Chapitre 1 — Exercices pratiques

Formation « AI : Module 1 - Fondations techniques des LLM » · INSSE inc.

Prérequis : Python 3.10+, pip install anthropic, clé d'API dans la variable d'environnement ANTHROPIC_API_KEY. Durée estimée : 45 à 60 minutes. Les solutions commentées suivent chaque exercice — essayez d'abord sans les regarder !

Exercice 1 — Votre premier appel, et l'effet de la température

Objectif

Manipuler les trois paramètres vus en cours (model, temperature, messages) et constater par vous-même l'effet de la température sur la stabilité des réponses.

Énoncé

- Reprenez le script du cours et faites-le fonctionner tel quel.
- Demandez au modèle : « Propose un slogan pour INSSE, société de services informatiques. » Exécutez la requête 3 fois avec temperature=0, puis 3 fois avec temperature=1.
- Notez les 6 réponses. Que constatez-vous ? Dans quel cas d'usage choisiriez-vous chaque réglage ?

Solution commentée

```
import anthropic

client = anthropic.Anthropic()  # lit ANTHROPIC_API_KEY

def slogan(temperature: float) -> str:
    reponse = client.messages.create(
        model="claude-sonnet-4-6",
        max_tokens=100,            # un slogan est court : on plafonne le coût
        temperature=temperature,
        messages=[{"role": "user", "content":
            "Propose un slogan pour INSSE, société de services informatiques."}],
    )
    return reponse.content[0].text

for t in (0.0, 1.0):
    print(f"--- temperature={t} ---")
    for i in range(3):
        print(f"{i+1}. {slogan(t)}")
```

> *Constat attendu : à température 0, les trois slogans sont identiques ou presque (le modèle choisit toujours le token le plus probable). À température 1, ils varient nettement. Réglage bas pour l'extraction et la classification ; réglage haut pour la créativité. Rappel du cours : même à 0, le déterminisme absolu n'est pas garanti sur toutes les plateformes.*

Exercice 2 — Extraction structurée à température 0

Objectif

Mettre en pratique le cas d'usage « analyste » par excellence : transformer un texte libre en données structurées fiables, et valider la sortie comme un développeur prudent.

Énoncé

- Donnez au modèle ce texte : « Facture FA-2026-0612 émise le 3 juin 2026 par Solutions Nordiques inc. Montant : 12 450,00 $ CA, payable sous 30 jours. »
- Demandez-lui d'en extraire le numéro, la date, le fournisseur et le montant, au format JSON strict, sans aucun texte autour.
- Validez la réponse avec json.loads() et affichez un message d'erreur clair si le JSON est invalide.
- Question bonus : pourquoi temperature=0 est-il indispensable ici ?

Solution commentée

```
import anthropic, json

client = anthropic.Anthropic()

TEXTE = ("Facture FA-2026-0612 émise le 3 juin 2026 par Solutions "
         "Nordiques inc. Montant : 12 450,00 $ CA, payable sous 30 jours.")

PROMPT = f"""Extrais de ce texte les champs suivants et réponds
UNIQUEMENT avec un objet JSON valide, sans texte autour :
numero (str), date_emission (AAAA-MM-JJ), fournisseur (str),
montant_cad (float).

Texte : {TEXTE}"""

reponse = client.messages.create(
    model="claude-sonnet-4-6",
    max_tokens=200,
    temperature=0,              # extraction : on veut du stable et factuel
    messages=[{"role": "user", "content": PROMPT}],
)
brut = reponse.content[0].text.strip()

try:
    donnees = json.loads(brut)
    print("Extraction réussie :", donnees)
except json.JSONDecodeError:
    print("ERREUR : le modèle n'a pas renvoyé un JSON valide :")
    print(brut)
```

> *Points pédagogiques : (1) la consigne « UNIQUEMENT un objet JSON » réduit le risque de bavardage autour de la réponse ; (2) on ne fait JAMAIS confiance aveuglément à la sortie — json.loads est notre filet de sécurité, première application concrète de « le LLM propose, l'humain (ou son code) dispose » ; (3) température 0 car deux extractions de la même facture doivent donner le même résultat.*

Exercice 3 — Toucher les limites du doigt

Objectif

Vérifier expérimentalement deux limites structurelles vues en cours : le calcul « deviné » et la date de coupure des connaissances.

Énoncé

- Demandez au modèle (temperature=0, sans lui dire que c'est un test) : « Combien font 7 387 × 4 213 ? Réponds seulement par le nombre. » Vérifiez ensuite le résultat en Python.
- Demandez-lui ensuite : « Quelle est la dernière version de Python et sa date de sortie ? » Comparez avec python.org.
- Concluez : pour chacun des deux cas, quelle est la parade vue en cours ?

Solution commentée

```
import anthropic

client = anthropic.Anthropic()

def demander(question: str) -> str:
    r = client.messages.create(
        model="claude-sonnet-4-6", max_tokens=200, temperature=0,
        messages=[{"role": "user", "content": question}],
    )
    return r.content[0].text.strip()

# 1) Le calcul : le modèle 'devine' le résultat comme du texte
reponse_calcul = demander("Combien font 7387 x 4213 ? Réponds seulement par le
nombre.")
vrai_resultat = 7387 * 4213    # Python, lui, calcule vraiment
print(f"Modèle : {reponse_calcul} | Python : {vrai_resultat} | "
      f"Correct : {reponse_calcul.replace(' ', '') == str(vrai_resultat)}")

# 2) La date de coupure
print(demander("Quelle est la dernière version de Python et sa date de sortie ?"))
```

> *Conclusions attendues : le grand produit arithmétique est souvent faux (parfois proche, ce qui est pire : plausible !) — parade : faire appeler une calculatrice ou exécuter du code, c'est le tool calling du chapitre 6. La version de Python citée peut être périmée — parade : recherche web ou documents fournis dans le contexte, c'est le RAG du chapitre 5. Si le modèle donne la bonne réponse aux deux questions, tant mieux — mais auriez-vous pu le savoir sans vérifier ? C'est exactement le point.*

Pour aller plus loin (facultatif)

- Reprenez l'exercice 2 et ajoutez un champ piège absent du texte (ex. : numero_bon_commande). Le modèle l'invente-t-il ou renvoie-t-il null ? Ajustez le prompt pour exiger null quand l'information est absente.
- Mesurez le coût : affichez reponse.usage (tokens d'entrée et de sortie) pour chaque appel et estimez le prix d'un traitement de 10 000 factures.

Bon travail ! Les solutions seront discutées en ouverture du chapitre 2. — Le professeur