# **Global Methodology Document**

## **1. Introduction**

Ce document décrit la méthodologie adoptée pour la conception et l’implémentation du pipeline de lecture et d’analyse de documents financiers. Ce pipeline s’articule autour de plusieurs composants :

* **Parsers** pour extraire le texte brut depuis différents formats de documents (PDF, DOCX, TXT).
* **Modèle NER** pour l’extraction d’entités financières dans les textes structurés (exemple : chats) avec la possibilité de fine-tuning.
* **Module Rule-based** pour les documents DOCX afin d’extraire des paires clé/valeur via la détection de tableaux.
* **Pipeline RAG** pour les fichiers PDF, permettant de réaliser du Q&A, du résumé ou de l’extraction d’entités en s’appuyant sur un LLM via Langchain et le modèle Groq.

Ce Global Methodology Document expose les arguments derrière chacun de ces choix et détaille les techniques mises en œuvre pour obtenir un système robuste, scalable et adaptable aux différents besoins du SI.

## **2. Choix et Fine-tuning du Modèle NER**

Ce document décrit la méthodologie pour adapter (fine-tuner) un modèle de reconnaissance d'entités nommées (NER) afin d'extraire des entités financières spécifiques à partir de différents types de documents (par exemple: chats, emails..). Les entités ciblées dans ce cas sont :

* **Counterparty** : BANK ABC
* **Notional** : 200 mio
* **ISIN** : FR001400QV82
* **Underlying** : AVMAFC FLOAT 06/30/28
* **Maturity** : 2Y EVG
* **Bid** : estr+45bps
* **Offer** **PaymentFrequency** : Quarterly

#### **2.1. Choix du Modèle de Base**

Pour ce Proof of Concept, nous avons opté pour le modèle spaCy « en\_core\_web\_sm » en raison de :

* Sa facilité d'utilisation et d'intégration dans des applications Python.
* Sa rapidité d'exécution pour des tests exploratoires.
* La possibilité de le fine-tuner sur un corpus annoté pour des entités financières spécifiques.

#### **2.2. Téléchargement et Utilisation du Modèle**

Le code fourni montre comment télécharger et utiliser un modèle NER généraliste pour extraire des entités d'un fichier texte avec la librairie SPacy. Ce modèle reconnaît par défaut des catégories standard (ex. ORG, MONEY, DATE) mais n'est pas adapté aux entités financières spécifiques sans entraînement complémentaire.

#### **2.3. Méthodologie de Fine-Tuning**

Pour améliorer la précision de l'extraction et adapter le modèle aux entités financières ciblées, la démarche suivante est proposée :

**a) Collecte et Annotation du Corpus**

* **Collecte de données :** Rassembler un ensemble de documents financiers représentatifs (chats, emails, documents contractuels, etc.) contenant les entités d'intérêt.
* **Annotation :** Utiliser des outils pour annoter manuellement ou avec des expressions régulières les entités spécifiques (Counterparty, Notional, ISIN, Underlying, etc.).

**b) Préparation des Données**

* Convertir les annotations dans un format compatible avec spaCy (par exemple, en format JSON).
* Diviser le corpus en ensembles d'entraînement, de validation et de test pour évaluer la performance du modèle.

**c) Fine-Tuning du Modèle**

* Utiliser la pipeline d'entraînement de spaCy pour réentraîner le modèle NER sur le corpus annoté. (create\_pipe / resume\_training)
* Adapter et créer de nouveaux labels pour les entités financières avec add\_pipe (par exemple, « COUNTERPARTY », « NOTIONAL », « ISIN », etc.).
* Ajuster les hyperparamètres (nombre d'époques, taux d'apprentissage, etc.) pour optimiser la performance du modèle.

**d) Évaluation et Itération**

* Mesurer les performances du modèle sur l'ensemble de test à l'aide de métriques telles que le F1-score.
* Analyser les erreurs et enrichir le corpus avec des exemples supplémentaires si nécessaire.
* Envisager l'utilisation de modèles spécialisés dans le domaine financier (comme FinBERT ou d'autres modèles pré-entraînés) pour comparer les résultats et potentiellement les intégrer à la solution.

## **3. Méthodologie RAG et Stratégies de Prompting**

### **3.1. Objectifs du Pipeline RAG**

Les fichiers PDF sont souvent volumineux et non structurés, ce qui rend l’extraction d’information classique difficile. La méthodologie Retrieval-Augmented Generation (RAG) permet de combiner la récupération d’informations pertinentes et la génération de réponses contextuelles à l’aide d’un LLM.

### **3.2. Composants du Pipeline RAG**

* **Extraction initiale du texte** : Utilisation du parser PDF pour obtenir le texte brut.
* **Embedding et vectorisation des données :** Utiliser des modèles d’embedding type BERT (Camembert ou FinBert..) et vectoriser les données.
* **Base de données vectorielle :** Pour des données de petites et moyennes tailles, nous pouvons créer une base de données vectorielle et la répertorier dans ChromaDB ou Faiss
* **Récupération de contexte** : À partir du texte vectorisé, extraire ou sélectionner les passages les plus pertinents (par exemple, par une segmentation ou une recherche de mots-clés). Avec le retrieval du RAG basé sur du cosinus similarité. On peut mettre en place des prompts systems qui vont réduire la data aux entités recherchées avant d’interroger le LLM pour utiliser moins de ressources.
* **Interrogation du LLM via Langchain** : Utilisation de Langchain pour orchestrer la génération de réponses ou de résumés. ( Le modèle utilisé dans notre exemple est llama3-8b-8192, accessible via une clé API de Groq dédiée.)
* **Prompting adapté** : En fonction de l'opération (Q&A, résumé ou extraction d’entités), des prompts spécifiques sont conçus pour guider le LLM. Par exemple :
  + Pour du Q&A : « En vous basant sur le texte suivant, répondez à la question : ... »
  + Pour du résumé : « Fournissez un résumé concis du texte suivant : ... »
  + Pour de l'extraction d’entités : « Extraites les entités suivantes au format JSON : ... »

### **3.3. Stratégies de Prompting**

* **Clarté et Contexte** : Les prompts doivent fournir suffisamment de contexte au LLM pour générer des réponses pertinentes. La formulation doit être claire, directe et contenir des instructions précises. **Pour affiner les réponses et obtenir les résultats souhaités, des structures de prompting comme CIDI (context instructions details input) peuvent être utilisées.**
* **Itération et Ajustement** : La qualité des résultats dépend grandement du prompt. Des tests itératifs permettent d’affiner les formulations pour maximiser la performance.
* **Combinaison de Récupération et Génération** : Le pipeline RAG intègre à la fois une phase de récupération (pour extraire les parties pertinentes du document) et une phase de génération (pour formuler la réponse ou le résumé). Cette approche hybride permet d’améliorer la précision et la robustesse du système.
* **Structure du Prompt :** 
  + Contexte détaillé : Inclure les passages récupérés par le retrieval.
  + Instructions explicites : Préciser les entités à extraire et fournir leur description (par exemple, "Counterparty : nom de la banque ou de l’institution financière", "Notional : montant nominal en devise", etc.).
  + Possibilité d’intégrer des instructions de l’utilisateur.

### **3.4. Intégration des Descriptions et Entités**

### **Ajouter les définitions :** Chaque entité est accompagnée d’une description qui aide le LLM à comprendre son rôle et sa structure.

### **Format de sortie structuré :** Demander une réponse en JSON pour faciliter l’intégration dans le système.

### **3.6. Amélioration proposée**

#### **Principe du Few-Shot Learning**

* **Fournir des exemples :** Intégrer quelques exemples annotés dans le prompt afin d’illustrer le format attendu et le type d’entités à extraire.
* **Réduction de l’ambiguïté :** Ces exemples guident le modèle dans la tâche et réduisent les erreurs d’interprétation.

**Exemples dans le prompt :** Ajouter 2 à 3 exemples au sein du prompt, par exemple:

Exemple 1:

Texte : "À 11:49:05, BANK ABC a confirmé un Notional de 200 mio avec l'ISIN FR001400QV82..."

Extraction attendue : {

"Counterparty": "BANK ABC",

"Notional": "200 mio",

"ISIN": "FR001400QV82",

"Underlying": "AVMAFC FLOAT 06/30/28",

"Maturity": "2Y EVG",

"Bid": "estr+45bps",

"Offer": null,

"PaymentFrequency": "Quarterly"

}

Exemple 2:

Texte : "..."

Extraction attendue : { ... }

1. **Pydantic**

Pydantic peut considérablement améliorer notre pipeline d'extraction d'entités en garantissant la validation et la cohérence des données. Par exemple, en définissant des modèles de données avec des annotations de type strictes pour représenter les entités extraites (exemple "Counterparty", "Notional", "ISIN", etc.), Pydantic assure que les données respectent le format attendu.