L’**analyse automatique et le parsing des CV** (ou "CV parsing" en anglais) désignent un processus qui permet d’extraire automatiquement, à partir d’un fichier de CV (PDF, Word, etc.), les **informations structurées** (nom, prénom, formation, expérience professionnelle, compétences, etc.). Cela permet notamment d’intégrer rapidement ces données dans un **système de gestion des candidatures** (ATS – Applicant Tracking System), une **base de données RH**, ou tout autre outil d’analyse.

**🔍 Comment ça fonctionne ?**

1. **Lecture du document** : le système lit le fichier (PDF, DOCX, etc.).
2. **Extraction des textes** : il récupère les textes exploitables.
3. **Détection des blocs d’informations** : il identifie les zones comme "expérience", "formation", "langues", etc.
4. **Interprétation sémantique** : via des règles, des expressions régulières ou de l’IA/NLP (traitement du langage naturel), il comprend le contenu.
5. **Structuration des données** : il organise ces informations dans un format exploitable (JSON, tableau Excel, base de données…).

**🛠️ Peut-on le développer en interne ?**

Oui, **c’est faisable en interne**, mais avec des **conditions** :

**✅ Ce qui est possible avec un bon développeur :**

* Extraire le texte brut de fichiers (via des bibliothèques comme pdfminer, PyMuPDF, docx, etc.).
* Détecter des sections avec des règles simples.
* Parser les dates, les diplômes, les intitulés de poste avec des expressions régulières.
* Stocker les données dans une base ou tableau.

**⚠️ Mais attention à la complexité :**

* Les CV sont **très hétérogènes** (pas de standard).
* Certaines informations sont implicites, mal structurées ou à interprétation variable (ex : "Consultant junior chez X" → il faut comprendre le poste et l’entreprise).
* Pour un parsing fiable, il faut souvent des **modèles NLP avancés** (spaCy, Transformers, BERT, etc.).
* Il faut aussi penser à la **gestion des langues**, des abréviations, des formats de dates, etc.

**🧠 Alternatives à envisager**

1. **Développement interne "léger"** :
   * Pour des cas simples (CVs internes, formulaires standards).
   * Pour une première version de test ou un POC.
2. **Utilisation de bibliothèques open source / API** :
   * **Libraries** : spaCy, Resume Parser, PyResparser.
   * **APIs tierces** : Sovren, RChilli, Affinda, HireAbility (souvent payantes, mais très précises).
3. **Acheter un outil déjà prêt (ATS avec parsing intégré)** :
   * Exemples : Talensoft, SmartRecruiters, Softy, Flatchr, etc.
   * Gain de temps, mais moins personnalisable.

**💡 Recommandation**

* Si vous avez des **ressources IT disponibles** et un **volume modéré de CVs**, un développement en interne pour un MVP est tout à fait envisageable.
* Si vous traitez des **centaines de CVs par semaine** ou avez besoin d’une grande précision multilingue → mieux vaut intégrer une solution existante ou utiliser une API spécialisée.

**Outils utilisés**

* pdfplumber : pour extraire du texte brut depuis un PDF.
* re : pour utiliser des expressions régulières (détection des mails, numéros, etc.).
* **Objectif** : détecter nom, email, téléphone, expériences et formations (de manière basique).

**🧪 Exemple de script Python simplifié**

import pdfplumber

import re

def extract\_text\_from\_pdf(file\_path):

with pdfplumber.open(file\_path) as pdf:

text = ""

for page in pdf.pages:

text += page.extract\_text() + "\n"

return text

def parse\_cv(text):

result = {}

# Email

email\_match = re.search(r'[\w\.-]+@[\w\.-]+', text)

result["email"] = email\_match.group(0) if email\_match else "Non trouvé"

# Téléphone

phone\_match = re.search(r'(?:(?:\+?\d{1,3})?[\s-]?)?\(?\d{2,4}\)?[\s-]?\d{2,4}[\s-]?\d{2,4}', text)

result["téléphone"] = phone\_match.group(0) if phone\_match else "Non trouvé"

# Nom (très simplifié – à améliorer avec NLP ou formalisme interne)

lines = text.strip().split("\n")

if lines:

result["nom"] = lines[0].strip() # hypothèse : le nom est en haut

# Expériences pro

experience\_keywords = ["expérience", "expériences", "professionnelle", "travail", "poste", "mission"]

experience\_text = "\n".join([line for line in lines if any(kw in line.lower() for kw in experience\_keywords)])

result["expériences"] = experience\_text if experience\_text else "Non détecté"

# Formations

formation\_keywords = ["formation", "diplôme", "université", "école", "baccalauréat", "licence", "master"]

formation\_text = "\n".join([line for line in lines if any(kw in line.lower() for kw in formation\_keywords)])

result["formations"] = formation\_text if formation\_text else "Non détecté"

return result

# Exemple d'utilisation

if \_\_name\_\_ == "\_\_main\_\_":

cv\_text = extract\_text\_from\_pdf("cv\_exemple.pdf") # Remplacer par le chemin réel

parsed\_data = parse\_cv(cv\_text)

for key, value in parsed\_data.items():

print(f"{key.upper()}:\n{value}\n{'-'\*40}")

🔧 Résultat attendu (exemple)

EMAIL:

herinome.sedera@example.com

----------------------------------------

TÉLÉPHONE:

+261 34 12 345 67

----------------------------------------

NOM:

RASOLOFONDRAMANAMBE Herinomenjanaharilala Sedera

----------------------------------------

EXPÉRIENCES:

Expérience professionnelle

2019-2020 Consultant RH chez ACEP Madagascar

2017-2019 Responsable RH chez OMNIVEST

----------------------------------------

FORMATIONS:

Master en Gestion des Ressources Humaines - Université d'Antananarivo

Licence en Droit Privé

----------------------------------------

**🧱 Etapes suivantes si vous voulez aller plus loin :**

* Ajouter un moteur NLP avec **spaCy** pour détecter des entités (poste, entreprise, date, etc.).
* Stocker dans une **base SQLite / Excel / API**.
* Gérer différents formats (DOCX, TXT, etc.).
* Créer une interface utilisateur web (via Flask ou Streamlit).

**Option 1 — Prototype simple (script console modifiable)**

* Extraction depuis un CV PDF
* Parsing de base : nom, email, téléphone, expériences, formations
* Résultats affichés dans le terminal ou enregistrés dans un fichier Excel ou CSV
* ➕ Facile à tester localement

**💻 Option 2 — Mini-application web (interface utilisateur simple)**

* Interface en **Streamlit** (interface rapide en Python)
* Téléversement du CV en PDF
* Affichage des résultats dans un tableau
* Option de téléchargement des données extraites
* ➕ Idéal pour présenter à une direction ou des collègues non-tech

**📦 Fichier inclus dans le prototype**

* main.py (code source)
* requirements.txt (bibliothèques à installer)
* Exemple de CV PDF test
* Fichier Excel/CSV exporté (optionnel)