T-ESP-700

Big Brain

Lucas FIXARI
Pierre ROCHETTE
William WOZIWODA

Description Fonctionnelle et Technique - IA d'Analyse Documentaire

1. Introduction

Ce document présente la description fonctionnelle et technique du projet d'IA d'Analyse Documentaire Locale. Il détaille les modules principaux de l'application, les technologies utilisées (exclusivement en Python) et les critères de performance pour chaque composant.

2. Architecture Générale

L'architecture du projet repose sur une infrastructure modulaire en Python permettant une analyse, une indexation et une recherche de documents en local. Les modules sont :

1. **Prétraitement des Données**
2. **Traitement Automatique du Langage Naturel (NLP)**
3. **Domaine de Connaissances (Optionnel)**
4. **Extraction et Intégration Sémantique**
5. **Base de Connaissances**
6. **Interface Utilisateur**

3. Modules Fonctionnels

3.1 Prétraitement des Données

Objectif : Préparer les documents en extrayant et en structurant le contenu pour un traitement NLP et une indexation efficace.

Technologies utilisées :

- **Extraction de texte et d'images** : *PyMuPDF* pour PDF, *python-docx* pour DOCX, *openpyxl* pour XLSX.
- **Reconnaissance optique de caractères (OCR)** : *pytesseract* pour extraire du texte à partir d'images scannées (JPG, PNG).
- **Tokenisation et Normalisation** : *spaCy* ou *nltk* pour la tokenisation et la lemmatisation ; *nltk.corpus.stopwords* pour filtrer les mots communs.

Description de Fonctionnalité :

- Extraction des contenus textuels et visuels des fichiers PDF, DOCX, XLSX, JPG et PNG.
- Transformation du contenu extrait en un format JSON pour simplifier l'indexation.
- Nettoyage des données, tokenisation, et normalisation pour garantir une cohérence textuelle.

Critères de Réussite :

- Extraction de texte avec une précision de 98% pour des documents standards.
- Temps de traitement par document de 5 secondes maximum.

Plan de Contournement : En cas de limitations avec certains formats, privilégier les formats PDF et DOCX.

3.2 Traitement Automatique du Langage Naturel (NLP)

Objectif : Interpréter les documents et les requêtes utilisateur en identifiant les entités et les intentions pour une recherche contextuelle.

Technologies utilisées :

- **Reconnaissance d'entités et étiquetage** : *spaCy* pour l'extraction d'entités, *Transformers* (via Hugging Face) pour des modèles avancés comme BERT ou GPT.
- **Analyse syntaxique** : *spaCy* pour l'analyse syntaxique et la détection des relations grammaticales.

Description de Fonctionnalité :

- Identification des entités et étiquetage des parties du discours pour chaque document.
- Analyse syntaxique pour comprendre les relations contextuelles dans le texte.

Critères de Réussite :

- Précision de 80% dans la reconnaissance des entités et la compréhension syntaxique.
- Temps de réponse inférieur à 3 secondes pour les questions simples.

Plan de Contournement : Simplifier les questions initiales en cas de limitation des modèles NLP.

3.3 Domaine de Connaissances (Optionnel)

Objectif : Enrichir le traitement avec une ontologie et des exemples pour une analyse plus approfondie.

Technologies utilisées :

- **Ontologie et Éditeur d'ontologie** : *rdflib* pour créer et gérer les ontologies en RDF.
- **Exemples supervisés** : *scikit-learn* ou *Transformers* pour l'apprentissage supervisé, si nécessaire.

Description de Fonctionnalité :

- Création et gestion d'une ontologie pour structurer les relations entre concepts.
- Utilisation d'exemples spécifiques pour améliorer l'interprétation contextuelle.

Critères de Réussite :

- Ontologie correctement définie pour structurer les connaissances du domaine.

- Exemples pertinents pour ajuster le modèle si besoin.

3.4 Extraction et Intégration Sémantique

Objectif : Extraire les relations sémantiques entre entités et structurer les informations dans la base de connaissances.

Technologies utilisées :

- **Extraction des relations sémantiques** : *spaCy* pour les relations simples, ou *Transformers* pour une extraction avancée.
- **Intégration des connaissances** : *networkx* pour structurer les connaissances sous forme de graphe.

Description de Fonctionnalité :

- Extraction des relations sémantiques entre les entités détectées pour contextualiser les réponses.
- Intégration des informations extraites dans une structure de base de connaissances.

Critères de Réussite :

- Temps de réponse pour la détection des relations inférieur à 5 secondes.
- Capacité à traiter au moins 1 000 documents sans perte de performance.

Plan de Contournement : Simplifier l'intégration en cas de limitation des performances.

3.5 Base de Connaissances

Objectif : Stocker et structurer les données extraites pour permettre des recherches rapides et efficaces.

Technologies utilisées :

- **Stockage structuré** : *SQLite* pour le stockage local, *Whoosh* pour la création d'un moteur de recherche textuel.

Description de Fonctionnalité :

- Indexation des documents avec métadonnées et stockage structuré des informations extraites.
- Gestion des requêtes pour des recherches rapides et optimisées.

Critères de Réussite :

- Temps de réponse pour les recherches inférieur à 5 secondes.
- Gestion de 1 000 documents sans perte de performance.

Plan de Contournement : Adapter SQLite pour des recherches simples en cas de limitation de Whoosh.

3.6 Interface Utilisateur

Objectif : Permettre à l'utilisateur de poser des questions en langage naturel et d'afficher les réponses de manière intuitive.

Technologies utilisées :

- **Interface web ou GUI** : *Flask*, *FastAPI*, ou *tkinter* pour créer l'interface utilisateur.
- **Synthèse et Résumé** : *Sumy* pour des résumés simples, ou *Transformers* (T5 ou BART) pour des résumés avancés.

Description de Fonctionnalité :

- Interface permettant aux utilisateurs de poser des questions et de recevoir des réponses sous forme de résumés.
- Option de visualisation des documents sources avec navigation par thème ou type de document.

Critères de Réussite :

- Affichage des résultats en moins de 5 secondes.
- Résumés précis et concis des documents pertinents.

Plan de Contournement : Afficher des extraits directs en cas de limitation des résumés automatiques.

4. Synthèse des Technologies Utilisées

Module	**Tâche**	**Technologie**
Prétraitement des Données	Nettoyage et normalisation	*re*, *unicodedata*
	Extraction de texte (PDF, DOCX, XLSX)	*PyMuPDF, python-docx, openpyxl*
	OCR pour images	*pytesseract*
Traitement NLP	Reconnaissance d'entités	*spaCy*, *Transformers*
	Étiquetage POS	*spaCy*
Domaine de Connaissances	Éditeur d'ontologie	rdflib
	Exemples supervisés	*scikit-learn*, *Transformers*
Extraction Sémantique	Détection de relations	*spaCy*, *Transformers*

Base de Connaissances	Stockage et gestion des connaissances	*SQLite*, *Whoosh*
Interface Utilisateur	Recherche et visualisation	*Flask*, *FastAPI*, *tkinter*

5. Critères de Qualité et Tests

Tests de Qualité

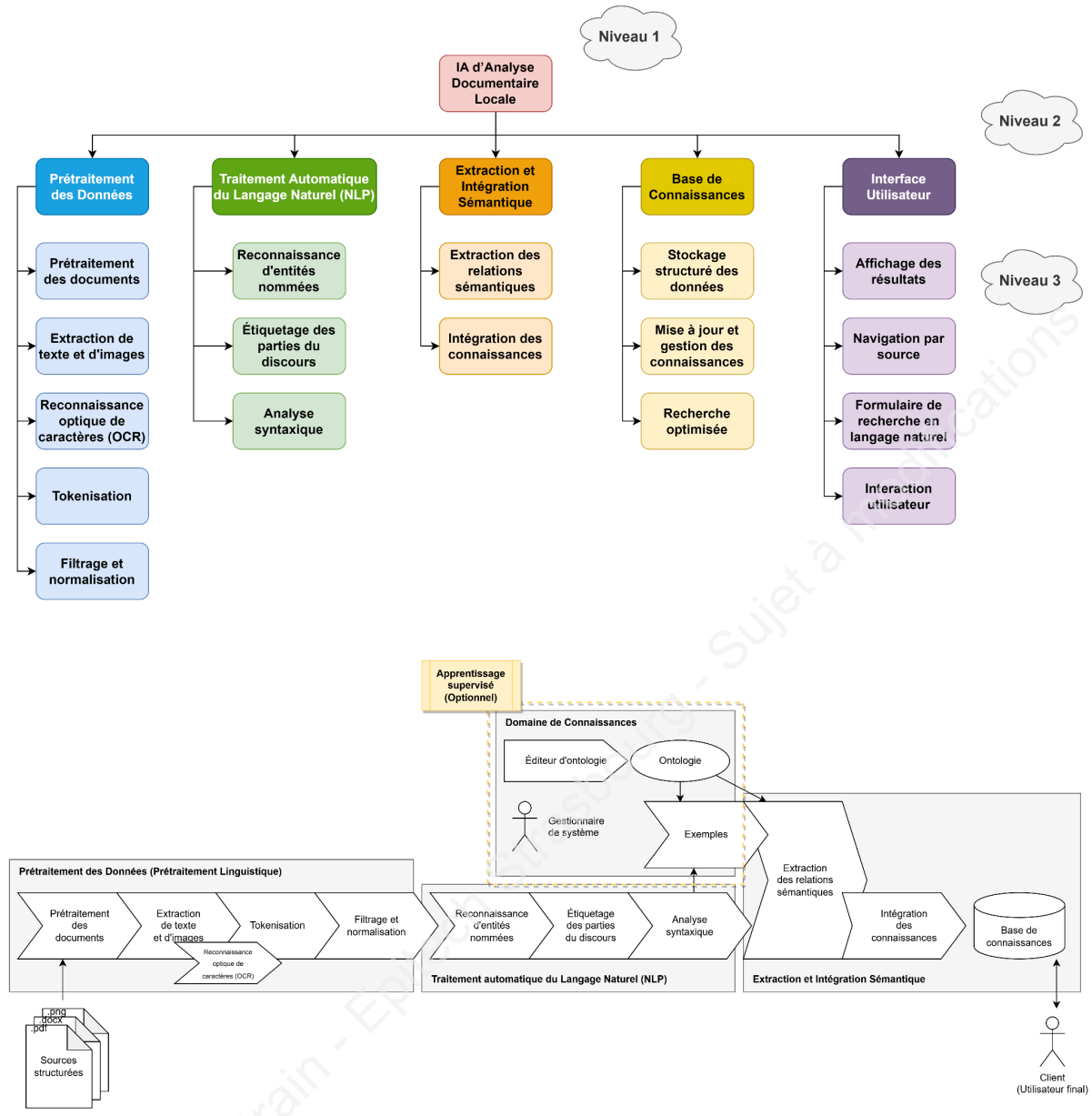
1. **Tests de Prétraitement** : Vérifier la précision des extractions pour les différents formats de documents (PDF, DOCX, images).
2. **Tests NLP** : Évaluer la précision du NLP dans l'interprétation des questions.
3. **Tests de Synthèse et Restitution** : Vérifier que les résumés sont clairs, pertinents et accompagnés des sources documentaires.
4. **Tests d'Indexation et Recherche** : Vérifier le temps de réponse des recherches et la capacité à traiter de grands volumes de documents.

Indicateurs Clés de Performance (KPIs)

- **Précision d'Extraction** : Au moins 98% pour les formats standard.
- **Temps de Réponse des Recherches** : Inférieur à 5 secondes pour les recherches de base.
- **Précision du NLP** : 80% de précision pour les questions simples.
- **Temps de Résumé** : Temps de réponse de 3 secondes maximum pour la synthèse de documents.

6. Diagramme de Flux des Modules

Un diagramme simplifié de l'interaction entre les modules principaux :



Ce flux garantit que chaque étape du traitement des documents se réalise en séquence, permettant une recherche efficace et une restitution contextuelle.

7. Documentation et Support Technique

Documentation Technique

- **Guide d'installation** : Instructions pour installer les dépendances Python (ex. pytesseract, spaCy, Flask).
- **Guide API** : Documentation de l'API pour chaque module (analyse, indexation, NLP, restitution).
- **Guide d'utilisation** : Explication de l'interface et des fonctionnalités principales pour les utilisateurs.

Support et Maintenance

- **Suivi des erreurs** : Utilisation de *loguru* pour capturer les erreurs et analyser les performances.
- **Mises à jour** : Prévoir un plan de mise à jour pour ajouter des fonctionnalités ou améliorer la précision NLP.