PROYECTO TERMINAL DE CIENCIA DE DATOS

‘

Selección de los mejores candidatos postulados a ofertas laborales mediante IA

Josue Gonzalo Galvan Ramos
Eduardo Sebastián González Ramírez
José Iván Andrade Rojas

CONTENIDO

Introducción.....4
Requerimientos.....6
Método.....7
Extracción de características.....8
Machine Learning Model.....9
Scoring Mechanism.....9
Pipeline & Workflow.....10
Implementation.....10
Diagrama de Arquitectura.....11
Recopilación de resultados.....12

Introducción

Reclutar a los candidatos adecuados para un puesto de trabajo es una tarea compleja, que requiere comprender tanto las cualificaciones del solicitante como los requisitos del puesto. Tradicionalmente, los reclutadores evalúan manualmente los currículos en función de las descripciones de los puestos, lo que lleva **mucho tiempo** y es subjetivo al reclutador.

El objetivo de este proyecto es **automatizar** y mejorar el proceso de emparejamiento de candidatos y puestos de trabajo mediante aprendizaje automático y el procesamiento del lenguaje natural . El sistema analizará informacion de los candidatos y descripciones estructuradas de puestos para determinar la idoneidad de un candidato para un determinado puesto.

Nuestro proposito es crear un sistema de **recomendación inteligente** que puntúe a los candidatos en función de su adecuación a la descripción del puesto, teniendo en cuenta sus aptitudes, experiencia y otros factores relevantes para finalmente mostrarlo de forma sencilla y facil de entender a los reclutadores.

Ingesta y Preparación de Datos

Exploracion

En la fase de exploración tenemos que cumplir los objetivos:

- Estructura de datos, encontrar como los datos del dataset estan estructurados.
- Encontrar el tipo de informacion disponible, Años de experiencia, Aptitudes, etc.

Preprocesamiento

Para el preprocesamiento de nuestros datos tenemos que cumplir :

- Limpieza, limpiar datos nulos e inconsistencias.
- Normalización de texto, lowercasing, stopwords, lematización.
- Feature engineering: extracción de habilidades, nivel de experiencia, etc.
- Codificación de variables categóricas (one-hot, embeddings, etc.)

Indexación estructurada en base de datos

Finalmente para la última parte nuestra preparación de datos tenemos que hacer que los datos sean legibles para los modelos de Machine Learning.

- Tablas separadas para: candidatos, trabajos, habilidades, evaluaciones, segun el dataset lo requiera.

Modelado y Entrenamiento

Nuestro objetivo aqui es construir un modelo supervisado capaz de emparejar eficientemente candidatos con diferentes tipos de trabajos.

¿Por qué usar un modelo supervisado?

- Tenemos datos históricos donde sabemos si un candidato fue *contratado*, *entrevistado* o recibió una evaluación de match con un trabajo.

- Con eso, podemos entrenar un modelo para aprender las relaciones entre:

    - Las características del candidato (skills, experiencia, educación, etc.)

    - Las características del trabajo

    - Y el resultado: si fue seleccionado o no, o qué tan buen match fue.

Esto es exactamente lo que hacen los modelos supervisados: aprenden una función **f(candidato, trabajo) -> resultado.**

¿Que modelo de machine learning usar?

Para saber que modelo de machine learning lo mas sensato seria probar los distintos modelos existentes con un dataset reducido, sin embargo eso puede tomar cierto trabajo así que decidimos investigar las pros y contras de cada modelo y compararlos con respecto al objetivo que tenemos actualmente.

Comparativa entre modelos supervisados

Modelo	Pros	Contras	Relevancia para el proyecto
Random Forest	- Robusto - Funciona bien con features mixtos (categóricos, numéricos) - Fácil de interpretar (feature importance)	- Lento con datasets muy grandes - Ocupa más memoria - No tan bueno para ranking fino	Bueno si nuestro dataset es de tamaño mediano y queremos interpretabilidad
XGBoost	- Preciso y rápido - Funciona muy bien con datos estructurados - Tiene métodos de ranking - Maneja datos faltantes mejor	- Más complejo de configurar - Algo más difícil de interpretar - Puede tener sobreajuste si no se regula	Excelente si queremos precisión y tenemos datos medianamente grandes
Logistic Regression	- Muy rápido - Fácil de interpretar - Ideal para línea base	- Lineal: no captura relaciones complejas - No tan bueno con muchos features categóricos o interacciones	Bueno como modelo base o si tenemos pocos datos
SVM	- Preciso en problemas bien separados - Útil con features no lineales si usas kernel	- No escala bien con datos grandes - Difícil de interpretar - Costoso computacionalmente	Solo útil si tenemos pocos datos y queremos probar separación clara

En resumidas cuentas **XGBoost** parece ser el mejor de los modelos para resolver casos como el nuestro y dada su naturaleza especializada sera el modelo con el que trabajaremos nuestra informacion.

Clasificación y Escoring

Para la clasificación y escoring tenemos que determinar cómo presentar los resultados del modelo de forma interpretable y útil, para esto tenemos que lograr:

- Normalizar el score del modelo entre 0 y 100 o usar percentiles.

Para la justificación del score (Explainability) usaremos técnicas de interpretabilidad: SHAP, LIME para mostrar **razones** por las que un candidato fue **recomendado**: experiencia, match de habilidades, etc..

Interfaz Gráfica / Visualización

El **objetivo** de nuestro proyecto es hacer que los reclutadores tengan una forma visible y rapida de saber cuales son los mejores candidatos, en pocas palabras hacer que nuestro modelo tenga un uso real dentro del mercado.

Para lograrlo **simulamos** un apartado de la página de pisa donde se podrá ver una lista con los candidatos ya procesados y se pueda filtrar por trabajo,nombre,score, etc.

Para el front end usaremos:

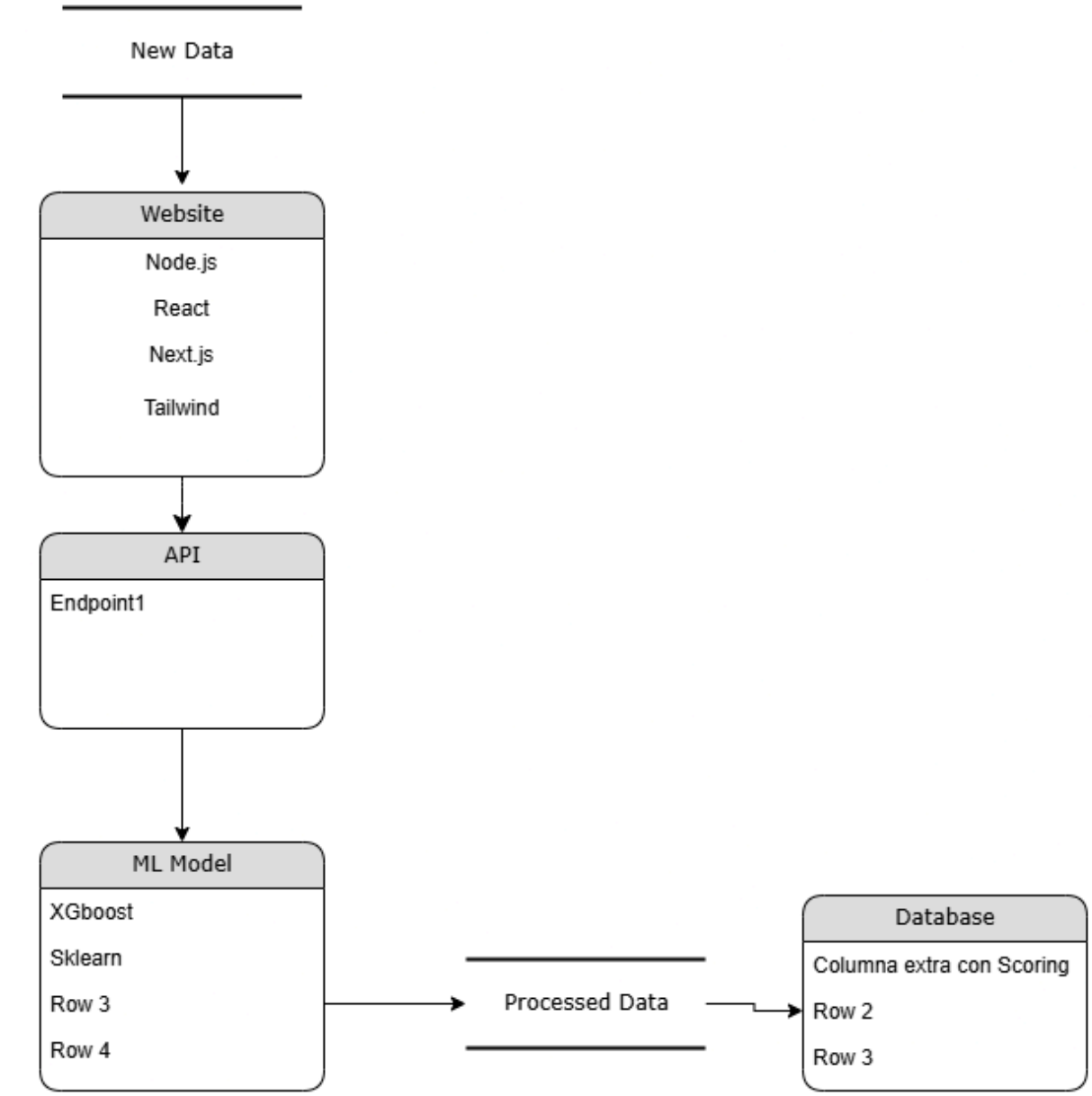
- React, Tailwind, Next

Para el backend:

- PostgreSQL,FastAPI,Django(por decidir)

Dashboard:

- Mostrar estadísticas de los candidatos.



(En construcción)

Requerimientos

1.- Must-Have (M) :

- Analice descripciones de puestos de trabajo a partir de archivos HTML estructurados.
- Modelo de aprendizaje automático para la adecuación candidato-empleo mediante técnicas de procesamiento del lenguaje natural (PNL).
- Un sistema de puntuación para clasificar a los candidatos en función de su adecuación al puesto.

2.- Should-Have (S):

- Reconocimiento de entidades con nombre (NER-Named entity recognition) para extraer atributos clave (por ejemplo, habilidades, educación, experiencia).

3.- Won't-Have (W):

- Envío automatizado de solicitudes de empleo para los candidatos.
- Intervención manual del reclutador para la puntuación.

Método

Procesamiento de datos:

- **Job Description Parsing (HTML)**
  - Utilice **BeautifulSoup** para extraer texto estructurado de HTML.
  - Identifique las secciones pertinentes, como el cargo, las competencias requeridas y la experiencia.
- **Preprocesamiento**
  - Convierte el texto a minúsculas, elimina las palabras vacías y normaliza la puntuación.
  - Tokenize palabras para su posterior procesamiento del lenguaje natural (PNL).

Extracción de características

- Named Entity Recognition (NER)

- Utilizar **Spacy** o **BERT-based NER** para detectar:
  - **Habilidades** (e.g., Python, SQL)
  - **Educación** (e.g., Master's Degree in CS)
  - **Experiencia** (e.g., "5 años en Google")

- Representación estructurada

- Convertir los datos extraídos en un DataFrame de pandas con columnas como:

```
Unset

| Candidate | Skills            | Experience | Education
|
|-----------|-------------------|------------|-----------|
| John Doe  | Python, ML, SQL   | 5 years    | MSc in AI |

| Jane Roe  | Java, React, AWS  | 3 years    | BSc in CS |
```

Figura 1: Representación estructurada de los candidatos en un DataFrame.

```
Unset

| Job       | Skills  needed   | Experience |
|-----------|-------------------|------------|
| Job 1     | Python, ML, SQL   | 5 years    |

| Job 2     | Java, React, AWS  | 3 years    |
```

Figura 2: Representación estructurada de los puestos en un DataFrame.

Machine Learning Model

➢ **Vectorización de datos de texto**

- Utilice técnicas de procesamiento del lenguaje natural (PNL) para representar numéricamente descripciones de puestos de trabajo y CV:
  - **TF-IDF**: Calcula la importancia de las palabras.
  - **Word2Vec / FastText**: Capta el significado de las palabras.
  - **BERT-based embeddings**: Representación profunda consciente del contexto.

➢ **Training a Model**

- Entrenar un modelo de clasificación o ranking para evaluar la similitud:
  - **Regresión logística / Random Forest** (para la correspondencia basada en reglas).
  - **Deep Learning** (Ajuste de BERT para similitud semántica).
- Utilice datos etiquetados siempre que sea posible (CV con resultados laborales conocidos).

Scoring Mechanism

- **Métricas de similitud**
  - Utilice la similitud del coseno para comparar la descripción del puesto de trabajo y las incrustaciones de CV.
  - Alternativamente, similitud de Jaccard (para el solapamiento de palabras clave).

- **Cálculo de la puntuación final**
  - Puntuaciones agregadas por competencias, experiencia y formación.
  - Asigne un porcentaje de coincidencia (por ejemplo, Juan Pérez: 85% de coincidencia).

Pipeline & Workflow

- **Paso 1:** Cargar CV y descripciones de puestos.
- **Paso 2:** Pre Procesar y extraer características clave.
- **Paso 3:** Calcular embebidos y aplicar el modelo ML.
- **Paso 4:** Puntuación y clasificación de los candidatos.
- **Paso 5:** Salida de los candidatos clasificados en un DataFrame.

Implementation

1. Desarrollar un analizador sintáctico de CV y descripciones de puestos de trabajo utilizando bibliotecas de Python (pdfplumber, python-docx, BeautifulSoup).
2. Implementar NER utilizando Spacy o un modelo basado en BERT.
3. Construir el pipeline NLP para la vectorización del texto y el cálculo de similitudes.
4. Entrenar y evaluar el modelo de aprendizaje automático.
5. Desarrollar el algoritmo de puntuación para la clasificación de candidatos.
6. Integrar todos los componentes en una cadena de procesamiento.

Diagrama de Arquitectura

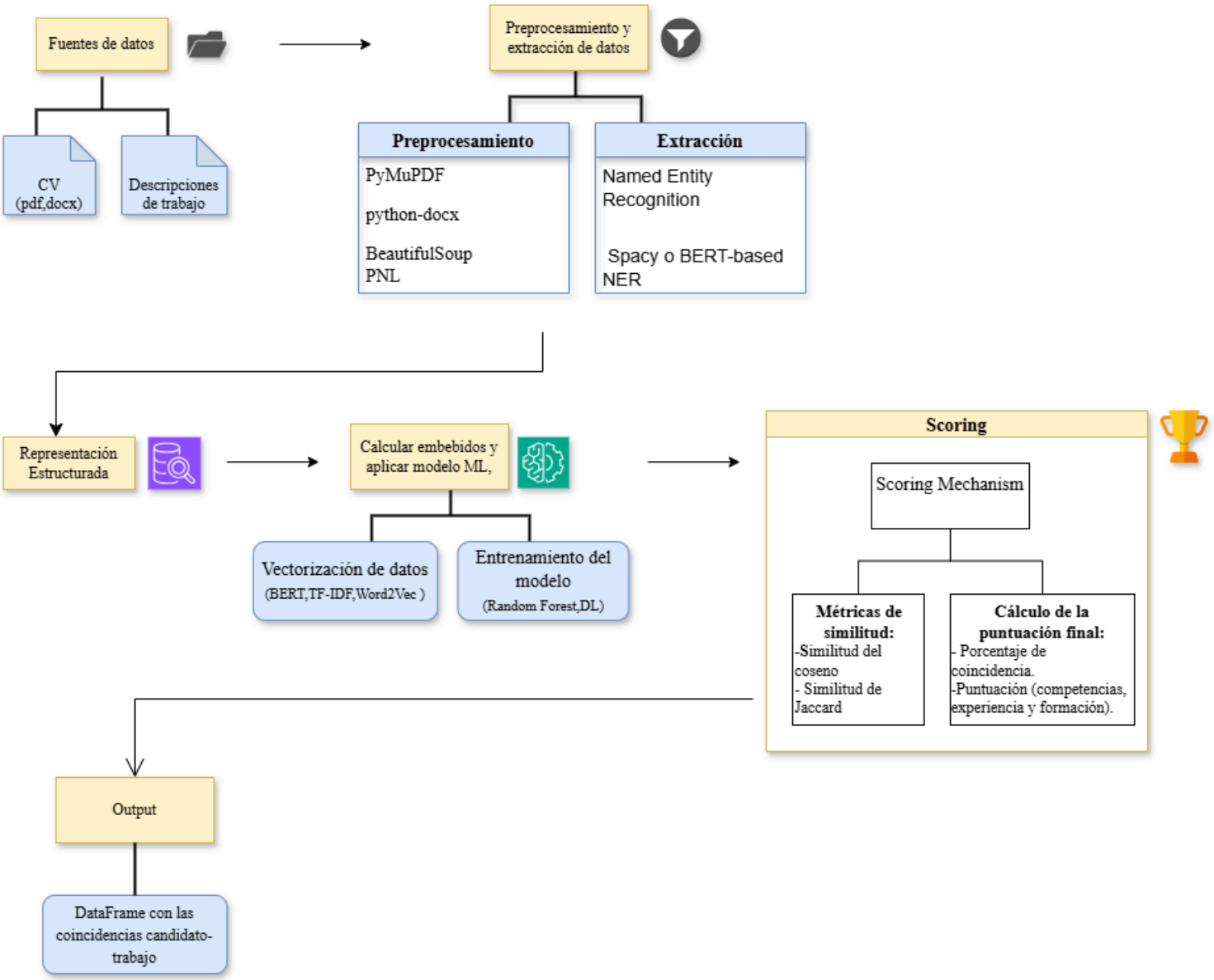


Figura 3: Diagrama de arquitectura del proyecto.

Recopilación de resultados

- Compare las clasificaciones de los candidatos generadas por el sistema con las de los reclutadores humanos.
- Medir la precisión y el rendimiento del modelo ML (precisión, recuperación, puntuación F1).