

Anticipez les "bad buzz" avec l'analyse de sentiments et le MLOps

Dans un monde hyperconnecté, les réseaux sociaux jouent un rôle crucial dans la réputation des entreprises. Les crises d'image, ou "bad buzz", peuvent avoir des conséquences importantes sur la crédibilité et les résultats commerciaux des entreprises. Grâce à l'intelligence artificielle (IA) et au Machine Learning Operations (MLOps), il est désormais possible de détecter ces signaux faibles en temps réel. Cet article explore la mise en œuvre d'une solution complète d'analyse de sentiments appliquée à des tweets, intégrant des principes MLOps pour garantir robustesse, efficacité et évolutivité.

Projet disponible sous : https://github.com/preudh/OC_IA_P7_analyse_sentiments_deep_Learning

Objectif du projet

Ce projet vise à développer un système capable de prédire les sentiments exprimés dans des tweets, afin d'aider une compagnie aérienne, "Air Paradis", à anticiper les crises potentielles. Il intègre des approches de modélisation avancées et met en œuvre des pratiques MLOps pour automatiser le cycle de vie des modèles en production.

Comparaison des trois approches

Trois approches principales ont été développées et comparées dans ce projet.

1. Régression logistique (TF-IDF)

Description :

La régression logistique a été utilisée comme baseline pour ce projet. Les tweets ont été transformés en vecteurs TF-IDF (Term Frequency-Inverse Document Frequency) afin de représenter les données textuelles. Cette méthode attribue un poids à chaque terme en fonction de sa fréquence dans le document et de sa rareté dans le corpus global.

Méthodologie :

- 1. Prétraitement des tweets :**
Les tweets ont été nettoyés à l'aide de techniques classiques comme la lemmatisation et la suppression des stop words.
- 2. Vectorisation TF-IDF :**
Transformation des tweets en vecteurs numériques, en limitant la matrice aux **10 000 termes les plus fréquents**.
- 3. Entraînement du modèle :**
Un modèle de régression logistique a été entraîné sur un ensemble d'entraînement équilibré.
- 4. Évaluation des performances :**
Les performances du modèle ont été évaluées à l'aide de métriques standard (accuracy, F1-score) et d'une matrice de confusion.

Visualisation des performances :

1. Rapport de classification dans le notebook :

Ce rapport détaille la précision, le rappel et le F1-score pour les classes "Positif" et "Négatif". Il montre une légère supériorité pour la classe positive en termes de rappel.

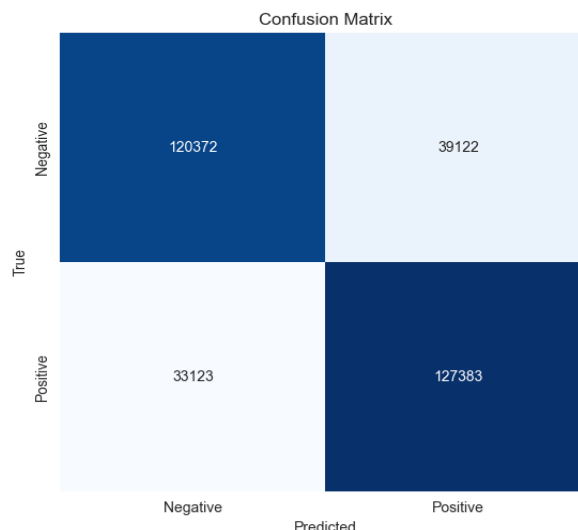
```
# Évaluation des performances du modèle
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

[7]

	precision	recall	f1-score	support
0	0.78	0.75	0.77	159494
1	0.77	0.79	0.78	160506
accuracy			0.77	320000
macro avg	0.77	0.77	0.77	320000
weighted avg	0.77	0.77	0.77	320000

Le modèle atteint une **accuracy globale de 77 %** et un **F1-score moyen pondéré de 0.77**. Ces résultats démontrent une capacité raisonnable à classer les tweets, bien que la méthode soit limitée par son incapacité à capturer le contexte sémantique.

2. Matrice de confusion :



La matrice de confusion montre :

- **120 372 tweets négatifs** correctement classés comme négatifs.
- **127 383 tweets positifs** correctement classés comme positifs.
- **39 122 erreurs** sur les tweets négatifs, prédits comme positifs.
- **33 123 erreurs** sur les tweets positifs, prédits comme négatifs.

Ces erreurs traduisent une certaine limite dans la capacité du modèle à généraliser correctement sur des cas ambigus ou inhabituels.

Points forts :

- **Rapidité d'entraînement et simplicité**
- **Simplicité d'implémentation** : Une approche standard largement utilisée pour les tâches de classification textuelle.

Limites :

- **Absence de contexte sémantique** : Le modèle ne tient pas compte de l'ordre des mots ni des relations contextuelles.
- **Performances moyennes** : Bien que raisonnables, les résultats peuvent être améliorés avec des méthodes plus avancées (embeddings ou transformers).

2. Embeddings GloVe et FastText avec LSTM

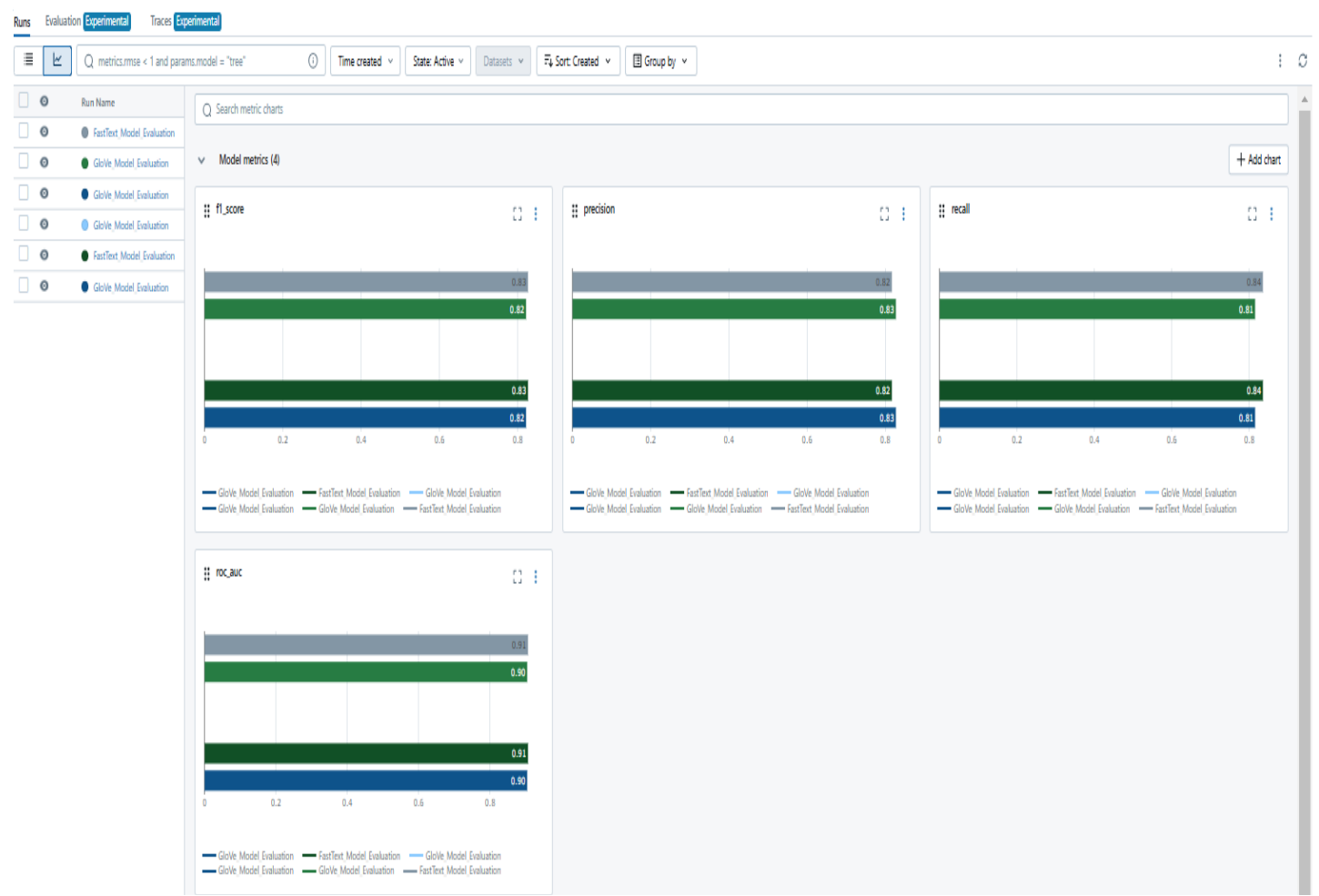
- **Description** : Les embeddings **GloVe** et **FastText** ont été utilisés pour convertir les tweets en représentations vectorielles contextuelles. Pour exploiter ces plongements de mots, un réseau neuronal récurrent **LSTM (Long Short-Term Memory)** a été entraîné.
- **Méthodologie** :
 1. Chargement des embeddings pré-entraînés :
 - **GloVe** : 100 dimensions.
 - **FastText** : 300 dimensions.
 2. Transformation des tweets en vecteurs en utilisant les embeddings.
 3. Construction d'un modèle avec une couche LSTM suivie d'une couche dense.
 4. Entraînement des modèles sur un ensemble d'entraînement équilibré.
 5. Évaluation des performances sur un ensemble de test.
- **Résultats** :
 1. Points forts : Les LSTM associés à FastText capturent efficacement les relations contextuelles.
 2. Limites : Temps d'entraînement long et ressources nécessaires (GPU recommandé).

Les performances des modèles GloVe et FastText combinés avec LSTM ont été suivies et comparées à l'aide de **MLflow**, un outil de tracking des expérimentations.

La capture ci-dessous montre les métriques principales pour chaque expérimentation : **F1-score**, **Précision**, **Rappel**, et **ROC-AUC**. On peut observer que le modèle FastText associé à LSTM surpasse légèrement GloVe sur tous les critères, avec une **précision et un rappel de 81%** et une **ROC-AUC atteignant 0.91**.

- ### Experiments
- Search Experiments
- ☐ BERT_Embedding_Experiment
 - ☐ Logistic_Regression_TFIDF_Experiment
 - ☒ FastText_Embedding_Experiment
 - ☒ GloVe_Embedding_Experiment

Displaying Runs from 2 Experiments



3. Fine-tuning de BERT (bert-base-uncased)

- **Description :** Le modèle **BERT (Bidirectional Encoder Representations from Transformers)** a été fine-tuné sur un sous-échantillon stratifié de 1 000 tweets. L'utilisation de la version pré-entraînée "bert-base-uncased" permet de capturer le contexte global des phrases grâce à une architecture transformer bidirectionnelle.
- **Méthodologie :**
 1. Prétraitement des tweets avec le tokenizer de Hugging Face.
 2. Transformation des tweets en vecteurs de tokens avec padding et troncature.
 3. Fine-tuning de BERT sur les données annotées (0 : négatif, 1 : positif).
 4. Évaluation des performances à l'aide des métriques d'accuracy et de F1-score.
- **Résultats :**
 1. Points forts : Bonne précision globale et compréhension fine du contexte pour un modèle BERT. Performances solides sur les deux classes avec des scores équilibrés.
 2. Limites : Temps d'entraînement et coût computationnel élevés, typiques de BERT.

← → ↻ 127.0.0.1:5000/#/experiments/835664034872152055/runs/07dc96c116514a4587eb0700a4d89668 🔍 ☆

traduction anglais fr... Réalisez vos premiè... Tableau de bord ... Déconnecté de Trello (20) OpenClassroo... Discord Python (85) Webinaire - co... Installez WampServ... Application Monitor... Instalacion Visual St... P5_Download and L... » Tous les favoris

mlflow 2.17.0 Experiments Models ⚙️ GitHub Docs

BERT_Embedding_Experiment > **awesome-grouse-454** ⋮ Register model

Overview Model metrics System metrics Artifacts

Description ⓘ
No description

Details

Created at	2024-10-22 08:04:53
Created by	pat
Experiment ID	835664034872152055 🔗
Status	🟢 Finished
Run ID	07dc96c116514a4587eb0700a4d89668 🔗
Duration	24.7s
Datasets used	—
Tags	Add
Source	🏠 C:\Users\pat\conda\envs\p7te\Lib\site-packages\pykernel_launcher.py
Logged models	🔗 keras
Registered models	—

Parameters (2)

Parameter	Value
epochs	3
model	BERT

Metrics (2)

Metric	Value
accuracy	0.8687499761581421
val_accuracy	0.7950000166893005

Cette capture d'écran MLflow montre une exécution terminée avec succès, associée à un modèle BERT entraîné sur 3 époques, et utilisant Keras. Les performances affichent une précision (*accuracy*) de 86.87 % et une validation (*val_accuracy*) de 79.5 %, indiquant un bon entraînement avec un écart modéré. L'exécution a duré 24.7 secondes, et les informations telles que le script source et l'identifiant de l'expérience sont enregistrées. Il est possible de sauvegarder ce modèle via l'option **Register model**. Aucun dataset n'est spécifié ici, mais peut être ajouté pour une meilleure traçabilité.

Comparaison des résultats

Approche	Accuracy	F1-score
Régression logistique	0.770	0.770
GloVe + LSTM	0.830	0.820
FastText + LSTM	0.820	0.830
BERT (bert-base-uncased)*	0.800	0.790

*

Différences d'accuracy :

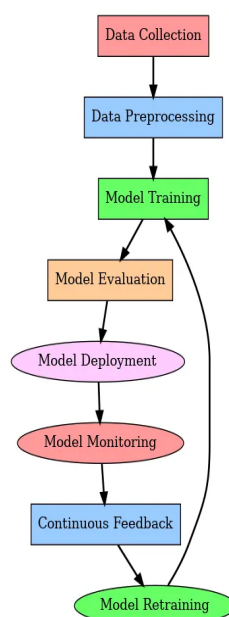
L'accuracy dans MLflow (0.8687) reflète les performances sur le jeu d'entraînement, tandis que celle du notebook (0.80) est une évaluation indépendante sur le jeu de test, plus réaliste. La `val_accuracy` (0.795) indique les performances sur le jeu de validation pendant l'entraînement pour surveiller la généralisation du modèle.

D'après mlflow, les valeurs sont similaires, mais FastText montre une légère supériorité sur le F1-Score et le ROC AUC. Cela indiquerait que FastText est potentiellement le meilleur modèle sur mesure avancé, **nous allons donc retenir ce modèle pour la suite de nos travaux et l'utilisation dans le cadre de l'API** qui sera développée pour la prédiction de la polarité des tweets avec FastText, FastAPI dans un premier temps en local,

Mise en œuvre orientée MLOps

La mise en production de modèles de machine learning nécessite des pratiques robustes pour garantir leur fiabilité, évolutivité et maintenabilité. Le MLOps s'impose comme une discipline clé pour automatiser le cycle de vie des modèles. Grâce à cette pipeline CI/CD configurée, il suffit de **committer et pusher sur GitHub** pour déclencher automatiquement la chaîne de déploiement, qui inclut les étapes de tests et de mise en production sur AZURE.

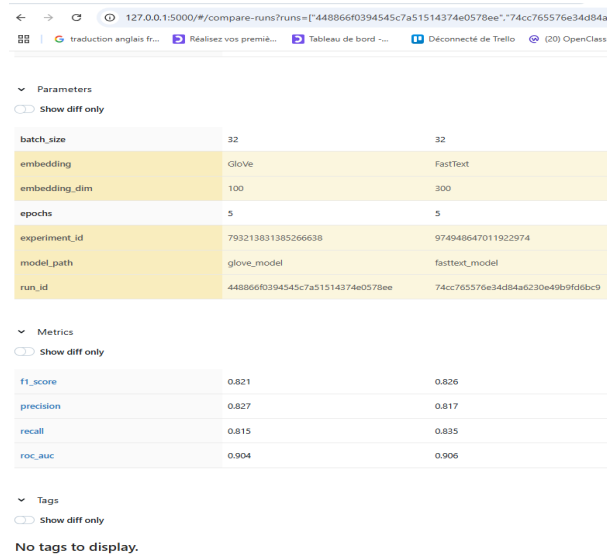
Cycle de vie Mlops : Le schéma ci-dessous illustre le cycle de vie standard du MLOps, de la collecte des données au réentraînement continu des modèles.



Principes MLOps appliqués

1. Traçabilité et suivi des expérimentations :

- Dans le cadre de ce projet, **MLFlow** a été configuré en local pour :
- Suivre les hyperparamètres et les performances des modèles.



Parameter	Run 1 (448866f0394545c7a51514374e0578ee)	Run 2 (74cc765576e34d84a6230e49b9fd6bc9)
batch_size	32	32
embedding	GloVe	FastText
embedding_dim	100	300
epochs	5	5
experiment_id	793213831385266638	974948647011922974
model_path	glove_model	fasttext_model
run_id	448866f0394545c7a51514374e0578ee	74cc765576e34d84a6230e49b9fd6bc9

Metric	Run 1 (448866f0394545c7a51514374e0578ee)	Run 2 (74cc765576e34d84a6230e49b9fd6bc9)
f1_score	0.821	0.826
precision	0.827	0.817
recall	0.815	0.835
roc_auc	0.904	0.906

No tags to display.

- Versionner les artefacts, tels que les modèles entraînés, pour garantir la reproductibilité des expérimentations.

Perspectives :

Pour un travail collaboratif et une meilleure intégration en production, MLFlow pourrait être configuré sur **Azure**. Cela permettrait :

- De centraliser les données des expérimentations pour une équipe.
- De stocker les artefacts dans Azure Blob Storage.
- D'exploiter une base de données Azure SQL pour une traçabilité robuste.

2. Gestion des versions et déploiement :

- Automatisation avec GitHub Actions et conteneurisation via Docker.

Workflow GitHub Actions montrant une exécution réussie (tests unitaires et déploiement sur Azure).

preudh / OC_IA_P7_analyse_sentiments_deep_Learning

<> Code Issues Pull requests **Actions** Projects Security Insights Settings

← CI/CD Pipeline with Build and Deploy

✓ **docker compose ci #49**

Summary

Jobs

- ✓ test
- ✓ build-and-deploy

Run details

- Usage
- Workflow file

Triggered via push 5 days ago

preudh pushed · 448ba0a master

Status: **Success**

Total duration: **4m 16s**

Artifacts: —

deploy.yml

on: push

test 1m 25s → build-and-deploy 2m 32s

Extrait du fichier `deploy.yml` décrivant la configuration du pipeline CI/CD :

preudh / OC_IA_P7_analyse_sentiments_deep_Learning

<> Code Issues Pull requests Actions Projects Security Insights Settings

Files

master + 🔍

Go to file

- github/workflows
 - deploy.yml**
- .idea
- api
- data
- models
- notebooks
 - .dockerignore
 - .gitignore
 - Dockerfile
 - README.md
 - commandes.txt
 - docker-compose.ci.yml
 - docker-compose.dev.yml
 - docker-compose.prod.yml

OC_IA_P7_analyse_sentiments_deep_Learning / .github / workflows / deploy.yml

preudh Add Azure Application Insights integration file deploy ✖

Code Blame 81 lines (69 loc) · 2.36 KB

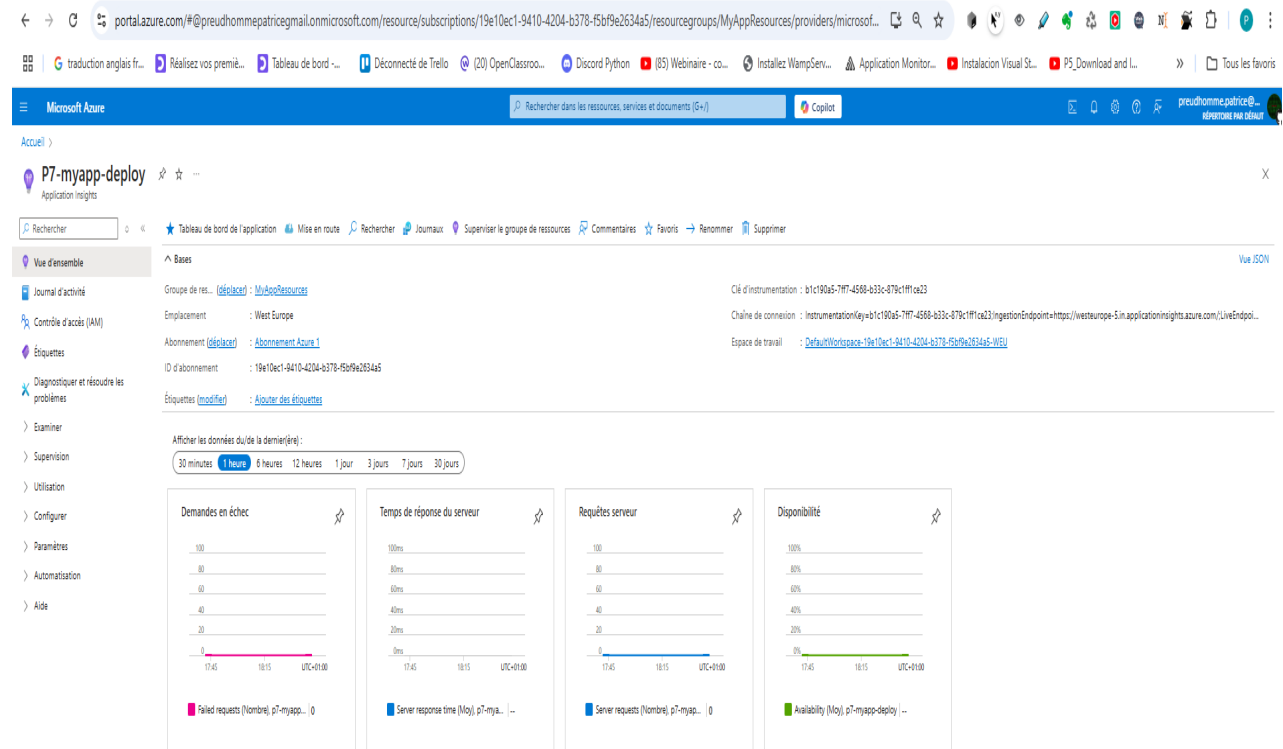
```

1  name: CI/CD Pipeline with Build and Deploy
2
3  on:
4    push:
5      branches: '*'
6    workflow_dispatch:
7
8  permissions:
9    id-token: write
10   contents: read
11   actions: write # Permissions nécessaires pour les actions GitHub
12
13  jobs:
14    # Job de test
15    test:
16      runs-on: ubuntu-latest
17      steps:
18        - name: Checkout code
19          uses: actions/checkout@v4
20
21        - name: Set up Docker Buildx
22          uses: docker/setup-buildx-action@v2

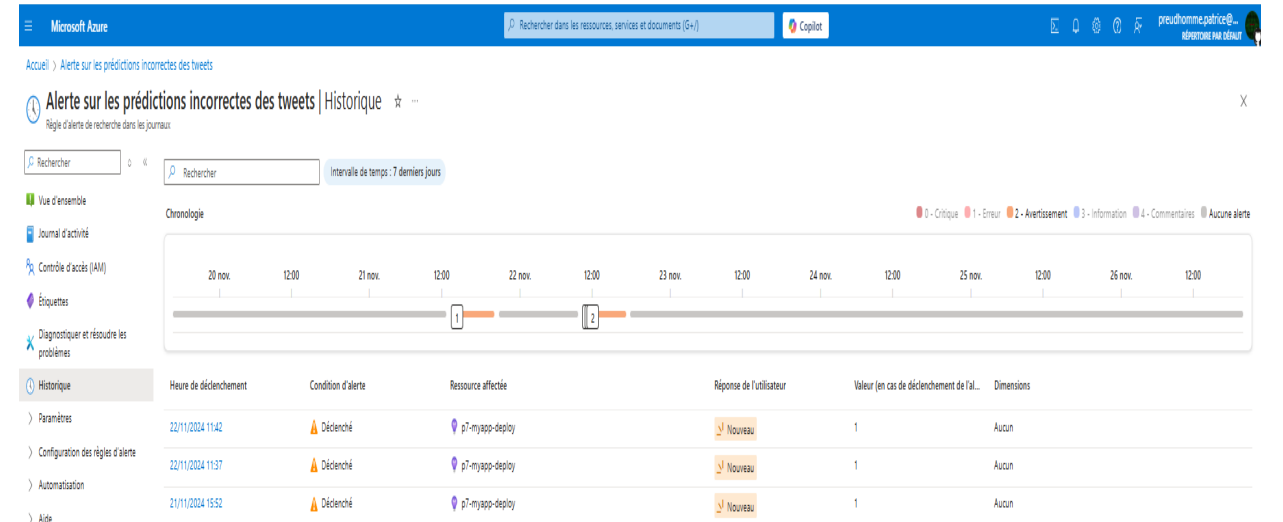
```

3. Monitoring en production :

Monitoring global des performances et de la disponibilité de l'application.



Capture d'une alerte déclenchée en cas de dérive (par exemple, plusieurs prédictions incorrectes en un temps court).



Conclusion et perspectives

Ce projet démontre la puissance de l'analyse de sentiments combinée aux pratiques MLOps pour anticiper les crises sur les réseaux sociaux. Grâce aux pratiques de tracking, monitoring et déploiement automatisé, la solution est scalable et maintenable.

Perspectives d'amélioration :

Suivi des métriques en production pour détecter les dérives :

Une stratégie de surveillance des dérives pourrait être mise en place en exploitant **Azure Application Insights** pour analyser les tendances des prédictions en temps réel. Voici une proposition :

- **Mesure des ratios :**
 - Analyser les 20 dernières prédictions (tweets positifs et négatifs).
 - Comparer ces ratios avec ceux de la base d'apprentissage pour détecter des écarts significatifs.
- **Déclenchement d'alertes :**
 - Si un écart important est constaté entre les ratios observés et ceux de la base d'apprentissage, une alerte serait générée.
- **Constatation de dérives :**
 - Par exemple, si la base d'apprentissage contient 60 % de tweets positifs, mais que les prédictions récentes montrent 90 % de tweets positifs, cela pourrait indiquer un problème potentiel, comme une dérive des données ou une dégradation du modèle.

Cette approche simple, basée sur des statistiques des prédictions, permettrait une identification précoce des dérives sans nécessiter une infrastructure complexe.