PROYECTO TERMINAL - Evaluación de candidatos de trabajo

Miguel Angel Tovar Rodríguez
Carlos Manuel Vélez
Andrés Martínez Cabrera

Este documento describe la arquitectura del sistema para evaluar la aptitud de los aspirantes a un puesto de trabajo, basándose en sus CVs y la descripción de las ofertas de empleo. El sistema tiene como objetivo proporcionar una herramienta automatizada que extrae, procesa y clasifica la información de los aspirantes para determinar la relevancia de su perfil frente a una oferta laboral.

1. Objetivos del Sistema

El sistema tiene como objetivo:

1. **Extraer información clave** de los CVs de los aspirantes, las ofertas de trabajo en formato HTML, y los sus relaciones en un documento de aplicaciones en archivo `.parquet`. (Brindados por Pisa Farmaceutica)

2. **Preprocesar y transformar** la información extraída en un formato adecuado para su análisis.

3. **Clasificar la aptitud de los aspirantes** mediante un modelo de aprendizaje supervisado basado en las similitudes entre las ofertas de trabajo y los CVs de los aspirantes.

4. **Almacenar los resultados** de la evaluación en un archivo JSON para facilitar la revisión y toma de decisiones.

---

2. Arquitectura General

La arquitectura del sistema se basa en un flujo modular de procesamiento, en el que se destacan tres componentes principales:

1. **Extracción de Datos**.

2. **Preprocesamiento de Datos**.

3. **Clasificación y Evaluación**.

4. **Almacenamiento de Resultados**.

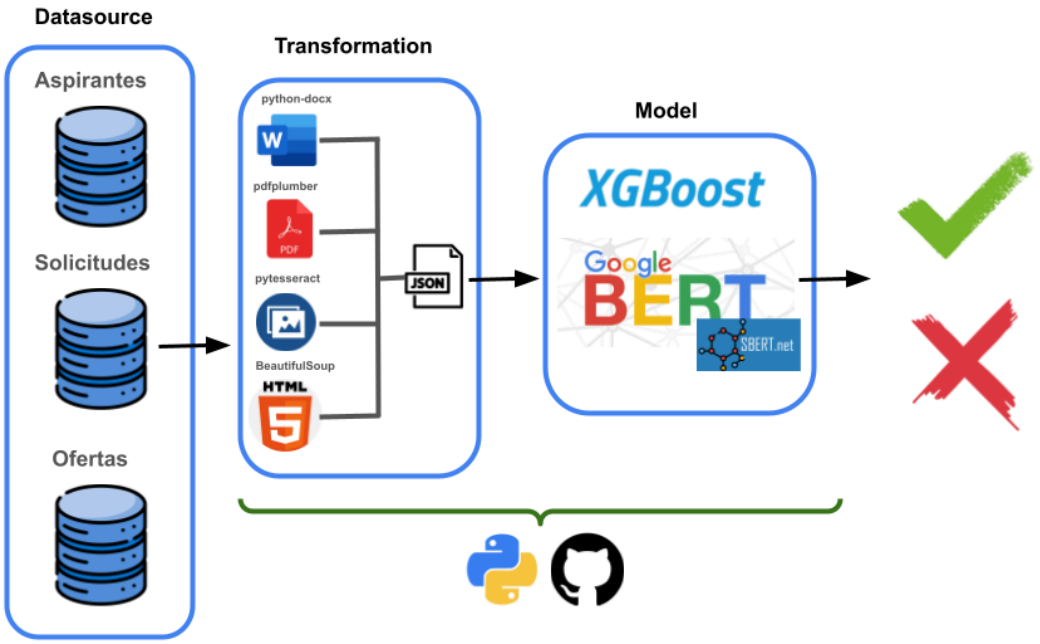
Flujo de Datos

1. **Extracción de Datos**:

    - **Ofertas de Trabajo (HTML)**: Se extrae el texto relevante de cada archivo HTML que describe la oferta.

    - **CVs de Aspirantes (PDF)**: Se extrae el texto de cada archivo PDF correspondiente a un aspirante.

    - **Información de Aspirantes (Parquet)**: Se lee el archivo `.parquet` que contiene datos adicionales de los aspirantes.



2. **Preprocesamiento**:

    - El texto extraído es limpiado, tokenizado y preparado para ser alimentado al modelo de clasificación.

3. **Clasificación y Evaluación**:

    - **Modelo de Clasificación**: Se utiliza un modelo de **XGBoost** o **BERT** para comparar la similitud entre las ofertas de trabajo y los CVs, y clasificar la aptitud de los aspirantes.

4. **Almacenamiento de Resultados**:
    - Los resultados de la clasificación se guardan en un archivo JSON que incluye el CV evaluado, la oferta correspondiente, la similitud de coseno y la clase predicha.

---

3. Variables y targets

CV:

- Datos personales*
- Skills (lista de soft/hard skills)
- Experiencia laboral (número de años, empresas).
- Educación y certificaciones

Puesto:

- Modalidad (remoto, presencial)*
- Seniority (Intern, Jr, Senior, etc)
- Match de perfil (carrera a la que se dirige la vacante)
- Skills requeridas

---

4. Componentes Técnicos

Lenguaje de Programación: Python

Librerías Principales:

1. **`BeautifulSoup`**: Para extraer datos de archivos HTML (ofertas de trabajo).
2. **`PyMuPDF`** (fitz) o **`pdfminer.six`**: Para extraer texto de archivos PDF (CVs).
3. **`pandas`**: Para trabajar con datos en formato `.parquet`.
4. **`sentence-transformers`**: Para calcular la similitud semántica entre textos utilizando modelos de BERT.
5. **`xgboost`**: Para crear un modelo de clasificación utilizando aprendizaje supervisado.
6. **`nltk`**: Para tokenizar y procesar el texto.

---

5. Flujo de Datos y Procesamiento

Extracción de Información

1. **Extracción de Datos de HTML (Ofertas)**:
    - Se carga cada archivo HTML correspondiente a una oferta de trabajo.
    - **`BeautifulSoup`** se utiliza para extraer el texto de la oferta, eliminando las etiquetas HTML y obteniendo el contenido textual.
2. **Extracción de Datos de PDF (CVs)**:
    - Cada archivo PDF se procesa utilizando **`PdfPlumber`**.
    - El texto extraído se almacena en formato limpio para su posterior análisis.
3. **Extracción de Datos de DOCX (CVs)**:
    - Para los archivos `.docx` (Microsoft Word), se utiliza la librería **`python-docx`** para extraer el contenido textual del documento.
    - Se procesan los archivos `.docx` de forma similar a los PDFs, pero en este caso extraemos texto estructurado desde las celdas de tablas o párrafos del archivo.
4. **Extracción de Datos de Imágenes (CVs escaneados)**:
    - Si los CVs están en formato de imagen (como .jpg o .png), se debe aplicar un proceso de **Reconocimiento Óptico de Caracteres (OCR)** para extraer el texto.
    - Se utilizará la librería **`pytesseract`** para extraer texto de imágenes escaneadas.
5. **Lectura de Información de Aspirantes (Parquet)**:
    - El archivo `.parquet` se carga usando **`pandas`**.
    - Se obtiene información adicional sobre los aspirantes, como experiencia, educación, y habilidades.

Preprocesamiento del Texto (Ingeniería de Características)

- **Limpieza**: Se eliminan caracteres especiales y se convierten los textos a minúsculas.
- **Tokenización**: Se divide el texto en palabras o tokens, eliminando stop words y realizando lematización.
- **Vectorización**: Se convierten los textos en vectores numéricos mediante **TF-IDF** o **embeddings**.

Clasificación y Evaluación

- **Modelo XGBoost**: Si se utiliza **XGBoost**, se entrenan características como la similitud entre el CV y la oferta de trabajo, junto con otras características extraídas de los textos.

- **Modelo BERT**: Si se utiliza **BERT** o **Sentence Transformers**, se generan embeddings para el CV y la oferta de trabajo, y se calcula la similitud semántica entre ambos. Se pueden usar modelos preentrenados como `paraphrase-MiniLM-L6-v2`.

Almacenamiento de Resultados

- Los resultados se guardan en un archivo JSON que contiene los detalles de la evaluación: similitud de coseno, clasificación (apto/no apto), y cualquier otra información relevante.

---

6. Modelos de Clasificación Propuestos

Modelo 1: XGBoost

XGBoost es un modelo de aprendizaje supervisado de **boosting** utilizado para clasificación. Este modelo se puede entrenar utilizando características derivadas del CV y de la oferta de trabajo, como similitudes de coseno y habilidades comunes.

Ventajas:

- Buena precisión para problemas tabulares con múltiples características.

- Se puede personalizar con diferentes hiperparámetros.

Modelo 2: BERT (Transformers)

BERT es un modelo de lenguaje preentrenado basado en transformers que se puede utilizar para tareas de clasificación de texto. Para este sistema, utilizaremos **`sentence-transformers`** para calcular los embeddings de los textos y medir la similitud semántica entre las ofertas de trabajo y los CVs.

Ventajas:

- Permite capturar relaciones semánticas profundas entre los textos.

- Ideal para problemas de texto, especialmente cuando los datos tienen una estructura más compleja.

---

7. Integración de Datos

1. **Extracción y Preprocesamiento**: Los datos de cada archivo (HTML, PDF, Parquet) se extraen y procesan de manera independiente, para luego ser unidos en un único formato estándar (tokens o embeddings).

2. **Modelo de Evaluación**: El modelo de clasificación utiliza los datos procesados (ya sea la similitud de coseno o las características derivadas del texto) para predecir la aptitud de los aspirantes para el puesto.

---

8. Entorno de Desarrollo

1. **Control de Versiones**: GitHub será el sistema de control de versiones para gestionar el código fuente, asegurando la colaboración eficiente entre los miembros del equipo y el seguimiento de cambios.

2. **Dependencias**:

    - Python 3.x

    - Librerías: `BeautifulSoup`, `PyMuPDF`, `pdfminer.six`, `pandas`, `nltk`, `xgboost`, `sentence-transformers`, `scikit-learn`.

3. **Entrenamiento y Evaluación de Modelos**:

    - Se realizará un entrenamiento inicial utilizando los datos etiquetados para crear el modelo de clasificación (XGBoost o BERT).

    - Se implementará validación cruzada y pruebas de rendimiento para ajustar el modelo.

---

9. Recomendaciones para Implementación

- **Optimización de Modelos**: Si el número de ofertas y CVs es elevado, puede ser necesario optimizar el modelo y el procesamiento de datos (ej. reducción de dimensionalidad, ajuste de hiperparámetros, paralelización).

- **Manejo de Errores**: Se debe implementar un sistema de manejo de excepciones robusto para manejar posibles errores en la extracción de datos (archivos corruptos, formatos inesperados).

- **Seguridad**: Asegurar que la información sensible (por ejemplo, datos personales de los aspirantes) se maneje adecuadamente y se mantenga protegida.

---

10. Conclusión

Esta arquitectura proporciona un enfoque modular y flexible para la extracción, procesamiento y clasificación de los CVs de los aspirantes y las ofertas de trabajo. Con el uso de técnicas avanzadas de procesamiento de lenguaje natural y modelos de aprendizaje supervisado, el sistema puede ofrecer evaluaciones precisas y relevantes para cada puesto, mejorando la eficiencia y reduciendo el sesgo humano en la toma de decisiones de contratación.