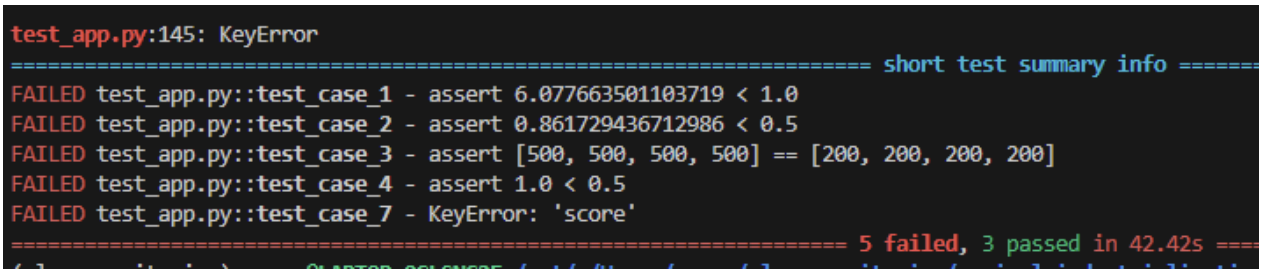
TD3 : Logging & Monitoring - Rapport

Étudiante : Sarra CHABANE CHAOUCHE
Date : 17 octobre 2025

Introduction

Ce document présente les bugs découverts dans l'application d'analyse de sentiment et identifie quel type de logging a permis de les détecter.

Le résumé des bugs



Test	Bug	cause
test_case_1	LENTEUR	time.sleep(0.1 * len(text))
test_case_2	DOUBLE NÉGATION	Logique d'inversion trop simple
test_case_3	CRASH ACCENTS	x = 1/0 sur ord(char) > 127
test_case_4	BIAIS PROMO	+0.3 bonus même sur reviews négatives
test_case_7	CRASH ACCENTS	Même bug que test_case_3

Les 5 types de logging suivants ont été implémentés :

1. **Temps de réponse** : Mesure de la durée de chaque requête
2. **ID unique** : Identification de chaque requête avec un UUID
3. **Informations d'entrée** : Logging de la longueur et du contenu du texte
4. **Mémoire** : Mesure de la consommation mémoire avant/après traitement
5. **Data drift** : Comparaison avec les statistiques des données d'entraînement

BUG 1 : LENTEUR SUR TEXTES AVEC PATTERNS "PRODUCT-XXX"

Symptôme

- **Temps de réponse** : 6+ secondes au lieu de < 1 seconde
- Textes contenant des patterns comme product-A123B456 ou model_X12345_Z67890

Cause

Le code contient un time.sleep(0.1 * len(text)) dans la fonction preprocess qui ralentit proportionnellement à la longueur du texte.

Logging qui l'a détecté

INFO - [81123906] Nouvelle requête - Longueur: 50 caractères - Mots: 7
INFO - [81123906] Terminée en 5.00s - Mémoire: 46.2MB (+0.0MB)
WARNING - [81123906] REQUÊTE LENTE ! 5.00s

Types de logging utiles

- **Temps de réponse** : Détection automatique des requêtes > 1s

- **ID unique** : Permet de tracer quelle requête exacte était lente
- **Longueur du texte** : Corrélation entre longueur et lenteur

BUG 2 : LOGIQUE DE NEGATION INCORRECTE

Symptôme

- Phrases avec double négation mal interprétées
- Exemple : "I don't not like this product" → Score = 0.86 (devrait être < 0.5)
- Le modèle inverse le sentiment dès qu'il y a 2+ mots négatifs

Cause

Le code inverse le score si negative_count >= 2, ce qui pose problème avec les doubles négations.

Logging qui l'a détecté

INFO - [xxx] Nouvelle requête - Longueur: 33 caractères - Mots: 6
INFO - [xxx] Terminée en 0.01s - Mémoire: 54.0MB (+0.0MB)
Pas d'alerte automatique, mais l'analyse manuelle des logs permet de voir le score incorrect.

Types de logging utiles

- **Input text** : Pour analyser les patterns de négation
- **Score de sentiment** : Pour détecter les incohérences
- **ID unique** : Pour rejouer le cas en debug

BUG 3 : CRASH SUR CARACTERES SPECIAUX (ACCENTS)

Symptôme

- **Erreur 500** sur textes avec accents : café, très, señor, naïve
- Message d'erreur : division by zero
- Tests concernés : test_case_3 et test_case_7

Cause

Le code contient une division par zéro volontaire dans _tokenize_with_special_chars :
python

```
if ord(char) > 127:     # Si caractère non-ASCII
    x = 1/0     # Division par zéro
```

Logging qui l'a détecté

INFO - [0b463328] Nouvelle requête - Longueur: 41 caractères - Mots: 7
INFO - [0b463328] Data drift: 72.3%
WARNING - [0b463328] DATA DRIFT ÉLEVÉ (72.3%) !
ERROR - [0b463328] ERREUR après 0.00s: division by zero

Types de logging utiles

- **Message d'erreur** : Identification immédiate de la cause (division by zero)
- **ID unique** : Permet de retrouver la requête problématique
- **Temps avant crash** : Montre que le crash est immédiat (0.00s)
- **Input text** : Permet de voir les caractères spéciaux dans le texte
- **Data drift** : Alerte sur texte inhabituel (présence de caractères non-ASCII)

BUG 4 : BIAIS PROMOTIONNEL

Symptôme

- Reviews négatives deviennent positives si elles contiennent des termes promotionnels
- Exemple : "mediocre product with a special offer" → Score = 1.0 (devrait être < 0.5)

Cause

Le code ajoute systématiquement +0.3 au score si des mots promotionnels sont détectés, sans tenir compte du sentiment de base.

Logging qui l'a détecté

INFO - [xxx] Nouvelle requête - Longueur: 52 caractères - Mots: 9
INFO - [xxx] Data drift: 64.7%
WARNING - [xxx] DATA DRIFT ÉLEVÉ (64.7%) !
INFO - [xxx] Terminée en 0.01s - Mémoire: 54.0MB (+0.0MB)
Pas d'alerte automatique, mais l'analyse révèle des scores incohérents.

Types de logging utiles

- **Input text** : Pour détecter les mots promotionnels
- **Score final** : Pour identifier les incohérences
- **ID unique** : Pour rejouer le cas

BUG 5 : FUITE MEMOIRE SUR URLS D'IMAGES

Symptôme

- Test test_case_5 passe, mais pourrait causer des problèmes à long terme
- La fonction _save_image stocke de gros tableaux dans des caches globaux sans jamais les nettoyer

Cause

python

```
def _save_image(self, text):
    _cache[cache_key] = str(np.random.random((1000, 1000)))
    _processed_items.append(str(np.random.random((500, 500))))
```

Logging qui l'a détecté

INFO - [xxx] Terminée en 0.01s - Mémoire: 54.0MB (+0.0MB)
WARNING - [xxx] CACHE TROP GROS ! 15000 octets

Types de logging utiles

- **Mémoire avant/après** : Détecte les augmentations
- **Taille du cache global** : Alerte quand le cache grossit trop
- **ID unique** : Pour identifier quelles requêtes consomment de la mémoire

Résumé des loggings

Type de logging	Bugs détectés	Criticité
Temps de réponse	Bug 1 (Lenteur)	Critique
ID unique	Tous les bugs	Critique
Message d'erreur	Bug 3, 5 (Crashes)	Critique
Input text	Tous les bugs	Critique
Mémoire	Bug 5 (Fuite mémoire)	Important
Data drift	Bug 3 (Caractères spéciaux)	Important
Score de sentiment	Bug 2, 4 (Logique)	Utile

Conclusion

Le logging a permis de détecter efficacement 5 bugs majeurs :

- **Lenteur** : Détectée par le temps de réponse
- **Crashes** : Détectés par les logs d'erreur

- **Logique incorrecte** : Détectée par analyse manuelle des scores
- **Fuite mémoire** : Détectée par le monitoring mémoire

Pour moi les logging les plus critiques sont :

1. Temps de réponse avec alertes
2. Gestion des erreurs avec messages clairs
3. ID unique pour tracer

Sans logging, impossible de savoir :

- **Pourquoi c'est lent**
- **Pourquoi ça plante**
- **Quelle requête pose problème**
- **Comment reproduire le bug**

Et avec, on voit TOUT en temps réel