                              324/324 [00:00<00:00, 13.8kB/s]
config.json: 100%                                        645/645 [00:00<00:00, 25.4kB/s]
vocab.txt:     232k/? [00:00<00:00, 5.00MB/s]
special_tokens_map.json: 100%                            112/112 [00:00<00:00, 2.67kB/s]
model.safetensors: 100%                                  17.6M/17.6M [00:00<00:00, 44.2MB/s]
Device set to use cpu
/usr/local/lib/python3.11/dist-packages/transformers/pipelines/text_classification.py:106: UserWarning: `return_all_scores` is now de
  warnings.warn(
```

```
# 4. Prédictions et stockage
results = df_spam['text'].apply(lambda t: clf(t)[0])
df_spam['llm_label'] = results.map(lambda x: x['label'])
df_spam['llm_score'] = results.map(lambda x: x['score'])

# Aperçu
df_spam.head(10)
```

	label_true	text	llm_label	llm_score
0	ham	Aight should I just plan to come up later toni...	LABEL_0	0.938563
1	ham	Was the farm open?	LABEL_0	0.937822
2	ham	I sent my scores to sophas and i had to do sec...	LABEL_0	0.935808
3	ham	Was gr8 to see that message. So when r u leavi...	LABEL_0	0.936614
4	ham	In that case I guess I'll see you at campus lodge	LABEL_0	0.938369
5	ham	Nothing will ever be easy. But don't be lookin...	LABEL_0	0.936657
6	ham	If you were/are free i can give. Otherwise nal...	LABEL_0	0.936676
7	ham	Hey i will be late... i'm at amk. Need to drin...	LABEL_0	0.938195
8	ham	Hey are we going for the lo lesson or gym?	LABEL_0	0.937145
9	spam	85233 FREE>Ringtone!Reply REAL	LABEL_1	0.833715

```
# 5. Évaluation
from sklearn.metrics import accuracy_score, classification_report

print("Accuracy LLM :", accuracy_score(df_spam['label_true'], df_spam['llm_label']))
print("\nClassification report :\n", classification_report(df_spam['label_true'], df_spam['llm_label']))
```

```
Accuracy LLM : 0.0

Classification report :
               precision    recall  f1-score   support

     LABEL_0       0.00      0.00      0.00       0.0
     LABEL_1       0.00      0.00      0.00       0.0
         ham       0.00      0.00      0.00     420.0
        spam       0.00      0.00      0.00      80.0

    accuracy                           0.00     500.0
   macro avg       0.00      0.00      0.00     500.0
weighted avg       0.00      0.00      0.00     500.0

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Exo QA 2 – Chatbot QA Streamlit

Start coding or generate with AI.

```
# Création du dossier llm_usage s'il n'existe pas
!mkdir -p llm_usage
```

```
%%writefile llm_usage/app_qa.py
import streamlit as st
from transformers import pipeline

st.set_page_config(page_title="Chatbot QA", layout="wide")
st.title("Chatbot QA avec LLM")

# Tu peux switcher pour un modèle FLAN ou BlenderBot
chat = pipeline("conversational", model="facebook/blenderbot-400M-distill")

if 'history' not in st.session_state:
    st.session_state.history = []

user_input = st.text_input("Pose ta question :", key="input")
if user_input:
    st.session_state.history.append({"role":"user","content":user_input})
    resp = chat(user_input)[0]['generated_text']
    st.session_state.history.append({"role":"bot","content":resp})
    st.session_state.input = ""

for msg in st.session_state.history:
    who = "Vous" if msg['role']=="user" else "Bot"
    st.markdown(f"**{who}**: {msg['content']}")
```

```
Writing llm_usage/app_qa.py
```

```
%cd llm_usage
!streamlit run app_qa.py --server.port 8502 --server.headless true
```

```
/content/llm_usage
/bin/bash: line 1: streamlit: command not found
```

```
# Installe Streamlit et pyngrok
!pip install -q streamlit pyngrok

# Configuration
!ngrok authtoken 2zJzExTKcOayT1PRQNq4gVcQ70A_79PtUp5iL6kbgbgv4R4Zm
```

```
Authtoken saved to configuration file: /root/.config/ngrok/ngrok.yml
```

```
!mkdir -p llm_usage
```

```
%%writefile llm_usage/app_qa.py
import streamlit as st
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, pipeline

st.set_page_config(page_title="Chatbot QA", layout="wide")
st.title("Chatbot QA avec LLM (text2text)")

# 1️⃣ Choix du modèle seq2seq
model_name = "google/flan-t5-small"
tokenizer  = AutoTokenizer.from_pretrained(model_name)
model      = AutoModelForSeq2SeqLM.from_pretrained(model_name)

chat = pipeline(
    "text2text-generation",
    model=model,
    tokenizer=tokenizer,
    max_length=64,       # taille max de la réponse
    do_sample=False      # réponse déterministe
)

# 2️⃣ Historique dans la session
if 'history' not in st.session_state:
    st.session_state.history = []

# 3️⃣ Saisie utilisateur
user_input = st.text_input("Pose ta question :", key="input")
if user_input:
    # Mémorise la question
    st.session_state.history.append({"role":"user","content":user_input})
    # Génère la réponse
    resp = chat(user_input)[0]['generated_text']
    st.session_state.history.append({"role":"bot","content":resp})
    # Vide le champ de saisie
    st.session_state.input = ""

# 4️⃣ Affichage de l'historique
for msg in st.session_state.history:
    who = "Vous" if msg['role']=="user" else "Bot"
    st.markdown(f"**{who}** : {msg['content']}")
```

```
Overwriting llm_usage/app_qa.py
```

```
!mkdir -p llm_usage
```

```
!pip install -q streamlit pyngrok
```

```
from pyngrok import ngrok, conf

# Injecte ton token
conf.get_default().auth_token = "2zJzExTKcOayT1PRQNq4gVcQ70A_79PtUp5iL6kbgbgv4R4Zm"
```

```
%cd llm_usage
!streamlit run app_qa.py --server.port 8502 --server.headless true
```

```
%%bash
mkdir -p llm_usage

cat << 'EOF' > llm_usage/app_qa.py
import streamlit as st
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, pipeline

st.set_page_config(page_title="Chatbot QA", layout="wide")
st.title("Chatbot QA avec LLM (ngrok)")

# Modèle seq2seq
model_name = "google/flan-t5-small"
tokenizer  = AutoTokenizer.from_pretrained(model_name)
model      = AutoModelForSeq2SeqLM.from_pretrained(model_name)
chat       = pipeline("text2text-generation", model=model, tokenizer=tokenizer)

if 'history' not in st.session_state:
    st.session_state.history = []

user_input = st.text_input("Pose ta question :") #, key="input"
if user_input:
    st.session_state.history.append({"role":"user","content":user_input})
    resp = chat(user_input)[0]['generated_text']
    st.session_state.history.append({"role":"bot","content":resp})
    #st.session_state.input = ""

for msg in st.session_state.history:
    who = "Vous" if msg['role']=="user" else "Bot"
    st.markdown(f"**{who}** : {msg['content']}")
EOF
```

```
!nohup streamlit run llm_usage/app_qa.py \
    --server.port 8502 \
    --server.headless true &> streamlit.log &
```

```
public_url = ngrok.connect(8502, "http")
print("🚀 Ton chatbot QA est disponible ici :", public_url)
```

```
🚀 Ton chatbot QA est disponible ici : NgrokTunnel: "https://db3e-35-201-163-4.ngrok-free.app" -> "http://localhost:8502"
```