Cahier des Charges : Application de Matching Factures/Relevé

Auteurs / Équipe Projet

- Sarri Imad Eddine
- MBOUNGOU MBOYO Marcelin

1. Objectif du Projet

Concevoir et implémenter une application web simple permettant de matcher automatiquement des factures (fournies sous forme d'images) avec les lignes de transactions d'un relevé bancaire (fourni au format CSV). L'application doit identifier la facture la plus probable pour chaque transaction bancaire pertinente.

2. Public Cible

Utilisateurs individuels ou petites entreprises souhaitant automatiser et simplifier le suivi de leurs dépenses et la justification de leurs transactions bancaires.

3. Fonctionnalités Attendues

- **Interface Utilisateur (Streamlit) :**
  - Zone d'upload pour un fichier de relevé bancaire au format CSV.
  - Zone d'upload pour une ou plusieurs images de factures (formats JPG, PNG, JPEG).
  - Affichage d'un aperçu des données chargées (premières lignes du relevé, liste des factures).
  - Bouton pour lancer le processus d'analyse et de matching.
  - Affichage des résultats du matching sous forme de tableau clair et trié.
  - Bouton pour télécharger les résultats du matching au format CSV.
  - Indication du statut des services externes (API Mistral, Modèle d'embedding).
  - Affichage de messages d'information, d'avertissement ou d'erreur clairs pour l'utilisateur.
- **Traitement Backend :**
  - Lecture et parsing robuste du fichier CSV (gestion des séparateurs, nettoyage des noms de colonnes).
  - Conversion et validation des types de données clés du CSV (montants en float, dates en datetime avec gestion des erreurs).
  - Extraction structurée des informations pertinentes des images de factures (Nom du commerce, Date, Montant total) en utilisant l'API Mistral/Pixtral avec

une requête en mode JSON.
    - Gestion des appels API (encodage base64, format Data URL, gestion basique du rate limiting).
    - Implémentation d'une logique de matching en deux étapes :
        1. **Filtrage primaire :** Sélection des factures candidates basées sur la correspondance du montant (avec tolérance) ET la proximité de la date (avec delta de jours, appliqué conditionnellement si les dates sont valides).
        2. **Classement secondaire :** Calcul de la similarité sémantique (via embeddings et similarité cosinus) entre la description de la transaction bancaire et le nom du commerce extrait de la facture, pour les candidats pré-filtrés. Sélection du meilleur match au-dessus d'un seuil.
    - Génération d'un DataFrame Pandas contenant les correspondances trouvées.

4. Données d'Entrée / Sortie

- **Entrée :**
    - **Relevé Bancaire :** Fichier CSV unique. Doit contenir au minimum des colonnes identifiables comme date, amount (montant numérique), et une colonne de description (vendor ou source).
    - **Factures :** Un ou plusieurs fichiers image (JPG, PNG, JPEG).
- **Sortie :**
    - **Affichage UI :** Tableau (st.dataframe) présentant les lignes de relevé matché avec les détails de la facture correspondante et le score de similarité. Colonnes typiques : Date Relevé, Montant Relevé, Description Relevé, Fichier Facture, Nom Commerce (Facture), Date (Facture), Montant (Facture), Score Similarité (%).
    - **Téléchargement :** Fichier CSV (resultats_matching_final.csv) contenant les mêmes informations que le tableau affiché.

5. Architecture / Pipeline de Traitement

1. **Initialisation :** Chargement des clés API (.env), initialisation et mise en cache du client Mistral AI et du modèle Sentence Transformer.
2. **Upload & Parsing :** L'utilisateur charge les fichiers via l'interface Streamlit.
    - Le CSV est lu, nettoyé, et les types sont convertis par utils/parser.py (v3).
    - Les fichiers images sont listés.
3. **Lancement du matching (Clic Bouton) :**
    - **Étape OCR (par facture) :**
        - Lecture des bytes de l'image.
        - Encodage en Base64.

        - Appel à call_pixtral_ocr_json : envoi à l'API Mistral/Pixtral (modèle pixtral-12b-2409) avec un prompt demandant un output JSON (date, nom_commerce, montant). Gestion du rate limiting incluse.
        - Parsing de la réponse JSON. Validation/Conversion des types (montant en float, date en datetime si possible).
        - Stockage des données extraites pour toutes les factures.
    - **Étape Matching (par transaction relevé) :**
        - Appel à filter_and_match_invoices :
            - Pré-filtrage des factures OCRisées basé sur la tolérance de montant et le delta de date conditionnel.
            - Si des candidats restent, calcul des embeddings du nom du commerce (factures) et de la description (relevé) via SentenceTransformer.
            - Calcul de la similarité cosinus.
            - Sélection de la meilleure facture si le score dépasse le seuil.
        - Stockage des correspondances trouvées.
4. **Affichage & Téléchargement :**
    - Création d'un DataFrame Pandas avec les correspondances.
    - Affichage du DataFrame dans Streamlit.
    - Préparation et mise à disposition du bouton de téléchargement CSV.

6. Choix Techniques Principaux

- **Langage :** Python (v3.13 ou compatible)
- **Interface Web :** Streamlit
- **Manipulation Données :** Pandas
- **OCR :** Mistral AI / Pixtral (modèle pixtral-12b-2409 ou mistral-large-latest) via la bibliothèque mistralai (v0.4.2 pour compatibilité). Extraction directe en JSON.
- **Matching Sémantique :** Bibliothèque sentence-transformers (modèle paraphrase-MiniLM-L6-v2) et scikit-learn (pour cosine_similarity).
- **Gestion Environnement :** python-dotenv (pour clés API), venv (pour isolation).
- **Traitement Image :** Pillow (pour lecture initiale).

7. Contraintes et Limites Connues

- **Dépendance API :** Nécessite une clé API Mistral valide et fonctionnelle. Soumis aux limitations de taux et à la disponibilité de l'API.
- **Qualité OCR :** La performance globale dépend fortement de la qualité de l'extraction JSON par Pixtral. Des erreurs d'OCR sur les montants, dates ou noms impacteront directement le matching.
- **Qualité Matching :**
    - Le filtre par montant est strict (tolérance faible).

    - Le filtre par date dépend de la capacité à parser les dates du CSV et de l'OCR. Les dates invalides désactivent ce filtre (conditionnellement).
    - La similarité sémantique dépend de la qualité de la description du relevé et du nom extrait par l'OCR, ainsi que de la pertinence du modèle d'embedding choisi.
    - Les seuils (tolérance montant, delta date, similarité) peuvent nécessiter des ajustements fins.
- **Formats CSV/Date :** L'application tente de gérer les formats de date courants mais peut échouer sur des formats exotiques (un avertissement est émis). La présence des colonnes amount, date, vendor/source est attendue.
- **Compatibilité mistralai:** Utilisation d'une version spécifique (0.4.2) pour assurer la compatibilité avec le code actuel, en attendant une migration vers la nouvelle interface client si nécessaire.

8. Planning Réalisé (Indicatif - Semaine du 31/03/2025)

- **Jour 1 (Lundi) :**
    - Compréhension du sujet, analyse des données fournies.
    - Mise en place de l'environnement de développement (Python 3.13, venv, Git).
    - Initialisation du projet Streamlit, création de la structure de base.
    - Début de rédaction du Cahier des Charges v1.
    - Implémentation de l'upload de fichiers (CSV, Images).
- **Jour 2 (Mardi) :**
    - Finalisation Cahier des Charges v1.
    - Développement et test du parsing CSV (utils/parser.py), incluant nettoyage des colonnes et gestion basique des types.
    - Recherche et premiers tests sur les technologies OCR (Pixtral via API Mistral, EasyOCR).
    - Début d'intégration d'une solution OCR (potentiellement EasyOCR ou simulation Pixtral).
- **Jour 3 (Mercredi) :**
    - Intégration de l'appel à l'API Pixtral avec gestion de l'encodage Base64 et du format Data URL.
    - Implémentation de l'extraction JSON via le prompt Pixtral.
    - Recherche et tests sur les méthodes de matching (Fuzzy, Embeddings/Similarité Cosinus).
    - Développement de la logique de matching sémantique avec Sentence Transformers.
- **Jour 4 (Jeudi - Deadline Initiale) :**
    - Intégration de la logique de matching en deux étapes (Filtre Montant/Date +

Similarité).
    - Implémentation du filtre de date conditionnel pour gérer les dates invalides.
    - Mise en place de l'affichage des résultats dans un tableau Streamlit.
    - Ajout de la fonctionnalité de téléchargement CSV.
    - Tests de la pipeline complète et débogage initial.
- **Jour 5+ (Post-Deadline) :**
    - Débogage avancé des problèmes de compatibilité des bibliothèques (mistralai).
    - Stabilisation de la version en fixant mistralai==0.4.2.
    - Optimisation du chargement des modèles via @st.cache_resource.
    - Amélioration de la gestion des erreurs et des messages utilisateur.
    - Finalisation de la documentation (dont ce Cahier des Charges v2).

Ce planning indicatif montre une progression logique intégrant la recherche, l'implémentation et le débogage pour aboutir à la version actuelle de l'application.