Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Tunis, École nationale supérieure d'ingénieurs de Tunis (ENSIT)

Mini Projet: Nouvelles Architectures

Date de remise : 11 décembre 2025

Section : 3GInfo- GLID & NTS

Mini-Projet MLOps : CallCenterAI – Classification intelligente des tickets clients

??🎯 Objectif

Mettre en place une solution MLOps complète pour classer automatiquement les tickets clients d'un centre d'appel (emails, chat, téléphone) en différentes catégories métiers (ex. *Facturation*, *Problème technique*, *Accès compte*, etc.), en utilisant deux approches NLP :

- Modèle classique : TF-IDF + SVM
- Modèle avancé : Transformer (Hugging Face)

Le système doit être dockerisé, monitoré, et intègre un pipeline CI/CD ainsi qu'un agent IA pour orchestrer les prédictions.

✅✅ Livrables attendus

- Un dépôt GitHub structuré avec :
  - Code d'entraînement (TF-IDF + SVM, fine-tuning Transformer)
  - Services API (FastAPI) pour chaque modèle + agent IA
  - Dockerfiles + docker-compose.yml
  - Pipeline MLOps (DVC, MLflow)
  - CI/CD (GitHub Actions)
  - Monitoring (Prometheus + Grafana)
  - Tests unitaires et d'intégration
- README clair avec instructions de lancement
- Rapport MLflow (métriques, hyperparamètres)
- Dashboard Grafana avec métriques des services

⚒ Technologies imposées

- Langage : Python 3.11
- Framework API : FastAPI
- Modèles NLP :
    - TF-IDF + SVM (scikit-learn)
    - Transformer (Hugging Face transformers)
- MLOps :
    - MLflow : suivi des expériences + registry
    - DVC : gestion des données et pipeline
    - Docker & Docker Compose : conteneurisation
    - CI/CD : GitHub Actions (tests, lint, build, push images)
    - Monitoring : Prometheus + Grafana
- Qualité & Sécurité :
    - Tests (pytest)
    - Lint (black, flake8, isort)
    - Scan sécurité (Trivy, Bandit)

🔍 Étapes du projet

Kaggle – IT Service Ticket Classification

- Taille : ~47 000 tickets
- Colonnes : Document (texte du ticket), Topic_group (catégorie)
- Catégories : Hardware, HR Support, Access, Miscellaneous, Storage, Purchase, etc. • Lien : IT Service Ticket Classification Dataset[1]

Phase 2 : Entraînement des modèles

- TF-IDF + SVM :
    - Pipeline scikit-learn avec TfidfVectorizer + LinearSVC
    - Calibration pour obtenir des probabilités
    - Log des métriques (accuracy, F1) dans MLflow
- Transformer :
    - Utiliser un modèle multilingue (distilbert-base-multilingual-cased ,Multilingue (104 langues, dont FR et AR)). C'est la version que tu dois utiliser si tu veux traiter FR/EN/AR dans CallCenterAI.
    - fine-tuning sur la dataset

Phase 3 : Services API

FastAPI est le meilleur choix pour ce projet, car il est :

- Rapide (basé sur Starlette + Pydantic)
- Facile à dockeriser
- Compatible avec Prometheus pour le monitoring
- Idéal pour microservices MLOps

- Service TF-IDF :
    - Endpoint /predict → labels + probas
    - Endpoint /metrics → Prometheus (Prometheus est un outil open source de monitoring et d'alerte conçu pour collecter et stocker des métriques (indicateurs chiffrés) provenant de vos applications et services)
- Service Transformer :
    - Endpoint /predict → labels + scores
- Agent IA :
    - Routage intelligent (basé sur confiance TF-IDF, longueur texte, langue) ○ Explication du choix
    - Scrub PII avant envoi (Scrub PII signifie supprimer ou masquer les informations personnelles identifiables (Personally Identifiable Information) dans un texte avant de le traiter ou de l'envoyer à un modèle)

Phase 4 : Conteneurisation

- Créer un Dockerfile pour chaque service
- Créer un docker-compose.yml pour lancer :
    - agent, tfidf_svc, transformer
    - mlflow, prometheus, grafana

Phase 5 : MLOps

- DVC :
    - Définir pipeline dvc.yaml (prepare → train_tfidf)
- MLflow :
    - Tracking des runs
    - Model registry (Production/Staging)
- CI/CD :
    - Workflow GitHub Actions :
    - Lint + tests
    - Build & push images Docker
    - Scan sécurité (Trivy)

- Monitoring :
    - Config Prometheus pour scrapper /metrics
    - Dashboard Grafana (latence, requêtes, erreurs)

Phase 6 : Tests & Validation

- Tests unitaires (training, API)
- Tests d'intégration (API contract)
- Vérification CI/CD verte

⭐🌟 Critères d'évaluation

- Architecture MLOps complète (30%)
- Qualité du code & CI/CD (20%)
- Performance des modèles (20%)
- Observabilité & monitoring (15%)
- Documentation & clarté (15%)

Dans ce mini-projet, **l'agent IA joue le rôle d'un chatbot conteneurisé** (déployé comme un microservice dans Docker) qui :

- **Reçoit les tickets clients** via une API REST (ou interface simple).
- **Nettoie les données sensibles** (scrub PII).
- **Analyse la requête** (langue, longueur, complexité).
- **Décide quel modèle utiliser** :
    - **TF-IDF + SVM** pour les cas simples et rapides.
    - **Transformer** pour les cas complexes ou multilingues.
- **Retourne la prédiction** (catégorie, confiance) avec une **explication du choix**.
- **Expose des métriques Prometheus** pour le monitoring.

En résumé, c'est un **chatbot intelligent** qui ne se contente pas de répondre, mais **oriente dynamiquement la requête** vers le bon moteur NLP, tout en étant intégré dans une architecture MLOps complète.