📚 Documentación Universal Downloader

📖 Índice

🎯 Descripción General

Universal Downloader es una herramienta de línea de comandos desarrollada en Python que permite descargar contenido de YouTube y YouTube Music de forma inteligente y automatizada. El script detecta automáticamente el tipo de contenido (música o video) y aplica la configuración óptima para cada caso.

Filosofía de Diseño

- **Detección automática**: Distingue entre música y video sin intervención del usuario
- **Prevención de duplicados**: Sistema de registro para evitar descargas repetidas
- **Flexibilidad**: Múltiples modos de operación y configuración personalizable
- **Robustez**: Manejo completo de errores y logging detallado
- **Usabilidad**: Interfaz rica con indicadores de progreso y notificaciones

---

⭐ Características Principales

🔍 Detección Inteligente

- **Auto-detección de contenido**: Identifica automáticamente si es música o video
- **Optimización por tipo**: Aplica configuraciones específicas según el contenido
- **Formatos adaptativos**: Selecciona el mejor formato disponible

🚫 Prevención de Duplicados

- **Registro persistente**: Archivo JSON que almacena URLs descargadas
- **Verificación automática**: Omite URLs ya procesadas
- **Logging completo**: Registro detallado de todas las operaciones

🎛️ Modos de Operación

- **Modo URL individual**: Descarga directa de una URL
- **Modo archivo**: Procesamiento por lotes desde archivo de texto
- **Modo Watch**: Supervisión continua de directorio

🔧 Configuración Flexible

- **Archivo YAML**: Configuración centralizada y editable
- **Calidades personalizables**: Múltiples opciones de calidad de video
- **Rutas configurables**: Directorios de salida personalizables

---

🏗️ Arquitectura del Código

Estructura Modular

```
universal_downloader.py
├── Configuración
│   ├── cargar_configuracion()
│   └── config (variable global)
├── Utilidades
│   ├── is_music_url()
│   ├── ya_descargado()
│   ├── registrar_descarga()
│   ├── log_event()
│   ├── notificar_mensaje()
│   ├── leer_urls_desde_archivo()
│   └── crear_directorios()
├── Motor de Descarga
│   ├── download_audio()
│   ├── download_video()
│   ├── progreso_hook()
│   └── procesar_url()
├── Modo Watch
│   ├── WatchHandler (clase)
│   └── modo_watch()
└── Interfaz CLI
    └── main()
```

Flujo de Ejecución

```mermaid
graph TD
    A[Inicio] --> B[Cargar Configuración]
    B --> C[Parsear Argumentos]
    C --> D{Modo de Operación}
    D -->|URL Individual| E[Procesar URL Única]
    D -->|Archivo| F[Leer URLs del Archivo]
    D -->|Watch| G[Iniciar Supervisión]
    E --> H[Verificar Duplicados]
    F --> I[Iterar URLs]
    G --> J[Configurar Observer]
    H --> K{¿Ya descargado?}
    I --> H
    K -->|No| L[Detectar Tipo]
    K -->|Sí| M[Omitir]
    L --> N{¿Es música?}
    N -->|Sí| O[Descargar Audio]
    N -->|No| P[Descargar Video]
    O --> Q[Registrar Descarga]
    P --> Q
    Q --> R[Log y Notificación]
    J --> S[Escuchar Eventos]
    S --> T[Procesar Archivos]
    T --> H
```

⚙️ Configuración

Archivo config.yaml

```yaml
# Configuración por defecto si no existe archivo
rutas:
  musica: './Musica'
  videos: './Videos'
  watch: './watch'

formatos:
  audio: 'mp3'
  video: 'mp4'

calidades:
  baja: '480'
  media: '720'
  alta: '1080'

watch:
  intervalo_revision: 5
  procesar_subdirectorios: false
  eliminar_archivo_procesado: true
  formatos_permitidos: ['.txt', '.urls', '.list']
```

Parámetros Configurables

Sección	Parámetro	Descripción	Valor por Defecto
`rutas`	`musica`	Directorio para archivos de audio	`'./Musica'`
`rutas`	`videos`	Directorio para archivos de video	`'./Videos'`
`rutas`	`watch`	Directorio supervisado	`'./watch'`
`formatos`	`audio`	Formato de salida para audio	`'mp3'`
`formatos`	`video`	Formato de salida para video	`'mp4'`
`calidades`	`baja/media/alta`	Resoluciones de video	`480/720/1080`
`watch`	`intervalo_revision`	Segundos entre revisiones	`5`
`watch`	`eliminar_archivo_procesado`	Eliminar archivo tras procesar	`true`

🎮 Modos de Operación

1. Modo URL Individual

Uso: `universal_downloader "URL"`

bash

```bash
# Ejemplos
universal_downloader "https://www.youtube.com/watch?v=dQw4w9WgXcQ"
universal_downloader -q alta "https://www.youtube.com/watch?v=dQw4w9WgXcQ"
```

Comportamiento:

- Detecta automáticamente si es música o video
- Aplica calidad especificada (solo para videos)
- Verifica duplicados antes de descargar
- Registra la descarga al completarse

2. Modo Archivo

Uso: `universal_downloader -f archivo.txt`

bash

```bash
# Ejemplo
universal_downloader -f urls.txt -q media
```

Formato del archivo:

```
# Comentarios empiezan con #
https://www.youtube.com/watch?v=video1
https://music.youtube.com/watch?v=audio1
https://www.youtube.com/watch?v=video2

# URLs vacías son ignoradas
https://www.youtube.com/watch?v=video3
```

Comportamiento:

- Procesa URLs línea por línea
- Ignora comentarios (líneas que empiezan con #)
- Omite líneas vacías
- Aplica verificación de duplicados a cada URL
- Muestra progreso por lotes

3. Modo Watch

Uso: `universal_downloader -w`

```bash
# Ejemplo
universal_downloader -w -q alta
```

Comportamiento:

- Supervisa continuamente el directorio `watch`
- Procesa archivos con extensiones permitidas (`.txt`, `.urls`, `.list`)
- Detecta archivos nuevos y modificados
- Procesa archivos existentes al iniciar
- Elimina archivos procesados (configurable)

Flujo del Modo Watch:

1. Crear directorios necesarios
2. Procesar archivos existentes
3. Configurar observador de sistema de archivos
4. Escuchar eventos de creación/modificación
5. Procesar nuevos archivos automáticamente

🔧 API de Funciones

Funciones de Configuración

`cargar_configuracion(path="config.yaml")`

Carga la configuración desde archivo YAML o usa valores por defecto.

Parámetros:

- `path` (str): Ruta al archivo de configuración

Retorna: dict - Diccionario con la configuración

Ejemplo:

```python
config = cargar_configuracion("mi_config.yaml")
```

Funciones de Utilidad

`is_music_url(url)`

Detecta si una URL pertenece a YouTube Music.

Parámetros:

- `url` (str): URL a verificar

Retorna: bool - True si es de YouTube Music

Ejemplo:

```python
if is_music_url("https://music.youtube.com/watch?v=abc"):
    print("Es música")
```

`ya_descargado(url, registro_path="descargados.json")`

Verifica si una URL ya fue descargada previamente.

Parámetros:

- `url` (str): URL a verificar
- `registro_path` (str): Ruta al archivo de registro

Retorna: bool - True si ya fue descargada

`registrar_descarga(url, registro_path="descargados.json")`

Registra una URL como descargada.

Parámetros:

- `url` (str): URL a registrar
- `registro_path` (str): Ruta al archivo de registro

`log_event(message, log_file="descargas.log")`

Registra un evento en el archivo de log con timestamp.

Parámetros:

- `message` (str): Mensaje a registrar
- `log_file` (str): Archivo de log

`leer_urls_desde_archivo(archivo_path)`

Lee URLs desde un archivo de texto.

Parámetros:

- `archivo_path` (str): Ruta al archivo

Retorna: list - Lista de URLs válidas

Funciones de Descarga

`download_audio(url)`

Descarga solo el audio de una URL.

Configuración yt-dlp:

```python
ydl_opts = {
    'format': 'bestaudio/best',
    'outtmpl': os.path.join(config['rutas']['musica'], '%(title)s.%(ext)s'),
    'postprocessors': [{
        'key': 'FFmpegExtractAudio',
        'preferredcodec': config['formatos'].get('audio', 'mp3'),
        'preferredquality': '192',
    }],
}
```

`download_video(url, quality)`

Descarga video con calidad específica.

Parámetros:

- `url` (str): URL del video
- `quality` (str): Calidad deseada

Configuración yt-dlp:

```python
ydl_opts = {
    'format': f'bestvideo[height<={quality}]+bestaudio/best[height<={quality}]',
    'outtmpl': os.path.join(config['rutas']['videos'], '%(title)s.%(ext)s'),
    'merge_output_format': config['formatos'].get('video', 'mp4'),
    'progress_hooks': [progreso_hook],
}
```

`procesar_url(url, quality="media")`

Función principal que procesa una URL individual.

Flujo:

1. Verificar si ya fue descargada
2. Detectar tipo de contenido
3. Llamar función de descarga apropiada
4. Registrar descarga exitosa
5. Log y notificaciones

Clase WatchHandler

Hereda de `FileSystemEventHandler` para manejar eventos del sistema de archivos.

Métodos Principales:

`on_created(event)`: Maneja archivos recién creados `on_modified(event)`: Maneja archivos modificados `procesar_archivo(archivo_path)`: Procesa un archivo específico

Características:

- Previene procesamiento múltiple del mismo archivo
- Verifica extensiones permitidas
- Maneja archivos vacíos o inaccesibles
- Elimina archivos procesados (opcional)

🚨 Manejo de Errores

Sistema de Logging

El script implementa un sistema robusto de logging en múltiples niveles:

1. Log de Eventos (`descargas.log`)

```
[2024-05-23 14:30:15] ÉXITO: https://www.youtube.com/watch?v=ejemplo
[2024-05-23 14:31:22] ERROR: https://invalid.url | Video unavailable
[2024-05-23 14:32:10] OMITIDA: https://www.youtube.com/watch?v=duplicado
```

2. Salida de Consola con Colores

- **Verde**: Operaciones exitosas
- **Amarillo**: Advertencias y URLs omitidas
- **Rojo**: Errores

- **Azul**: Información general
- **Cian**: Proceso de descarga

3. Notificaciones del Sistema

- **Linux**: `notify-send`
- **macOS**: `osascript`
- **Windows**: `win10toast` (opcional)

Tipos de Errores Manejados

Errores de Red

- Conexión perdida durante descarga
- Video no disponible
- URL inválida

Errores de Sistema

- Permisos de escritura
- Espacio en disco insuficiente
- Archivos bloqueados

Errores de Configuración

- Archivo config.yaml malformado
- Directorios inaccesibles
- Dependencias faltantes

Estrategias de Recuperación

1. **Reintentos automáticos**: yt-dlp maneja reintentos internamente
2. **Omisión de errores**: Continúa con siguiente URL en modo lote
3. **Logging detallado**: Permite diagnóstico post-ejecución
4. **Validación previa**: Verifica archivos y URLs antes de procesar

---

📋 Ejemplos de Uso

Ejemplos Básicos

bash

```bash
# Descargar un video
universal_downloader "https://www.youtube.com/watch?v=dQw4w9WgXcQ"

# Descargar música de YouTube Music
universal_downloader "https://music.youtube.com/watch?v=dQw4w9WgXcQ"

# Especificar calidad
universal_downloader -q alta "https://www.youtube.com/watch?v=dQw4w9WgXcQ"
```

Ejemplos Avanzados

bash

```bash
# Procesar lista de URLs
universal_downloader -f playlist.txt -q media

# Modo watch con calidad alta
universal_downloader -w -q alta

# Usando alias corto
udl "https://www.youtube.com/watch?v=dQw4w9WgXcQ"
```

Archivo de URLs (ejemplo)

txt

```
# Mi playlist favorita
https://www.youtube.com/watch?v=dQw4w9WgXcQ
https://music.youtube.com/watch?v=L_jWHffIx5E

# Videos educativos
https://www.youtube.com/watch?v=educativo1
https://www.youtube.com/watch?v=educativo2

# Música relajante
https://music.youtube.com/watch?v=relax1
https://music.youtube.com/watch?v=relax2
```

Casos de Uso Prácticos

1. Descarga Masiva Nocturna

bash

```bash
# Crear archivo con URLs del día
echo "https://www.youtube.com/watch?v=nuevo1" >> descargas_hoy.txt
echo "https://www.youtube.com/watch?v=nuevo2" >> descargas_hoy.txt

# Programar descarga (ejemplo con cron en Linux)
# 0 2 * * * /usr/local/bin/universal_downloader -f /home/user/descargas_hoy.txt
```

2. Supervisión Automática

bash

```bash
# Terminal 1: Iniciar modo watch
universal_downloader -w

# Terminal 2: Agregar URLs dinámicamente
echo "https://www.youtube.com/watch?v=nuevo" > watch/nuevas.txt
```

3. Organización por Calidad

bash

```bash
# Videos HD para archivo
universal_downloader -f importantes.txt -q alta

# Videos rápidos para revisión
universal_downloader -f revision.txt -q baja
```

📦 Dependencias

Dependencias Principales

Paquete	Versión	Propósito
`yt-dlp`	Latest	Motor de descarga principal
`pyyaml`	Latest	Parsing de configuración YAML
`rich`	Latest	Interfaz rica de consola
`watchdog`	Latest	Supervisión de sistema de archivos

Dependencias del Sistema

Sistema	Dependencia	Propósito
Todos	`ffmpeg`	Procesamiento de audio/video
Windows	`win10toast`	Notificaciones (opcional)
Linux	`libnotify-bin`	Notificaciones
macOS	Integrado	Notificaciones AppleScript

Instalación de Dependencias

```bash
# Python packages
pip install yt-dlp pyyaml rich watchdog

# FFmpeg (Ubuntu/Debian)
sudo apt install ffmpeg

# FFmpeg (Windows - Chocolatey)
choco install ffmpeg

# FFmpeg (macOS - Homebrew)
brew install ffmpeg
```

📁 Estructura de Archivos

Estructura por Defecto

```
proyecto/
├── universal_downloader.py     # Script principal
├── config.yaml                 # Configuración (opcional)
├── setup_windows.bat           # Instalador Windows
├── setup.sh                    # Instalador Linux/macOS
├── INSTALACION.md              # Guía de instalación
├── descargados.json            # Registro de descargas (generado)
├── descargas.log               # Log de eventos (generado)
├── Musica/                     # Descargas de audio (generado)
│   ├── cancion1.mp3
│   └── cancion2.mp3
├── Videos/                     # Descargas de video (generado)
│   ├── video1.mp4
│   └── video2.mp4
└── watch/                      # Carpeta supervisada (generado)
    ├── urls.txt
    └── playlist.list
```

Archivos Generados

`descargados.json`

```json
[
  "https://www.youtube.com/watch?v=dQw4w9WgXcQ",
  "https://music.youtube.com/watch?v=L_jWHffIx5E"
]
```

`descargas.log`

```
[2024-05-23 14:30:15] ÉXITO: https://www.youtube.com/watch?v=dQw4w9WgXcQ
[2024-05-23 14:31:22] ERROR: https://invalid.url | Video unavailable
[2024-05-23 14:32:10] OMITIDA: https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

Personalización de Rutas

Puedes personalizar todas las rutas en `config.yaml`:

```yaml
rutas:
  musica: '/home/user/Musica'
  videos: '/home/user/Videos'
  watch: '/home/user/DropZone'
```

🔍 Solución de Problemas Comunes

Error: "yt-dlp not found"

```bash
pip install yt-dlp
# o
pip3 install yt-dlp
```

Error: "ffmpeg not found"

Instalar FFmpeg según tu sistema operativo (ver sección de dependencias).

Error: "Permission denied"

bash

```bash
# Linux/macOS
chmod +x universal_downloader.py

# Windows
# Ejecutar como administrador si es necesario
```

URLs no se procesan en modo watch

- Verificar extensiones permitidas en config.yaml
- Verificar permisos de la carpeta watch
- Revisar logs para errores específicos

Descargas lentas

- Verificar conexión a internet
- Considerar cambiar servidor DNS
- Verificar que no hay límites de ancho de banda

🚀 Tips de Rendimiento

1. **Descargas en paralelo**: El script procesa URLs secuencialmente por diseño para evitar sobrecarga
2. **Calidad óptima**: Usar "media" (720p) para balance entre calidad y velocidad
3. **Limpieza regular**: Eliminar archivos de log antiguos periódicamente
4. **SSD recomendado**: Para mejor rendimiento en escritura de archivos

---

📝 Notas de Desarrollo

Extensibilidad

El código está diseñado para ser fácilmente extensible:

- **Nuevos formatos**: Modificar configuración yt-dlp
- **Nuevas fuentes**: Extender detección de URLs
- **Nuevas notificaciones**: Agregar métodos en `notificar_mensaje()`
- **Nuevos modos**: Seguir patrón existente en `main()`

Consideraciones de Seguridad

- No almacena credenciales
- Valida extensiones de archivos en modo watch

- Sanitiza nombres de archivos automáticamente
- Logs no contienen información sensible

Compatibilidad

- **Python**: 3.7+
- **Sistemas**: Windows 10+, Linux (Ubuntu 18+), macOS 10.14+
- **Arquitecturas**: x86, x64, ARM64 (Apple Silicon)

Documentación generada para Universal Downloader v1.0