Prueba Técnica: Scraping de Información en Página Web

Extracción automatizada de datos de contacto empresarial desde fuentes públicas

Autora: Amelia Cristina Herrera Briceño
Data Analyst & Scientist en transición hacia BI y automatización

Entorno de trabajo: Python · Pandas · BeautifulSoup · Regex · CSV · Exploración reproducible

Fecha Entrega: 15 Agosto 2025

1. Problema

Se cuenta con listados de empresas (nombre y CIF), pero se carece de sus datos de contacto directo (emails corporativos, teléfonos, direcciones). Para establecer comunicación comercial, es necesario localizar y extraer automáticamente esta información desde sus sitios web oficiales u otras fuentes públicas disponibles en internet.

2. Solución Técnica

Desarrollar un proceso de scraping web que permita obtener datos de contacto empresariales de forma automatizada a partir de los nombres de las empresas.

3. Alcance de la Prueba

- **Dataset:** Muestra de 50–100 empresas españolas (nombre + CIF).
- **Proceso esperado:**
    1. Localizar la web corporativa oficial de cada empresa.
    2. Extraer datos de contacto mediante scraping web.
- **Datos objetivo:**
    - URL oficial
    - Email corporativo principal (info@, contacto@, comercial@)
    - Teléfono de contacto
    - Dirección postal completa
- **Entregables:**
    - Archivo CSV/Excel con los datos obtenidos
    - Documentación técnica del proceso
    - Registro de limitaciones (robots.txt, captchas, etc.)
    - Tiempo estimado de procesamiento por empresa
    - Coste estimado del proceso
    - Video explicativo del flujo técnico

4. Implementación Técnica

- **Scraping sincrónico reproducible**: requests + BeautifulSoup + regex
- **Limpieza post-scraping**: DataFrames exportables, validación de encoding y estructura
- **Exportación**: Archivos CSV listos para análisis y visualización

5. Métricas y Evaluación

- **Tasa de éxito**: calculada por porcentaje de empresas con datos completos
- **Limitaciones encontradas**: robots.txt, captchas, páginas sin datos visibles
- **Tiempo estimado por empresa**: medido con barra de progreso
- **Coste estimado**: bajo en entorno local; escalable con APIs

6. Video Explicativo

Se graba un video mostrando el flujo, explicando el código y los resultados obtenidos.

7. Escalabilidad

Para flujos masivos, se propone una versión asincrónica con aiohttp o crawl4ai, que mejora el rendimiento y permite scraping concurrente.

8. Comparativa de Métodos de Búsqueda de URLs Oficiales

Método	Gratuito	Límite	Costo estimado	Ventajas	Desventajas
googlesearch (scraping)	Sí	No oficial	$0	Sin registro, ideal para pruebas	Bloqueos, resultados no estructurados
SerpAPI	Sí (250/mes)	Pago desde $50/mes	~$50	JSON limpio, sin bloqueo	Requiere API Key, coste mensual
Zenserp	Sí (50/día)	Pago desde $29/mes	~$29	Fácil de usar, resultados precisos	Menor volumen gratuito
Google Custom Search API	Sí (100/día)	$5/1000 búsquedas	~$5	Oficial, configurable	Requiere configuración previa

Método	Gratuito	Límite	Costo estimado	Ventajas	Desventajas
Bing Search API	Sí (1000/mes)	Pago desde $3/1000	~$3	Económica, buena cobertura	Menor precisión en empresas locales

9. Recomendación Final

Para esta prueba técnica (50–100 empresas), se recomienda utilizar googlesearch por su simplicidad y coste cero. Para escalar el proceso, se sugiere evaluar APIs comerciales como SerpAPI y adoptar scraping asincrónico con aiohttp o crawl4ai.

10. Corrección Fonética y Semántica de Nombres Empresariales

Objetivo del flujo:

Corregir nombres empresariales con validación fonética (rapidfuzz) y semántica (unidecode), asegurando trazabilidad y limpieza formal.

Entorno virtual limpio:

- Creado con python -m venv .venv_limpio
- Librerías instaladas: pandas, rapidfuzz, unidecode, jupyterlab (opcional)

Ejemplo de corrección fonética:

python

```
import re
from unidecode import unidecode

def normalizar_nombre(nombre):
    nombre = unidecode(nombre)
    nombre = re.sub(r'[^\w\s]', '', nombre)
    nombre = nombre.strip().upper()
```

```
    return nombre

nombre_1 = "Compañía ABC S.A."
nombre_2 = "Compania - ABC, SA"

print(normalizar_nombre(nombre_1))  # COMPANIA ABC SA
print(normalizar_nombre(nombre_2))  # COMPANIA ABC SA
```

11. Flujo de Corrección y Validación

Archivos y propósito principal

Archivo	**Propósito principal**
normalizacion.py	Funciones para limpiar y normalizar nombres
correccion.py	Corrección fonética con RapidFuzz y validación semántica
validacion.py	Detección de correcciones sospechosas y revisión manual
main.py	Orquestación del flujo completo

12. Archivos CSV y Logs Funcionales

Archivo CSV	**Rol en el flujo**	**Contenido esperado**	**Sugerencias de mejora**
empresas_limpias_ corregidas_mejorado.csv	Post-normalización y corrección automática	Nombres limpios + correcciones automáticas	Agregar columna ORIGEN_CORRECCIÓN y FECHA_PROCESO
revision_manual.csv	Correcciones manuales aplicadas	Casos límite revisados por uno mismo	Agregar columna OBSERVACIONES y VALIDACIÓN_MANUAL
correcciones_ sospechosas.csv	Casos con ambigüedad o errores detectados	Correcciones dudosas, posibles sobreajustes	Agregar columna TIPO_ERROR y RECOMENDACIÓN

Archivo CSV	Rol en el flujo	Contenido esperado	Sugerencias de mejora
empresas_limpias_ corregidas_final.csv	Resultado final validado	Nombres corregidos y validados	Agregar columna VALIDACIÓN_FINAL y FUENTE_CORRECCIÓN
log_de_correcciones.csv	Registro trazable de cada corrección	Entradas, salidas, tipo de corrección	Ya bien estructurado, solo falta ID y TIMESTAMP

13. Informe de Cobertura y Limitaciones del Scraping

Resultados generales

- Empresas procesadas: **100**
- Con datos completos: **19**
- Con datos parciales: **81**
- Sin datos encontrados: **0**
- Costo estimado por empresa útil: **1.89 horas**

Hallazgos clave

1. Datos incompletos en la página principal
2. Carga asincrónica de contenido
3. URLs inválidas o redireccionadas
4. Datos mezclados o mal formateados

14. Mejoras Sugeridas

- Ajustar patrones de regex
- Filtrar URLs sin http
- Documentar casos sin datos
- Explorar subpáginas automáticamente
- Usar Selenium o Playwright

- Consultar APIs internas
- Registrar logs por empresa

15. Reflexión Final

Este ejercicio permitió identificar las limitaciones del scraping tradicional y la necesidad de modularizar el flujo. Se documentaron errores como evidencia de aprendizaje y se dejaron mejoras futuras para auditoría y enseñanza.

16. Anexo: Tiempo y Esfuerzo Invertido

Tareas realizadas

- Preparación del entorno virtual
- Limpieza y validación de nombres
- Gestión de asincronía y subpáginas
- Documentación de incidentes
- Comparación de flujos
- Generación de evidencia

Tiempo total invertido ~36 horas distribuidas en 3 días intensivos

17. Tabla de Incidentes Técnicos y Validaciones

Tipo de incidente	Descripción breve	Ejemplo o patrón detectado	Mejora futura sugerida
#asincronía	La carga de datos depende de JavaScript o peticiones dinámicas	Empresas con páginas que no muestran datos en HTML estático	Usar herramientas como Selenium o capturas manuales

Tipo de incidente	Descripción breve	Ejemplo o patrón detectado	Mejora futura sugerida
#estructura_inaccesible	El sitio no tiene estructura clara o los datos están embebidos en imágenes	Empresas con contacto solo en banners o PDF	Documentar como “no extraíble automáticamente”
#subpágina_oculta	Los datos están en subpáginas no enlazadas directamente desde el home	Empresas con sección “Contacto” separada sin enlace directo	Implementar detección de enlaces internos relevantes
#validación_manual	Se requiere criterio humano para interpretar nombres o datos ambiguos	Empresas con nombres genéricos o duplicados	Dejar constancia de ejemplos y diccionario personalizado
#datos_incompletos	Se extrae solo parte del contacto (ej. solo email o solo teléfono)	Empresas con formularios sin email visible	Marcar como “parcial” y registrar en log
#estructura_inconsistente	El HTML varía entre empresas, dificultando el uso de selectores únicos	Empresas con plantillas distintas o CMS personalizado	Modularizar el scraping por tipo de estructura detectada
#bloqueo_scraping	El sitio detecta scraping y bloquea o redirige	Empresas con Cloudflare o captcha	Documentar como “requiere intervención manual”

Tipo de incidente	Descripción breve	Ejemplo o patrón detectado	Mejora futura sugerida
#errores_de_entorno	Problemas con librerías, rutas o entornos virtuales durante la ejecución	Conflictos entre kernels o rutas relativas	Registrar en README y limpiar entornos obsoletos

//.