ComicAI

Tabla de contenido

Sistema de Narración de Cómics para Personas con Discapacidad Visual. El presente proyecto pretende ayudar a las personas con discapacidad visual a disfrutar del mundo del comic a través de la narración de las historias gráficas.

Objeto

Convertir las imágenes y el texto, tanto de los bocadillos de las conversaciones como de las onomatopeyas en audio, representa un desafío fascinante en el campo de la narración digital y la accesibilidad. Este proceso implica una cuidadosa transcripción y adaptación de cada elemento visual y textual para crear una experiencia auditiva.
En primer lugar, la conversión de las imágenes requiere una descripción detallada de cada escena, capturando todos los aspectos visuales importantes. Esto incluye no solo la descripción de los personajes y sus acciones, sino también los matices del entorno, la atmósfera y cualquier otro detalle que contribuya a la narrativa visual. Por ejemplo, una viñeta que muestra un paisaje oscuro y tormentoso debe transmitir esa sensación de inquietud y dramatismo a través de la narración, utilizando palabras y tonos adecuados para evocar las mismas emociones en el oyente.
Los bocadillos de las conversaciones, por su parte, deben ser transformados en diálogos claros y expresivos. En cuanto a las onomatopeyas, estas son una parte crucial del ambiente sonoro en cualquier narración visual, especialmente en cómics y novelas gráficas. Palabras como "¡BANG!", "¡CRASH!" o "¡ZAP!" deben ser convertidas en efectos de sonido que correspondan con su representación escrita. Estos sonidos no solo deben ser precisos, sino que también deben sincronizarse perfectamente con la narrativa para mantener la coherencia y el ritmo de la historia.
En resumen, la transformación de las imágenes y el texto, incluyendo las conversaciones y las onomatopeyas, en un formato auditivo requiere una combinación de habilidades técnicas y artísticas. Cada detalle, desde la descripción de una escena hasta la reproducción de un sonido, contribuye a crear una experiencia auditiva que no solo cuenta una historia, sino que también transporta al oyente a los mundos imaginados por los creadores originales. Este proceso amplía la accesibilidad de la obra a personas con discapacidades visuales.

Estructura de y funcionalidad de los módulos

La cadena de procesamiento del proyecto se lleva a cabo mediante una serie de operaciones sucesivas, cada una implementada a través de módulos individuales que funcionan de manera secuencial:

- 1_divide_page

Divide el archivo original .pdf (el cual puede ser un comic completo o capítulos individuales) en páginas en formato .jpeg.

En este código se utilizan varias librerías para diferentes propósitos: fitz (de PyMuPDF) se emplea para abrir y manipular documentos PDF, específicamente para extraer las imágenes de cada página; os se usa para interactuar con el sistema operativo, permitiendo la manipulación de directorios y archivos, como la creación, eliminación y listado de contenidos; shutil se utiliza para operaciones de alto nivel en archivos y colecciones de archivos, en este caso para eliminar subdirectorios; y csv se utiliza para escribir en archivos CSV, en este caso, para crear un archivo que contiene las rutas de las imágenes extraídas del PDF.

- 2_identify_panels

Identifica cada viñeta de las páginas del paso anterior y la guarda el formato .jpeg.

En este código se utilizan varias librerías para diferentes propósitos: cv2 (OpenCV) se usa para la manipulación y procesamiento de imágenes, incluyendo la lectura, conversión a escala de grises, aplicación de desenfoque, detección de bordes y dibujo de contornos; os se emplea para interactuar con el sistema operativo, permitiendo la manipulación de directorios y archivos, como la creación, eliminación y listado de contenidos; shutil se utiliza para operaciones de alto nivel en archivos y colecciones de archivos, en este caso, para eliminar directorios y su contenido; y matplotlib.pyplot se utiliza para la visualización de imágenes con los rectángulos de las viñetas dibujados, ajustando el tamaño de la figura y mostrando la imagen procesada.

- 3_bubbles_to_text

Indetifica el diálogo entre los personajes y el texto que aparece en la viñeta como onomatopeyas o descripciones. Esta descripción es en español

En este código se utilizan varias librerías para diferentes propósitos: torch (PyTorch) se usa para verificar la disponibilidad de CUDA y para manejar el dispositivo (CPU o GPU) en el que se ejecutarán los modelos de procesamiento de texto e imagen; transformers de Hugging Face se emplea para cargar los modelos preentrenados de TrOCR y su procesador, permitiendo realizar el reconocimiento óptico de caracteres en las imágenes; easyocr se utiliza para detectar y leer texto en imágenes sin necesidad de CUDA. En este caso se utiliza para el reconocimiento de las zonas donde hay texto, ya que el reconocimiento del propio texto se hace con el modelo finetuned de trOCR; cv2 (OpenCV) se usa para cargar y manipular imágenes, incluyendo el recorte y dibujo de cajas delimitadoras alrededor de las áreas de texto detectado; os se utiliza para interactuar con el sistema operativo, permitiendo la navegación por los directorios de entrada y salida, así como la creación de nuevas carpetas; matplotlib.pyplot se usa para visualizar las imágenes con las cajas delimitadoras dibujadas; y PIL (Pillow) se emplea para convertir imágenes de formato OpenCV a formato PIL para su procesamiento por los modelos de Hugging Face.

- 4_img_to_nlp

Realiza la descripción de de lo que ocurre en la viñeta, ya sea la descripción del entorno o de los propios peronajes. Esta descripción es en inglés

En este código se utilizan varias librerías para diferentes propósitos: torch (PyTorch) se emplea para verificar la disponibilidad de CUDA y manejar el dispositivo (CPU o GPU) donde se ejecutarán los modelos, así como para cargar y mover el modelo afinado al dispositivo; transformers de Hugging Face se utiliza para cargar los modelos preentrenados BLIP para la generación de descripciones de imágenes y sus procesadores correspondientes, permitiendo realizar el captioning de las imágenes; os se usa para interactuar con el sistema operativo, permitiendo la navegación por los directorios de entrada y salida, así como la creación de nuevas carpetas; PIL (Pillow) se utiliza para cargar y convertir imágenes a formato RGB para su procesamiento; y matplotlib.pyplot se emplea para visualizar las imágenes junto con las descripciones generadas por los modelos.

- 5_translate

Traduce el texto generado el paso anterior para adaptarlo al español.

En este código se utilizan varias librerías para diferentes propósitos: deep_translator se emplea para traducir texto usando la clase GoogleTranslator, permitiendo la traducción de textos de inglés a español; os se usa para interactuar con el sistema operativo, permitiendo la navegación por los directorios de entrada y salida, así como la creación de nuevas carpetas; os.path se utiliza para manipular las rutas de archivos y directorios, facilitando la construcción

de rutas absolutas y relativas; y builtins se emplea para operaciones de lectura y escritura de archivos, permitiendo la manipulación del contenido de los archivos de texto que se encuentran en el directorio de entrada y la escritura de las traducciones en el directorio de salida.

- 6_tts

Por último se narra el texto generado en los pasos 3 y 5. En este paso también se unen ambos texto para que tengan coherencia dentro de una viñeta. Cada archivo de audio se genera individualmene para cada viñeta

En este código se utilizan varias librerías para diferentes propósitos: gtts: La librería gtts (Google Text-to-Speech) se usa para convertir texto a audio. Utiliza el servicio de TTS de Google para generar archivos de audio en diferentes idiomas. moviepy.editor: La librería moviepy se utiliza para manipular archivos de audio y video. En este caso, se usa para concatenar clips de audio. pathlib.Path: Proporciona una manera de trabajar con rutas de archivos y directorios de forma más intuitiva y moderna que el módulo os.path. os: La librería os se usa para interactuar con el sistema operativo, permitiendo navegar por directorios y crear carpetas si es necesario.

Obstáculos

El proyecto ha enfrentado una serie de problemas a medida que evolucionaba. A continuación, se presenta una lista de los principales desafíos y cómo se han resuelto:

1. Creación de directorios durante la ejecución de la cadena de procesamiento: Inicialmente, la ausencia de directorios necesarios causaba errores en la ejecución. Este problema se solucionó añadiendo una función que verifica la existencia de los directorios y los crea si no están presentes.

2. Identificación de las viñetas con OpenCV: La identificación inicial de las viñetas no era suficientemente precisa para un Producto Mínimo Viable (PMV). Este problema se resolvió ajustando los parámetros de relación de aspecto y área total, lo que permitió seleccionar viñetas dentro de un rango específico de áreas.

3. Reconocimiento de texto manuscrito: Se comenzó utilizando la biblioteca EasyOCR, pero no era lo suficientemente robusta para los cómics. Por lo tanto, se optó por el modelo TrOCR de Microsoft (https://huggingface.co/microsoft/trocr-large-handwritten). Se realizó un ajuste fino (fine-tuning) del modelo preentrenado utilizando 11 capítulos del cómic, equivalentes a 136 fotografías con texto manuscrito, el cual fue transcrito manualmente para cada una de ellas. Finalmente se realizó una combinación de easyOCR para detectar la zona donde hay texto y trOCR para la transcripción del texto.

4. Descripción de viñetas en formato cómic: El modelo BLIP (https://huggingface.co/Salesforce/blip-image-captioning-large) utilizado inicialmente no podía reconocer adecuadamente las imágenes en formato cómic, ya que estaba entrenado para imágenes reales. La solución fue realizar un ajuste fino del modelo con 11 capítulos, equivalentes a 346 imágenes, donde se realizaron descripciones manuales para cada una.

5. Problemas al combinar los audios de descripción de viñetas y texto: La combinación correcta de estos audios requería una estructura adecuada de directorios y archivos. En caso de que una viñeta no tuviera descripción o texto, se generaba un audio vacío para luego combinarlo correctamente con su par correspondiente.

Cada uno de estos problemas fue solucionado, asegurando así que el producto final fuera viable y funcional.

Estructura de directorios

El proyecto se compone de la siguiente estructura de directorios:

- data (archivos de entrada y salida)
    - input (carpeta para cargar el comic)
        - Comic_test (carpeta donde se aloja el comic a estudiar en formato pdf. Se puede guardar cada capítulo por separado en carpetas)
    - output (carpetas de salida de cada proceso)
        - divide_images_test (imágenes derivadas de la identificación de viñetas)
        - divide_pages_test (imágenes derivadas separación por páginas del comic)
            - pages_raw (imágenes de cada página del comic)
            - pages_routes (archivo csv que aloja las rutas de las imágenes de la carpeta pages_raw)
        - ouput_audio_test (salida final de audio para cáda viñeta del comic)
        - output_description_test (archivos de texto donde se describe lo que aparece en cada viñeta)
        - output_description_translate_test (archivos de texto con la traducción de los archivos de descripción de cada viñeta)
        - output_text_test (archivos de texto donde se transcribe el texto que aparece en cada viñeta)
- documentation (documentación final del proyecto)
    - data (datos usados para la presentación y/o el paper)
    - paper.docx (paper del proyecto)
    - Presentation.pptx (presentación del proyecto)
- environments (entornos para cargar en anaconda)
    - SATURDAYS-2.yaml (entorno con todas las librerías necesarias para ejecutar el presente proyecto)
- execution_files (archivos de ejecución del proyecto en 2 formatos)
    - notebooks (notebooks de jupyter)
        - 1_divide_page.ipynb (divide las páginas del comic y las guarda como imágenes)
        - 2_identify_panels.ipynb (identifica cada viñeta y la guarda como imagen)

        - 3_bubbles_to_text.ipynb (identifica el texto que aparece en la viñeta y lo guarda en un archivo de texto)
        - 4_img_to_nlp.ipynb (identifica lo que ocurre en la imagen, hace una descripción en inglés y la guarda en un archivo de texto)
        - 5_translate.ipynb (traduce los archivos del punto 4 a español y los guarda en archivos de texto)
        - 6_tts.ipynb (narra los ficheros de descripción de la imagen y el texto que aparece en ella. Puntos 3 y 5)

    - py_files (archivos python para el pipeline)
        - 1_divide_page.py (divide las páginas del comic y las guarda como imágenes)
        - 2_identify_panels.py (identifica cada viñeta y la guarda como imagen)
        - 3_bubbles_to_text.py (identifica el texto que aparece en la viñeta y lo guarda en un archivo de texto)
        - 4_img_to_nlp.py (identifica lo que ocurre en la imagen, hace una descripción en inglés y la guarda en un archivo de texto)
        - 5_translate.py (traduce los archivos del punto 4 a español y los guarda en archivos de texto)
        - 6_tts.py (narra los ficheros de descripción de la imagen y el texto que aparece en ella. Puntos 3 y 5)
        - packages.py (bibliotecas necesarias)

- train (archivos para el entramiento del proceso de finetuning de los modelos de descripción de la imagen y texto en la imagen)

    - data (archivos de entrada y salida)
        - input (archivos de entrada en raw)

            - Comic_train (imágenes de cada página)
            - imagenes_prueba (imágenes aleatorias)

        - output (archivos procesados del comic)

            - divide_images_train (imágenes de cada viñeta de la páginas de input/Comic_train)

    - models (modelo finetuneados de trOCR)

        - modelo_finetuneado_trOCR
        - procesador_finetuneado_trOCR

    - results (modelo finetuneados de BLIP)
    - 1_identify_padels_for_finetuning.ipynb (proceso para dividir las páginas del training en viñetas)
    - 2_bubbles_to_text_finetuning.ipynb (proceso para realizar el finetuning de trOCR)
    - 3_img_to_nlp_finetuning.ipynb (proceso para realizar el finetuning de BLIP)

- .gitignore (gitignore)

- comic_AI.py (pipeline para la ejecución de los archivos de exeution_files/py_files en cadena)
- README.md (el presente archivo)

Ejecución del proceso completo

En este apartado se describe cómo ejecutar el proceso completo para pasar de un cómic en .pdf a los archivos finales de audio que describen cada viñeta:

1. Carga de ficheros

El proceso funciona desde la carga de un cómic en el directorio /data/input/Comic_test. Dentro de esta carpeta se puede cargar un fichero individual en .pdf o crear un subdirectorio de carpeta en el que cada carpeta sea un capítulo y dentro de esta cargar cada capítulo en formato .pdf.

En el caso de cargar un único fichero el sistema no es capaz de reconocer la división por capítulo para crear los subdirectorios oportunos.

2. Ejecución de la pipeline

La ejecución del proceso completo se hace ejecutando el archivo comic_AI.py del directorio raiz. Este no necesita la introducción de parámetros. Previo a este paso será necesario instalar todas las librerías necearias o cargar el entorno de anaconda que se encuentra en el directorio /environments. El terminal de ejecución debe ejecutarse en ese entorno proporcionado. También será necesario decargar los modelo de la siguiente ruta (https://drive.google.com/drive/folders/1oBLlCRzSQtV-mYpyaOOrT3YkxbWxSkVU?usp=drive_link) y cargarlos en el directorio /train

Será necesario abrir un terminal el del directorio raíz de ComicAI y ejecutar uno de estos 2 comandos:

python comic_AI.py

python3 comic_AI.py

3. Salida de resultados

El proceso se ejecutará al completo, imprimiendo el log perteneciente de cada módulo, y finalmente devolverá el tiempo de ejecución de todo el proceso y la frase: ENHORABUENA, YA SE PUEDE ESCUCHAR ESTE COMIC !!

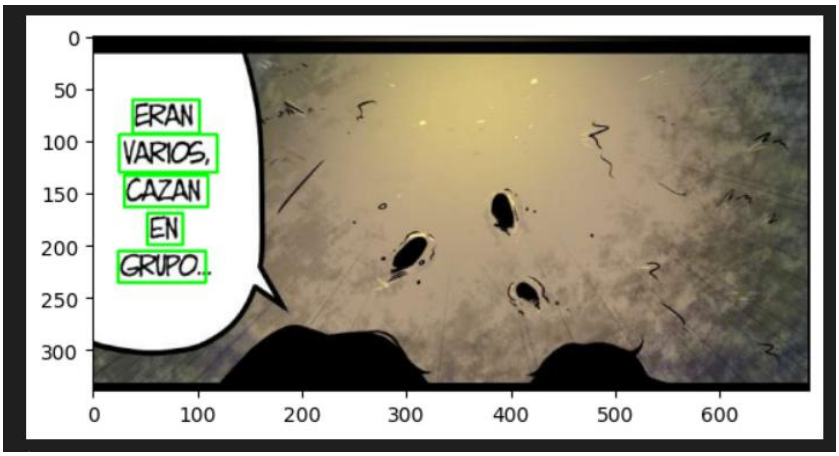
Los archivos de audio de cada viñeta se pueden encontrar en el directorio data/output/output_audio_test/combined.

Por otro lado en cada una de los siguientes directorios encontramos los archivos de salida de cada módulo o proceso:

- data/output
    - divide_images_test (imágenes derivadas de la identificación de viñetas)
    - divide_pages_test (imágenes derivadas separación por páginas del comic)
        - pages_raw (imágenes de cada página del comic)
        - pages_routes (archivo csv que aloja las rutas de las imágenes de la carpeta pages_raw)

- ouput_audio_test (salida final de audio para cáda viñeta del comic)
- output_description_test (archivos de texto donde se describe lo que aparece en cada viñeta)
- output_description_translate_test (archivos de texto con la traducción de los archivos de descripción de cada viñeta)
- output_text_test (archivos de texto donde se transcribe el texto que aparece en cada viñeta)

Ejemplos de ouput:



panel_1.mp3

panel_4.mp3

Tecnologías utilizadas

Las tecnologías utilizadas en este proyecto han sido diversas y especializadas, garantizando un flujo de trabajo eficiente y bien organizado. Entre ellas se incluyen:

- GitHub: Empleado para el control de versiones, facilitando la colaboración y el seguimiento de cambios en el código fuente.
- Visual Studio Code: Utilizado como editor de código, proporcionando un entorno de desarrollo robusto y flexible.
- Jupyter Notebooks: Implementados para depuración y procesamiento en Python, permitiendo una exploración interactiva de los datos y el código.
- Python y archivos .py: Usados para el procesamiento continuo en cascada, gestionando la ejecución secuencial de los distintos módulos del proyecto.
- Google Drive: Utilizado para almacenar los modelos finetuned, asegurando un acceso rápido y seguro a los archivos de gran tamaño.
- Anaconda: Empleado para la creación de entornos de desarrollo, encapsulando todas las dependencias en archivos .yaml para facilitar la reproducción del entorno de trabajo.

Esta combinación de herramientas y tecnologías ha permitido un desarrollo ágil y coordinado, asegurando la integridad y eficiencia del proyecto en cada etapa.

Evolución

El proyecto tiene un amplio margen de mejora y optimización, lo que permitirá su evolución continua y una mayor eficiencia. La estructura modular del proyecto asegura tanto su escalabilidad como la comodidad para el trabajo colaborativo. A continuación, se detallan algunas de las principales áreas de mejora:

1. Optimización de los modelos finetuned: Perfeccionar los modelos afinados para lograr resultados más precisos y fiables en el reconocimiento y procesamiento de los datos.

2. Mejora del reconocimiento del contorno de viñetas con OpenCV: Refinar los algoritmos de OpenCV para una detección más precisa de los contornos de las viñetas, mejorando así la calidad y exactitud del procesamiento de las imágenes.

3. Optimización del tono de voz de narración: Ajustar y perfeccionar el tono y la calidad de la voz narradora para una experiencia auditiva más natural y agradable.

4. Implementación del reconocimiento de diálogos por personajes: Desarrollar la capacidad de identificar y diferenciar los diálogos según los personajes, mejorando la coherencia y la fluidez de la narración.

5. Implementación del reconocimiento automático de capítulos: Automatizar la detección de capítulos dentro del contenido, facilitando la organización y navegación del material procesado.

6. Desarrollo de una interfaz gráfica: Crear una interfaz gráfica de usuario, ya sea móvil, web o de escritorio, que permita la carga de archivos de entrada y la visualización y escucha del resultado final de manera intuitiva y accesible.

7. Desarrollo de un módulo para el reconocimiento de páginas de cómic a través de fotografías: Implementar una funcionalidad que permita la entrada de páginas de cómic

mediante fotografías, complementando el punto anterior y mejorando la versatilidad del sistema.

8. Extrapolación a otros entornos artísticos: Ampliar las capacidades del proyecto a otros ámbitos, como la descripción de obras de arte, lo que abriría nuevas posibilidades y aplicaciones en el campo del arte y la cultura.

Estas mejoras no solo optimizarán el rendimiento del proyecto, sino que también ampliarán su alcance y funcionalidad, permitiendo su aplicación en diversos contextos y ofreciendo una experiencia más rica y completa para los usuarios.

Enlaces de interés

Enlace al proyecto completo en GitHub: https://github.com/JackDM/ComicAI