Miharu Scan Helper

Autor
Enrique Ávila Rodríguez

UNIVERSIDAD COMPLUTENSE
MADRID

Trabajo de fin de grado

Grado en Ingeniería Informática
Facultad de Informática

Dirigido por
Adrián Riesco Rodríguez

Madrid, 2020–2021

Resumen

El proceso de traducción de cómics, denominado *scanlation* (del inglés *scan* y *translation*), es complejo y conlleva la realización de diversas tareas con competencias disjuntas: la traducción del texto y efectos de sonido, el limpiado de los bocadillos para quitar el texto original, el redibujado de los efectos de sonido que suelen estar dibujados directamente sobre los paneles sin bocadillos, y la composición tipográfica del texto traducido. Este proyecto tiene como finalidad facilitar los procesos de traducción y composición tipográfica, ya que las tareas de limpieza y redibujado se pueden llevar a cabo de forma independiente y existen gran cantidad de herramientas de editado digital de imágenes para ello. Adicionalmente, la herramienta se ha desarrollado únicamente para su uso con *manga* (cómics de origen japonés).

La aplicación desarrollada es una herramienta gráfica que emplea un motor de OCR así como diversos mecanismos para utilizar servicios de traducción en línea y agrupar los resultados. Una vez traducido el *manga*, se puede exportar el resultado a varios formatos de texto plano. Estos contienen los números o nombres de página, la traducción y pueden incluir el texto original japonés. Los archivos generados tienen como finalidad su uso en herramientas de composición tipográfica.

Todo esto, junto con un sistema de anotaciones de las traducciones, ha resultado ser una herramienta relativamente popular para *scanlations* y traducciones independientes.

Palabras clave

Scanlation, traductores, manga, composición tipográfica, OCR, C#, .NET, web scraping

Abstract

The translation of comics, refered to as scanlation (from *scan* and *translation*), is a complex process for which a variety of non overlapping skills are required: translating the text and sound effects, cleaning bubbles to remove original text, redrawing sound effects which are usually drawn straight over the panels with no bubbles, and typesetting the translated text. This project aims to help only in the processes of translating and typesetting, as the cleaning and redrawing tasks can be carried out independently and there are already a multitude of digital image manipulation tools to do so. Additionally, the tool has only been developed for use with *manga* (Japanese comics).

The developed application is a graphical tool which makes use of an OCR engine as well as various mechanisms to fetch online translation services and group the results. Once the *manga* has been translated, the translation can be exported to a variety of plain text formats which contain page numbers or names, translations, and may include the original Japanese text. The generated files are meant to be used with typesetting tools.

Alongside a system to add notes to translations, the tool has become relatively popular for *scanlations* and independent translations.

Keywords

Scanlation, translator tools, manga, typesetting, OCR, C#, .NET, web scraping

Índice general

Capítulo 1

Introducción

Scanlation, del inglés *scan* y *translation*, es el proceso de la traducción de cómics. En este documento nos vamos a referir en particular a los comics de origen japonés, denominados *manga*, y su traducción al inglés, pero este proceso se puede llevar a cabo sobre cualquier cómic independientemente del idioma de origen o al que se pretende traducir.

En la figura 1.1 se muestra el proceso de las scanlations, compuesto por dos líneas de trabajo que se juntan antes de la composición tipográfica. Por un lado se realiza la limpieza de las *raws* (imágenes originales) para eliminar el texto original así como los *SFX* (efectos de sonido, onomatopeyas normalmente escritas sobre el dibujo) y se redibujan aquellos paneles en los que el texto estuviera directamente superpuesto sobre el dibujo. Al mismo tiempo se puede realizar la traducción del texto original y *SFX*. Una vez están estos tres procesos terminados se procede a la composición tipográfica del texto traducido sobre las imágenes editadas y finalmente se suele realizar un control de calidad para evitar erratas y otros errores que se pueden producir durante la edición de las imágenes y la composición tipográfica.

De estos procesos, los de limpieza y redibujado se llevan a cabo con herramientas de edición digital de imágenes como Photoshop, Krita o Gimp que ya tienen sus propios sistemas de automatización ([1, 2, 3], respectivamente). Por ello el ámbito de este proyecto no abarca dichas tareas.

El idioma japonés tiene un sistema de escritura que utiliza tres alfabetos distintos, dos de ellos (hiragana y katakana) son fonogramas silábicos relativamente simples de aprender. Sin embargo el tercer alfabeto (kanji) es un sistema de caracteres logográficos importado del alfabeto chino durante el siglo 4 D.C. Desde el año 2010 existen en la lengua japonesa 2.136 kanji (jōyō kanji, literalmente kanji de uso regular [4, 5]) considerados el mínimo conocimiento necesario para saber leer y escribir en japonés, y es la cantidad que se enseña durante la educación obligatoria. Sin embargo, muchos más kanji son utilizados, especialmente en literatura, por lo que resulta normal para un adulto conocer alrededor de 4.000 kanji e incluso más de 8.000 en el mundo académico literario. Esto, sumado a que en las imágenes digitales de los *mangas* el texto no es seleccionable, supone un gran obstáculo para la traducción si se tiene un nivel bajo de japonés.

Durante el proceso de traducción es común utilizar diversos servicios de traducción (como Google Translate, DeepL, Bing Translator o Yandex Translate) para comparar los resultados y obtener una idea general del significado de una frase. Esto implica para cada

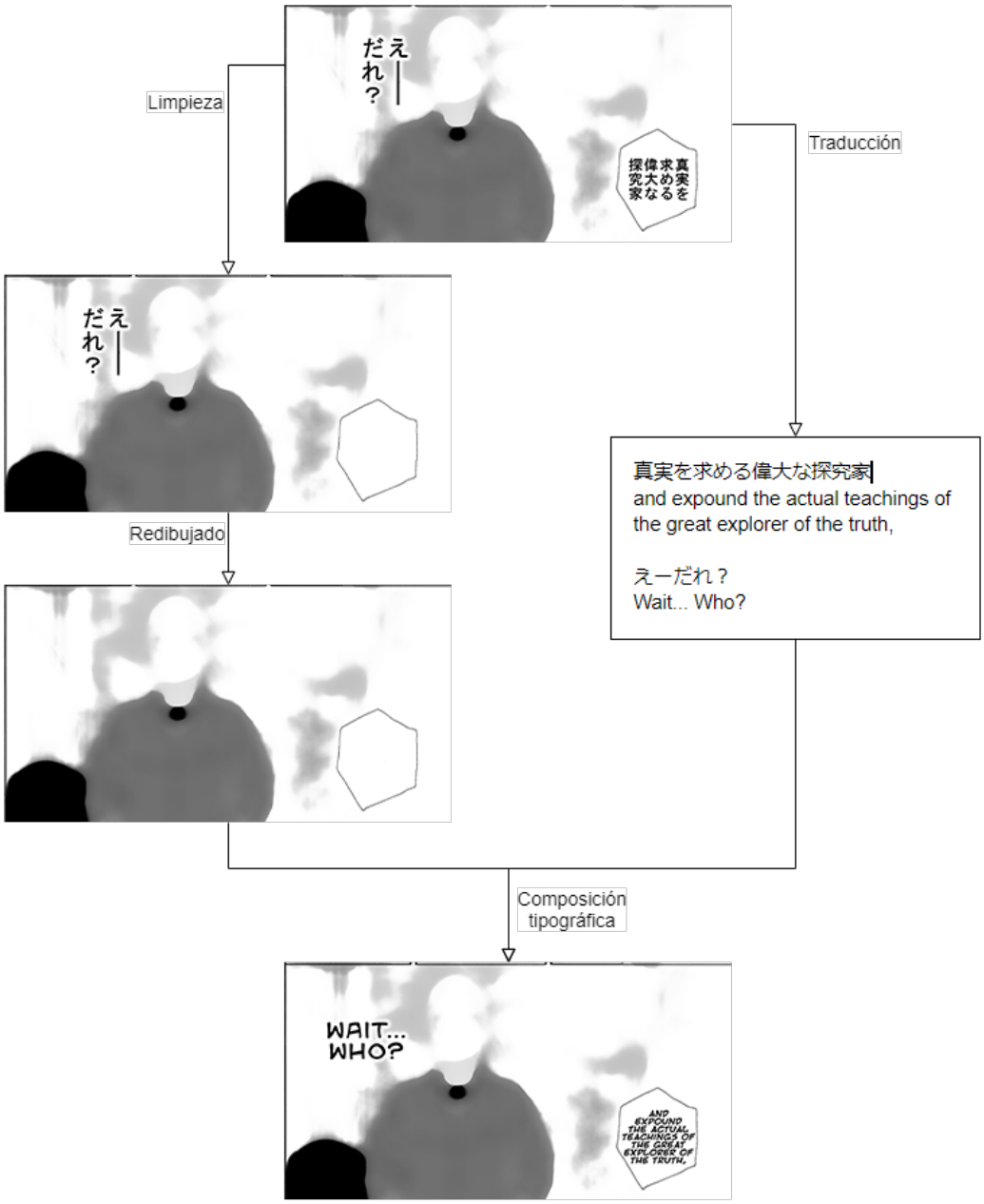

Figura 1.1: Proceso de las scanlations.

frase nueva tener que introducir el texto en cada uno de los servicios usados, además de tener que cambiar de un servicio a otro constantemente, lo que supone una gran pérdida de tiempo.

El proceso de composición tipográfica consiste en sobreponer digitalmente el texto traducido sobre las imágenes limpias. Esta tarea se lleva a cabo con herramientas de edición digital de imágenes y presenta dos complicaciones:

- Una de ellas es la introducción del texto en la herramienta, que normalmente se realiza copiando el texto desde un editor de texto y pegándolo en la herramienta. Este proceso se puede agilizar gracias a la existencia de extensiones en las herramientas de edición digital de imágenes, capaces de leer archivos de texto e introducir el

texto sin necesidad de salir de la aplicación, pero es necesario que el texto siga un formato estándar.

- La otra complicación es saber con exactitud dónde se debe colocar cada frase del texto sobre la imagen.

El objetivo de este trabajo es el desarrollo de una herramienta, que se ha denominado Miharu, para agilizar y facilitar los procesos de traducción y composición tipográfica de las scanlations.

Motivación

La literatura y el dibujo japoneses tienen una estructura y un estilo muy distintos al arte occidental al que estamos acostumbrados y están imbuidos de la cultura japonesa. Compartir el medio artístico del *manga*, que ha capturado la imaginación y los corazones de gran cantidad de personas a nivel mundial, es el principal motivador detrás de las scanlations. La principal motivación en el desarrollo de la herramienta es hacer este difícil proceso más accesible.

El estado del arte hoy día consiste en herramientas privativas utilizadas exclusivamente por los grupos que las han desarrollado y conjuntos de herramientas gratuitas y de pago, que de forma individual solo cubren algunas de las necesidades de nuestros usuarios. Esto supone el uso de varias herramientas a la vez, lo que implica una pérdida de tiempo. Poner a disposición del público una aplicación libre y gratuita para realizar scanlations fue también uno de los principales motivos para el desarrollo de Miharu.

Objetivos

El objetivo general de la aplicación es agilizar y automatizar los procesos de traducción y composición tipográfica.

En el proceso de traducción existen dos obstáculos: la complejidad del lenguaje japonés escrito y la traducción en sí. Por ello, es necesario que la herramienta sea capaz de reconocer los caracteres de forma automática en las imágenes digitales y agrupar los servicios de traducción, presentando los resultados de forma simple y compacta.

En el proceso de composición tipográfica también existen dos obstáculos que la herramienta pretende evitar: la introducción del texto de forma rápida y saber en qué parte de la imagen se debe introducir la parte correspondiente del texto. Miharu debe ser capaz de exportar el texto traducido a un formato estandarizado para extensiones de composición tipográfica y debe presentar la posición del texto sobre la imagen de forma inequívoca.

En resumen, la aplicación permitirá:

- Mediante OCR, extraer texto en japonés de una imágen.
- Acceder a servicios de traducción online y extraer los resultados.

- Permitir establecer la traducción final al usuario.
- Producir archivos de texto plano con las traducciones para uso en herramientas externas.
- Mostrar qué zonas de una imagen corresponden con cada parte del texto traducido.
- Todo ello se debe presentar de forma simple y compacta al usuario.

Uso y feedback recibido durante el desarrollo

Desde un primer momento se ha pretendido crear una aplicación de usuario. Por ello ha sido muy importante durante el desarrollo promover su uso para recibir *feedback* tanto de usabilidad como de errores. Estas comunicaciones se han llevado a cabo de forma personal a través de la plataforma de comunicación *Discord*, con miembros de comunidades relacionadas con la traducción de *manga*, y también se ha recibido este tipo de *feedback* a través de las *issues* de GitHub.

Sobre Miharu Scan Helper

Miharu Scan Helper es una herramienta de software libre con licencia MIT.

El código se puede encontrar en https://github.com/Ynscription/MiharuScanHelper .

También hay disponibles ejecutables portables (solo para Windows) en https://github.com/Ynscription/MiharuScanHelper/releases.

Organización de la memoria

Este documento presenta los siguientes capítulos:

- **Tecnologías utilizadas**: Información detallada sobre las tecnologías que la herramienta utiliza para su funcionamiento.
- **Implementación**: Una visión general de la arquitectura e implementación de la herramienta.
- **Herramienta**: Explicación detallada del funcionamiento de la herramienta y su uso.
- **Conclusiones**: Este capítulo presenta los resultados obtenidos y plantea trabajo que se puede realizar en el futuro para mejorar la aplicación.

Introduction

Scanlation, from *scan* and *translation*, is the process of translating comics. This document is particularly centered on japanese comics, called *manga*, and their translation to English, but this process can refer to any comic, independantly of original language or language to be translated to.

The scanlation process, as shown on figure 1.2, is composed of two lines of work that converge before typesetting. One of them consists of the editing (cleaning) of the raw files to remove the original text as well as SFX (Sound Effects, onomatopoeia often overlayed over the drawing) after which, those panels where text was directly over the drawing must be redrawn. The other one is the translation of the original text and SFX and can be carried out in parallel. Once these three processes are done, the translated text and SFX can be typeset over the edited images. Usually, at the end of the process, there is a quality control step to avoid typos and other errors that could arise during image editing and typesetting.

Out of these processes, cleaning and redrawing are carried out with digital image manipulation tools such as Photoshop, Krita, or Gimp that already have their own automation systems ([1, 2, 3] respectively). Thus these tasks are outside the scope of this project.

The Japanese language has a writing system that uses three distinct alphabets, two of them (hiragana and katakana) are relatively simple to learn syllabic phonograms. However, the third alphabet (kanji) is a logographic character system imported from the Chinese alphabet during the 4th century A.D. Since the year 2010 the minimum required knowledge to be considered literate in Japanese, taught throughout compulsory education, is 2,136 kanji (jōyō kanji, literally commonly used kanji) [4, 5]. However, there are many kanji in use not included in this set, specially in literature, making it rather common for the average adult to know around 4,000 kanji and even 8,000 in the literature academic world. This, added to the fact that the text on the digital images of *manga* is not selectable, becomes a significant hurdle for someone with a low level of Japanese when translating.

During the translation process, it is common to use various translation services (such as Google Translate, DeepL, Bing Translator or Yandex Translate) to compare their results in order to get a general idea of the meaning of a sentence. This entails that for each new sentence that must be translated, the text must be introduced into all the used services, and having to frequently change from one service to another, which results in a significant time loss.

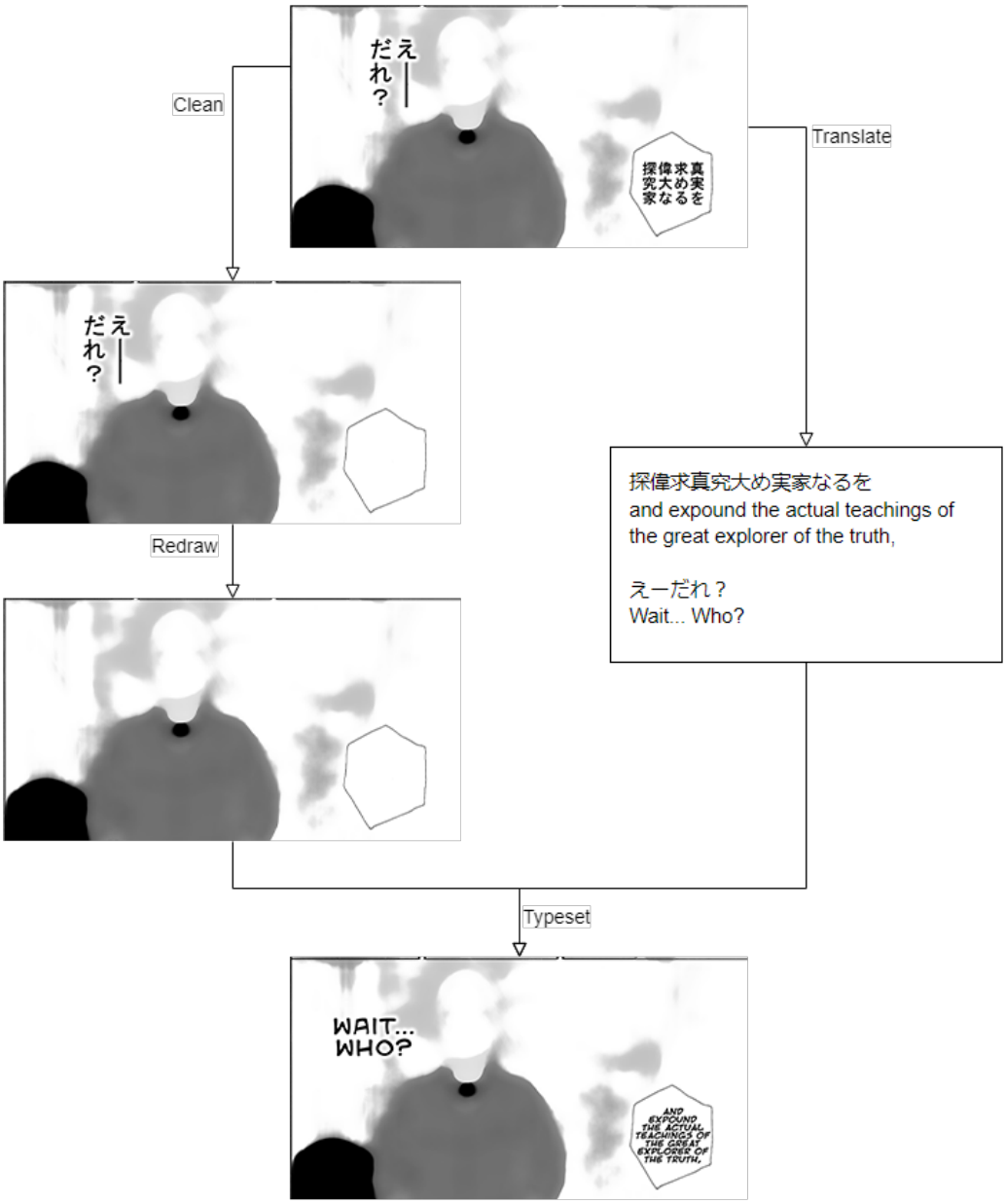

Figura 1.2: Scanlation process.

Typesetting is the process whereby the translated text is superimposed digitally over the cleaned images. This task is carried out with digital image manipulation tools and presents two obstacles.

- One of them is getting the text into the tool, usually done by copying it from a text editor and pasting it in the tool. This process can be sped up through extensions of the image manipulation tool that can read text from files and insert the text without having to exit the program, but for this the text must follow a standard format.
- The other problem is knowing exactly which part of the text must be placed on which part of the image.

The objective of this project is the development of a tool, which has been named Miharu, for the purpose of speeding up and aid in the processes of translation and typesetting of scanlations.

Motivation

Japanese literature and drawing have a very distinct structure and style from the occidental art than we are used to and Japanese culture permeates throughout the works. Sharing this artistic medium that has captured the imagination and hearts of so many around the world, is one of the main drives behind scanlations.
One of the main drives in the development of the tool is to make this difficult process more accessible.

The state of the art nowadays is comprised of privative tools used exclusively by the groups that develop them and sets of free and paid tools each one of them only covering part of the needs of our users. This results in the use of multiple tools at the same which, in turn, causes a loss of time. Releasing a free tool (as in freedom and as in free beer) to make scanlations was another main drive in the development of Miharu.

Objectives

The general objective of the application is to speed up and automate the translation and typesetting processes of scanlations.

The translation process presents two obstacles: the complexity of the language and the translation proper. For this reason, the tool must be able to automatically recognize the characters on digital images and aggregate the translation services, displaying the results in a simple and compact manner.

There are also two obstacles in the process of typesetting that Miharu aims to overcome: inserting text in a speedy manner and knowing which part of the text must go on which part of the image. Miharu must be able to export the translated text to a standardized format for the use on typesetting extensions and must clearly show the position of text on the image.

In summary, the application will allow to:

- Using OCR, extract Japanese text from an image.
- Access and retrieve the results of online translation services.
- Allow the user to set the final translation to be used.
- Generate plain text files with the translations for use with external tools.
- Show which part of the translated text must go on which part of the image.
- All of this must be displayed neatly and in a simple way.

Usage and Feedback Received during Development

From the very beginning Miharu was set to be a user application. For this reason, promoting and sharing the use of the tool has been important during development in order to receive feedback, both of usability and bugs. This communications have happened mostly privately through the communication platform Discord, with members of communities related to the translation of *manga*. GitHub issues have also been a source of this kind of feedback, but to a lesser extent.

About Miharu Scan Helper

Miharu Scan Helper is a free and open source tool under the MIT license.

The source code can be found at https://github.com/Ynscription/MiharuScanHelper

There are also portable binaries (for Windows only) at
https://github.com/Ynscription/MiharuScanHelper/releases

Contents of this Document

This document includes the following chapters:

- **Used Technologies**: Detailed information about the technologies that the tool uses to run.
- **Implementation**: An overview of the tool's architecture and implementation.
- **Tool**: Detailed guide on the use of the tool.
- **Conclusions**: This chapter exposes the obtained results and future work that can improve the application.

Capítulo 2

Tecnologías utilizadas

El lenguaje de programación elegido para el desarrollo del proyecto fue C# [6, 7], un lenguaje de alto nivel similar a Java, más moderno y flexible. C# es un lenguaje de programación orientado a objetos fuertemente tipado para el desarrollo de aplicaciones sobre el ecosistema de .NET, explicado a continuación.

Se ha elegido este lenguaje ya que el objetivo del proyecto es desarrollar una aplicación de usuario en un tiempo limitado y un lenguaje de alto nivel permite un desarrollo más ágil que uno de bajo nivel. Como alternativa se podría haber utilizado Java, pero teniendo más experiencia en C# el desarrollo de la aplicación se realizaría de forma más rápida y eficaz.

A continuación se detallan las principales tecnologías que utiliza Miharu.

.NET Framework

.NET Framework [8] es el entorno desarrollado por Microsoft que provee principalmente la máquina virtual, también referida como motor de ejecución, llamada CLR (Common Language Runtime) y una biblioteca de clases (.NET Framework Class Library) que provee código testado y reutilizable para los programadores. El motor de ejecución CLR interpreta un código intermedio llamado CIL (Common Intermediate Language) mediante compilación *just-in-time*. C# no es el único lenguaje cuyo código se puede ejecutar sobre el entorno de .NET: Microsoft desarrolla y soporta también Visual Basic y F# [9]; asimismo, también existen otros lenguajes compilables al lenguaje intermedio CIL [10] y, por tanto, ejecutables sobre el entorno de .NET. Miharu utiliza la versión 4.8 de .NET Framework.

Al comienzo del desarrollo de la aplicación, la única alternativa a .NET Framework para ejecutar programas escritos en C# era Mono [11], una reimplementación del .NET Framework de Microsoft, de código libre y multiplataforma. Mono surgió con el propósito de ejecutar aplicaciones desarrolladas para .NET Framework sobre GNU/Linux, pero no es una implementación completa y carece de ciertas *features*, entre otras WPF y directivas async, ambas utilizadas por Miharu. Debido a esto y a la complejidad que añadiría el desarrollo multiplataforma se decidió desarrollar la aplicación únicamente para Win-

dows. Teniendo esto en cuenta, el uso de Mono no aportaba ningún beneficio para el desarrollo.

Una alternativa actual sería el reciente .NET 5 [12], una reimplementación del .NET Framework, de código libre y multiplataforma desarrollada por la Fundación .NET (.NET Foundation) principalmente llevada a cabo por Microsoft y Mono. .NET 5 no es solo una reimplementación de .NET Framework, sino que ha pasado a ser la implementación principal de .NET.

Windows Presentation Foundation (WPF)

Para el desarrollo de la interfaz gráfica de usuario se decidió el uso de WPF (Windows Presentation Foundation) [13]. WPF es un subsistema gráfico de código libre y forma parte del entorno de .NET. WPF utiliza XML para el diseño de la interfaz y se vincula con la lógica del programa mediante eventos, interacciones de usuario y ligaduras de datos, accesibles a través de .NET Framework.

El principal motivo por el que se decidió usar WPF es por ser la biblioteca para el desarrollo de interfaces gráficas propia de .NET (al momento del comienzo del desarrollo). Esto implicaba un sistema robusto, estable y soportado por Microsoft con amplia documentación y un gran ecosistema de desarrolladores.

Una alternativa habría sido Windows Forms [14], la biblioteca de interfaces de usuario de .NET previa, que no se usó por su antigüedad, no solo aparente en la API de la biblioteca sino también en la apariencia de la interfaz gráfica resultante. Otra alternativa popular a WPF es Avalonia [15], una biblioteca de código libre, multiplataforma e independiente de Microsoft, para el desarrollo de interfaces de usuario en .NET. Se descartó el uso de Avalonia al tener experiencia previa en Windows Forms, que tiene similitudes con WPF (al ser su predecesor), lo que supondría un mejor uso de WPF.

Tesseract OCR

Tesseract [16] es una aplicación de código libre que proporciona un motor de reconocimiento visual de caracteres (OCR, Optical Character Recognition), que permite, dada una imagen, extraer el texto que haya en ella. Tesseract utiliza redes neuronales para el reconocimiento de caracteres y utiliza diferentes datos de entrenamiento para cada idioma. Dada la complejidad del sistema de escritura japonés, es una parte clave del proyecto.

En lugar de utilizar Tesseract mediante una biblioteca y API, debido la existencia de problemas a la hora de enlazar la biblioteca con .NET, Miharu realiza llamadas al programa de línea de comandos de Tesseract. Adicionalmente, Miharu utiliza dos conjuntos de datos de entrenamiento no incluídos por defecto en Tesseract. Esto se debe a que los datos de entrenamiento para el reconocimiento del japonés incluidos en Tesseract fueron recolectados con japonés escrito en horizontal de izquierda a derecha, mientras que normalmente el japonés se escribe en vertical de derecha a izquierda. Esto provoca gran cantidad de

errores en el OCR y por ello Miharu incluye dos conjuntos de datos de entrenamiento, uno para el japonés escrito en vertical y otro para el escrito en horizontal.

Tesseract es una de las pocas aplicaciones de OCR gratuitas y de código libre. Existen alternativas, pero la mayoría como ABBYY FineReader [17] son privativas y de pago. Otras, como OCR.Space [18], son herramientas en línea que exponen una API y ofrecen uso gratuito limitado. Para Miharu era necesario que la herramienta fuera gratuita, preferiblemente de código libre, y sin limitaciones, por lo que Tesseract era el único candidato viable.

Selenium

Selenim [19] es un entorno que permite controlar navegadores web mediante código. Esto se realiza mediante comandos o llamadas a la API de un controlador que se comunica con el navegador y devuelve los resultados.

Selenium fue desarrollado como una herramienta para la automatización de tests en el desarrollo de páginas y aplicaciones web. Sin embargo, sus usos se extienden mucho más allá y Miharu utiliza Selenium para acceder a diversos servicios de traducción a través de la web que no proveen una API gratuita o la API que proveen es limitada.

Existen otros sistemas similares a Selenium, como Cypress [20] una herramienta freemium más centrada en el ámbito de testeo, lo que supondría mayores dificultades de implementación para las necesidades de Miharu. La única alternativa real para el uso con Miharu sería Karate [21], un framework de testeo de código libre, gratuito y con pocas dependencias creado precisamente como alternativa a Selenium . No se eligió este *framework* porque requiere el uso un lenguaje específico de dominio (DSL, Domain Specific Language) para llevar a cabo las tareas que Miharu necesita. Selenium, mientras que es un *framework* antiguo y difícil de manejar en ciertos casos, ofrece *bindings* para .NET que permiten el desarrollo de Miharu con una única base de código, mientras que Karate requiere su uso a través de comandos del sistema operativo y archivos de código fuente escritos en el lenguaje específico de la aplicación.

KozakuraDB

KozakuraDB [22] es una base de datos SQLite que recopila cierta información de los archivos *radkfile* y *kanjidic2* [23]. Estos archivos son diccionarios del idioma japonés creados por Electronic Dictionary Research and Development Group. KozakuraDB fue creada para su uso en este proyecto (más información sobre el funcionamiento interno en el Capítulo 3) y contiene tablas que relacionan los kanji del idioma japonés con sus radicales, un conjunto de símbolos comunes que se combinan para formar los kanji. Esto permite buscar para un conjunto determinado de radicales todos los kanji que los contienen y es un sistema comunmente utilizado por diccionarios japoneses (físicos como el de Kodansha [24] y digitales como Jisho [25]) y sistemas de escritura electrónicos [26].

La decisión de crear un sistema propio para acceder a esta información surge de la inexistencia de sistemas semejantes. La única alternativa son los propios archivos *radkfile* y

kanjidic2. Estos ficheros son archivos de texto plano que incluyen gran cantidad de información (la mayoría irrelevante para las necesidades de Miharu) y resultan difíciles de navegar para cumplir con las necesidades de nuestra herramienta. Por ello se extrajo la información relevante de dichos archivos y se compiló un archivo SQLite que permite extraer la información necesaria de forma mucho más sencilla y rápida.

SQLite

SQLite [27] es un motor de base de datos SQL autocontenida, sin servidor, de cero-configuración y transaccional. En lugar de utilizar un sistema cliente-servidor para la manipulación de la base de datos, los datos se gestionan directamente sobre ficheros y el sistema que los maneja queda embebido en la aplicación final. Esto permite la manipulación de datos con la facilidad del uso de archivos de texto plano pero con la potencia de las consultas sobre bases de datos. Miharu utiliza SQLite únicamente para hacer consultas sobre la base de datos KozakuraDB.

No existen alternativas reales a SQLite, ya que todos los sistemas de bases de datos alternativos se basan en el sistema cliente-servidor, que para el uso tradicional de una base de datos arbitrariamente accesible por una cantidad arbitraria de clientes es la mejor estrategia, pero para el uso que hace Miharu, levantar un servidor para acceder a los datos necesarios conlleva un coste de eficiencia y recursos intolerable. Habría sido posible implementar directamente un sistema de lectura de ficheros, pero se perdería la capacidad que ofrece SQLite de ejecutar consultas SQL sobre los datos en el fichero.

Capítulo 3

Implementación

Miharu, al ser una herramienta gráfica destinada al usuario medio, se ha desarrollado con una arquitectura centrada en el patrón Modelo-Vista-Controlador. El código está separado en tres *namespaces* principales: *BackEnd*, *FrontEnd* y *Control* que contienen el Modelo, la Vista y el Controlador, respectivamente. Adicionalmente se ha dado prioridad a la capacidad de respuesta (*responsiveness*) de la aplicación, por lo que se usan modelos de programación concurrente en múltiples sistemas y tareas que de otra forma bloquearían el programa de cara al usuario.

MVC - Modelo

El flujo de la aplicación se basa en la edición de un capítulo, que contiene un conjunto de páginas (siendo cada página una imagen) y cada página, a su vez, contiene un conjunto de bloques de texto (también llamados en código "entradas de texto" en aquellos sitios donde el nombre bloque de texto pudiera ser ambiguo). El modelo contiene las estructuras de datos necesarias para representar estos elementos, así como la lógica subyacente para su manipulación y el resto de funcionalidad necesaria para la ejecución del programa.

Estructuras de datos

Los bloques de texto contienen un rectángulo que representa el área de la imagen (página) en píxeles en la que se encuentran. También contienen el texto en japonés contenido en dicha área, el texto obtenido mediante traducciones web, la traducción manual establecida por el usuario y una lista de anotaciones de usuario.

Las páginas contienen la ruta de fichero de la imagen que representan, así como la lista de bloques de texto contenidas en la página.

Un capítulo contiene la lista de páginas que lo forman, así como la ruta del fichero donde se guardan (serializan) todos los datos del capítulo.

Todos estos datos son serializables a un archivo en formato JSON. Todas las rutas de fichero incluidas en el archivo JSON son relativas al propio fichero, lo que permite compartir el fichero de forma sencilla. El sistema de serialización se detalla a continuación.

Funcionalidad

Las tareas de traducción son complejas y requieren tiempo, por lo que es clave la capacidad de guardar el trabajo realizado y poder recuperarlo entre sesiones de trabajo. Por ello se ha implementado la capacidad de serializar y deserializar (guardar y cargar) un capítulo en formato JSON. Adicionalmente, durante el desarrollo de la aplicación, al añadir nuevas características al programa, los archivos de guardado quedaban obsoletos haciendo que la aplicación no pudiera recuperar la información. Por ello se implementó un sistema de versiones de archivos de guardado capaz de identificar la versión de un archivo al intentar abrirlo y de actualizarlo a la versión actual en el caso de que el archivo fuera de una versión anterior. Adicionalmente, en caso de fallo irrecuperable de la aplicación, se ejecuta un sistema que intenta guardar el trabajo actual de forma automática, lo que ha permitido en muchos casos evitar la pérdida de datos.

Para obtener los mejores resultados al ejecutar la extracción de texto mediante Tesseract es necesario proveer a Tesseract con una imagen en la que preferiblemente solo haya texto. Para ello se recorta el área de la página en la que se encuentra el texto y sobre el resultado de esta operación se ejecuta el OCR.

Durante el desarrollo de la aplicación, para poder identificar y corregir los errores encontrados por usuarios se añadió un sistema de registro de configuraciones y errores. Este sistema crea un archivo de registro en cada sesión donde se escriben los datos de depuración de cualquier excepción encontrada durante la ejecución. En caso de fallo irrecuperable de la aplicación, se crea un registro independiente que permite identificar la fuente de los errores de esta naturaleza.

En el modelo también se encuentra el sistema de acceso a la base de datos KozakuraDB (más información en el capítulo 2). Este acceso es de solo lectura y proporciona a la vista los datos necesarios para permitir al usuario introducir kanji mediante el sistema de búsqueda por radicales (también detallado en el capítulo 2). KozakuraDB fue creada para el uso con esta aplicación y es una base de datos relacional compuesta por tres tablas:

- **Kanji** que contiene los kanji. Incluye el código UTF-8, el carácter y el número de trazos de cada kanji.
- **Rad** que contiene los radicales. Incluye el código UTF-8, el carácter más representativo (algunos radicales no tienen carácter UTF-8 propio), el número de trazos y el índice en el que aparece en el fichero RADKFILE cada radical.
- **KanjiByRad** que contiene entradas para cada par kanji-radical en los que el radical está presente en el kanji.

Esto permite realizar consultas que, dado un conjunto de radicales, devuelven un conjunto, posiblemente vacío, de kanji que los contienen.

Finalmente, el modelo también incluye el sistema de traducciones, que se explica más adelante en los sistemas concurrentes.

MVC - Controlador

En Miharu los controladores actúan como interfaz entre el modelo y la vista y mantienen el estado de la aplicación. Los controladores están separados por el ámbito del que son responsables:

- Entradas de texto.
- Páginas.
- Capítulos.
- Traducción.
- Entrada de Kanji por radicales.

Los distintos controladores tienen referencias entre ellos para poder informar de cambios en el estado de la aplicación provocados en un controlador que afecten al estado de otros controladores.

Cada controlador tiene referencias a los datos de su ámbito y pasan llamadas a estos desde la vista añadiendo parámetros del estado de la aplicación donde sea necesario.

Los controladores también exponen eventos para la actualización de la vista. Este sistema se explica en la siguiente sección.

MVC - Vista

En WPF existen tres elementos visuales principales: ventanas, páginas y controles. Las ventanas se corresponden con el concepto tradicional de ventana de un programa visual y los controles son componentes visuales que se distribuyen mediante *layouts* y contenedores. Miharu utiliza únicamente ventanas y controles ya que las páginas de WPF proporcionan funcionalidad principalmente para el desarrollo de aplicaciones web.

La vista está separada en tres partes generales:

- Ventana principal, contiene la barra de herramientas y los dos controles siguientes.
- Vista de página, un control en el que se visualiza la página actual y permite la selección de bloques de texto.
- Vista de entrada de texto (bloque de texto), un control que muestra y permite modificar la entrada de texto actual.

Además, existen ventanas adicionales que añaden funcionalidad fuera del ámbito de las partes anteriores:

- Ajustes, que permiten modificar el comportamiento de ciertas partes de la aplicación.
- Edición de capítulo, permite añadir, eliminar y alterar el orden de las páginas del capítulo actual.

- Acerca de (About), es una ventana que contiene la licencia de Miharu (y link al repositorio de GitHub). También contiene las licencias del software y los datos de terceros utilizados por la aplicación, así como, las fuentes de las traducciones web.

Interacciones

Cada elemento visual de la Vista tiene referencias a los controladores de aquellos datos que el elemento represente o manipule. Cambios provocados por el usuario a través de la interfaz se realizan con llamadas al controlador del sistema apropiado.

Miharu hace uso del sistema de eventos de C# para las actualizaciones de los componentes visuales provocadas en otras partes del código. De esta forma, en lugar de utilizar el patrón *observer*, cada elemento visual se suscribe a los eventos necesarios del controlador apropiado y responde a ellos. Esto hace que la responsabilidad de actualización de la vista quede casi completamente encapsulada en la propia vista.

Adicionalmente, algunos elementos utilizan el sistema de ligaduras de datos (*data bindings*) de WPF que permite vincular un elemento visual directamente con una estructura de datos. Esto resulta en actualizaciones de la vista ante cambios en dicha estructura sin necesidad de código adicional.

La figura 3.1 muestra un diagrama de las interacciones entre el modelo, controlador y vista.

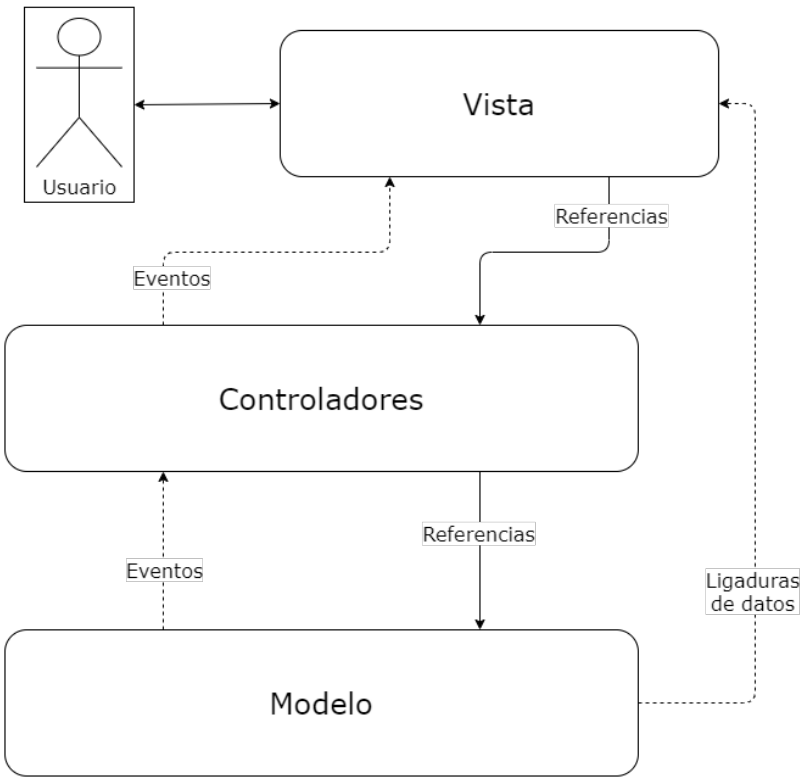

Figura 3.1: Diagrama de interacciones MVC.

Sistemas concurrentes

Existen tres hilos de la aplicación: el hilo principal que se convierte en el hilo de la interfaz gráfica tras la inicialización de la aplicación, el hilo del sistema de traducción y el hilo agente de Selenium. La figura 3.2 sirve como guía para la explicación del funcionamiento e interacciones de estos hilos, que se presenta a continuación.

El hilo del sistema traducción se crea durante la inicialización de la aplicación y se mantiene vivo hasta la finalización del programa. Este hilo contiene una cola de peticiones de traducción (que contienen el texto a traducir y la fuente de la que se debe traducir) y se mantiene dormido a la espera de una señal. Desde el hilo principal, a través del controlador de traducciones se realizan peticiones que se añaden a la cola del hilo de traducción y se activa la señal para indicar que hay peticiones sin procesar. Al despertar, el hilo de traducciones comprueba la cola de peticiones y para cada petición se ejecuta en paralelo, con el sistema de tareas de C#, el código de traducción apropiado. Las traducciones de cada fuente se ejecutan de forma similar, realizando una petición HTTP o accediendo a la web apropiada mediante Selenium y procesando el resultado para obtener el texto traducido. Las traducciones obtenidas (o los datos de error en caso de fallo) se proveen al controlador de traducciones mediante funciones *callback*.

El hilo agente de Selenium, al igual que el hilo de traducción, se crea durante la inicialización de Miharu y solo se termina al finalizar la aplicación. Este hilo se encarga de mantener el driver de Selenium activo y de realizar las operaciones requeridas por las tareas de traducción que requieren simulación web. Esto conlleva navegar a la dirección web apropiada, introducir el texto a traducir y recuperar el resultado a través del navegador *headless* de Selenium. Debido a las limitaciones de Selenium, este hilo tiene un cerrojo para que las tareas de traducción realicen sus operaciones de principio a fin de forma secuencial.

Durante la carga (deserialización) de un capítulo uno de los pasos consiste en cargar las imágenes de cada página a memoria. Estas imágenes son potencialmente enormes y dependiendo del número de páginas puede suponer una gran cantidad de tiempo. Por ello para agilizar el proceso, solamente se carga de forma secuencial la imagen de la primera página. La carga del resto de imágenes se delega a una tarea asíncrona de C# lo que permite que el programa continúe la ejecución mientras se carga el resto de páginas en segundo plano. Adicionalmente, se utilizan señales accesibles a través del controlador de páginas para prevenir que la vista intente mostrar una página que no haya sido cargada.

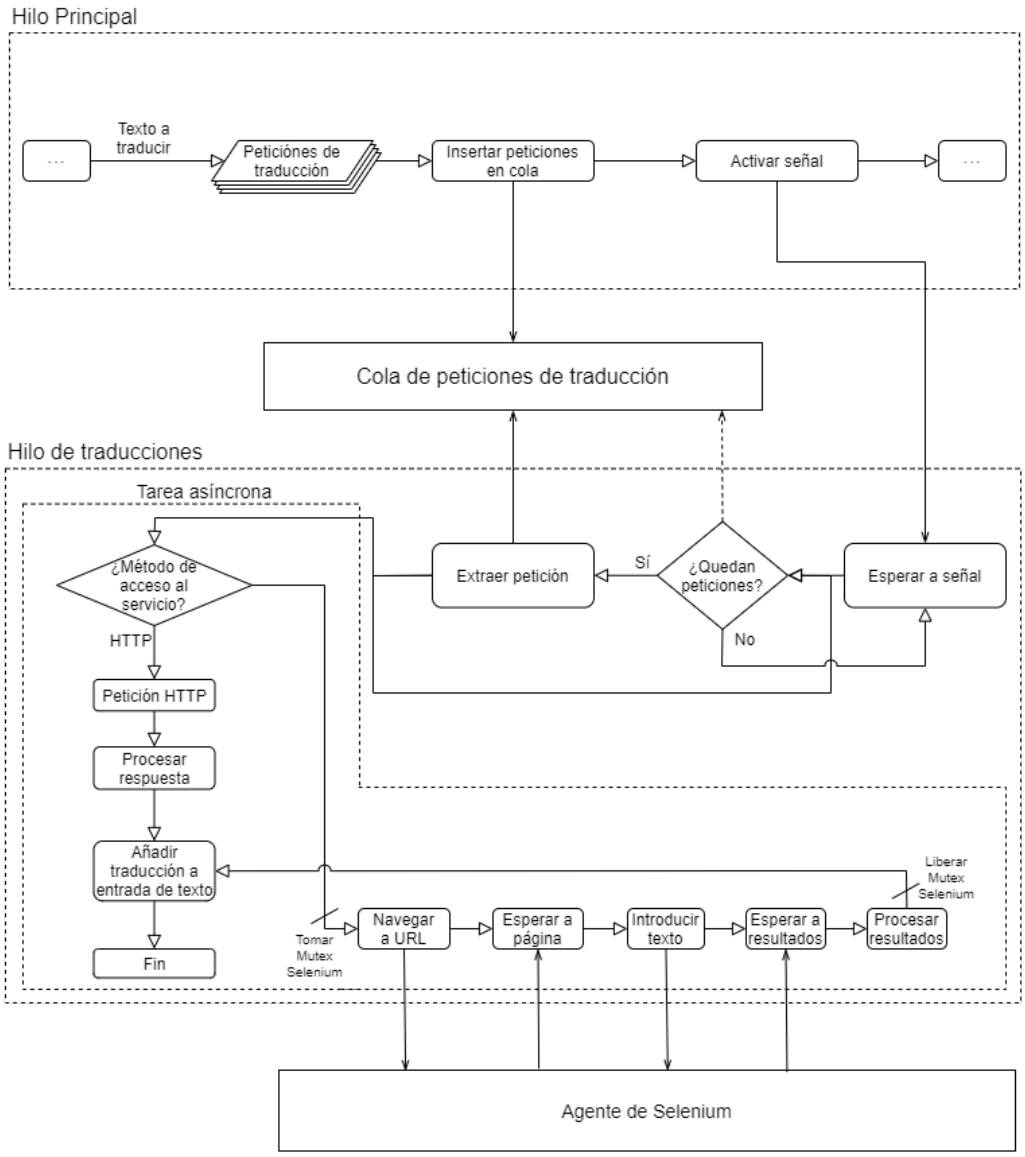

Figura 3.2: Diagrama de sistemas concurrentes de traducción.

Capítulo 4

Herramienta

En este capítulo se muestra el uso de la herramienta en detalle. La herramienta ha sido desarrollada exclusivamente para Windows y se puede obtener el ejecutable en https://github.com/Ynscription/MiharuScanHelper/releases.

Inicio

Al iniciar la herramienta invocando el ejecutable se muestra el espacio de trabajo vacío como se muestra en la figura 4.1. En este estado al no haber cargado ningún proyecto (denominados capítulos en la herramienta) las funciones principales están deshabilitadas. En este estado de la aplicación solamente se tendrá acceso a la Barra de menús, situada en la parte superior de la aplicación.

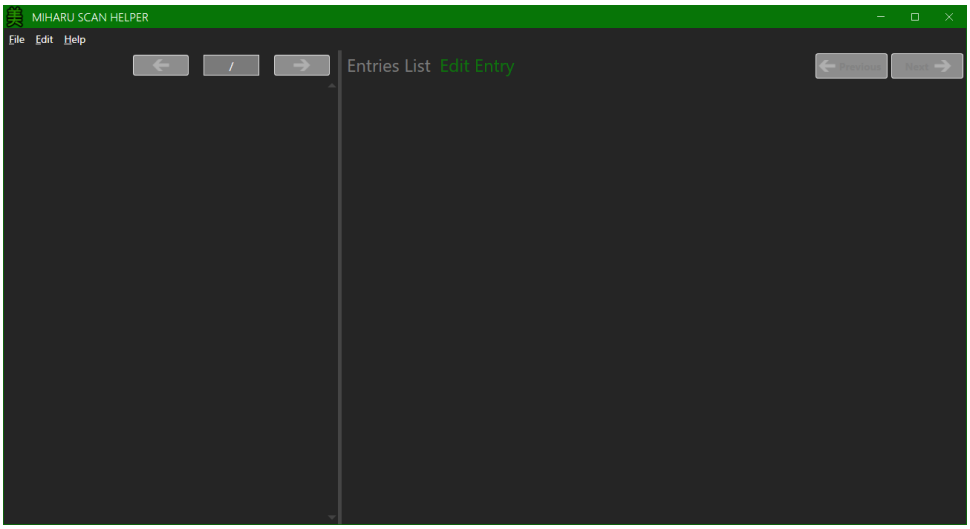

Figura 4.1: Miharu con el espacio de trabajo vacío.

Adicionalmente Miharu permite ser iniciado con un archivo como argumento. Si el archivo es un capítulo de Miharu (archivos con extensión *.scan*), la aplicación cargará el capítulo al abrirse. Esto permite abrir archivos *.scan* con la herramienta al hacer doble click, una vez Miharu haya sido establecido como el programa por defecto para abrir archivos de este tipo a través de Windows.

Barra de menús

La barra de menús contiene tres submenús principales:

- *File* (Archivo) que contiene todas las opciones relacionadas con la manipulación de ficheros. Adicionalmente permite terminar la ejecución de la herramienta.
- *Edit* (Editar) que contiene una opción para editar las páginas y su orden del capítulo activo (*Chapter Pages...*) y otra opción para acceder a la ventana de preferencias (*Preferences...*).
- *Help* (Ayuda) que contiene una única opción para acceder a la ventana que detalla información sobre la herramienta (*About...*).

Dentro del menú *File* se presentan cinco opciones relacionadas con la manipulación de capítulos (ver figura 4.2), una opción para generar los archivos de texto para herramientas de composición tipográfica (y otros usos) y una opción para salir de la herramienta:

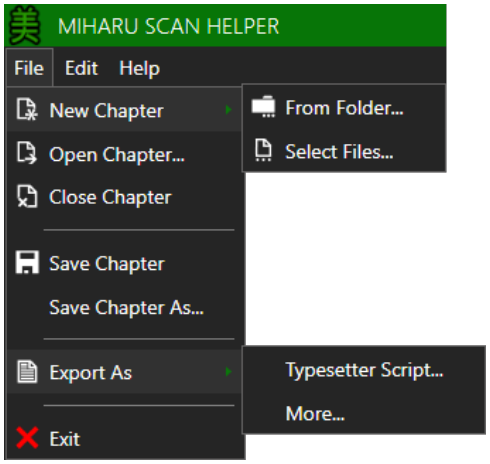

Figura 4.2: Barra de menús y submenú *File*.

- *New Chapter* es un submenú que contiene dos opciones que permiten crear un capítulo nuevo. El objetivo de ambas opciones es proporcionar a la herramienta un conjunto de imágenes que formarán las páginas del capítulo. La opción *From Folder...* abrirá un diálogo que nos permitirá seleccionar un directorio e importará todas las imágenes contenidas en dicho directorio como páginas de un nuevo capítulo. La otra opción *Select Files...* abre un diálogo que permite seleccionar uno o varios ficheros concretos de imagen, de utilidad si no se desea incluir todas las imágenes de un directorio en el capítulo.
- *Open Chapter...*. Esta opción abre un diálogo para seleccionar un archivo de capítulo (extensión *.scan*) lo que permite cargar un capítulo ya existente.
- *Close Chapter* cierra el capítulo activo, haciendo que la herramienta quede en un estado de trabajo vacío.
- *Save Chapter As...* abre un diálogo que permite guardar el capítulo activo en un fichero *.scan*.

- *Save Chapter* guarda el capítulo activo en el último fichero donde se haya guardado. Si no se ha guardado el capítulo nunca, se abre el mismo diálogo que si se hubiera seleccionado la opción *Save Chapter As...*
- *Export As* es un submenú con opciones para exportar el capítulo a archivos de texto plano. La primera opción *Typesetter Script...* permite exportar el capítulo a un fichero de texto con formato estandarizado para el plugin *Typesetter* de Photoshop. La opción *More...* abre un diálogo que permite personalizar el contenido del fichero de texto al que se va a exportar el capítulo.
- *Exit* termina la ejecución del programa.

Las opciones *New Chapter*, *Open Chapter...*, *Close Chapter* y *Exit* tienen como consecuencia la eliminación del capítulo activo (si hubiera uno) por lo que todas ellas avisan al usuario de cambios en el capítulo activo no guardados y permiten al usuario guardar el capítulo activo antes de continuar con la operación, continuar sin guardar o cancelar la operación.

Vista principal

La vista principal, como se puede observar en la figura 4.3, contiene dos áreas: a la izquierda la vista de página y a la derecha el área de las entradas de texto.

Figura 4.3: Vista principal de Miharu.

Vista de página

La vista de página (ver figura 4.4) contiene una visualización de la página sobre la que se está trabajando. Esta visualización permite desplazarse por la imagen mediante las barras de desplazamiento, aumentar o reducir el tamaño de la imagen (Ctrl + Rueda del ratón), seleccionar áreas de texto para crear nuevas entradas de texto (pulsando y arrastrando el botón secundario del ratón) y seleccionar entradas de texto ya existentes (pulsando el botón principal del ratón). Las áreas de texto se muestran como rectángulos sobre la imagen. El rectángulo correspondiente a la entrada de texto activa se muestra en azul mientras que el resto de rectángulos se muestran en rojo. Al pasar el cursor sobre un rectángulo se aumenta el grosor del borde, indicando el rectángulo que se va a seleccionar si se pulsa el botón principal del ratón.

Figura 4.4: Vista de página de Miharu.

En la parte superior de la vista de página se muestra el nombre del archivo que corresponde a la página actual, así como controles para cambiar la página actual. De estos controles, las flechas permiten acceder a las páginas anterior o siguiente de la página actual y en medio se muestra el índice de la página actual y el total de páginas. Haciendo click sobre el índice se puede escribir un nuevo índice y al pulsar la tecla *Enter* se navegará al nuevo índice.

Área de entradas de texto

El área de las entradas de texto contiene dos vistas disntintas, accesibles a través de las pestañas *Entries List* (Lista de entradas) y *Edit Entry* (Editar entrada). Adicionalmente, en la parte superior derecha, se encuentran dos botones que permiten seleccionar la entrada de texto anterior y siguiente a la actual.

Vista de lista de entradas de texto

La lista de entradas muestra todas las entradas de texto de la página en el orden en el que aparecerán al exportar. Las distintas entradas muestran una vista previa del área de la imagen donde se encuentran, el texto en japonés extraído mediante OCR (o editado por el usuario) y la traducción establecida por el usuario. Todo esto se puede observar en la figura 4.5.

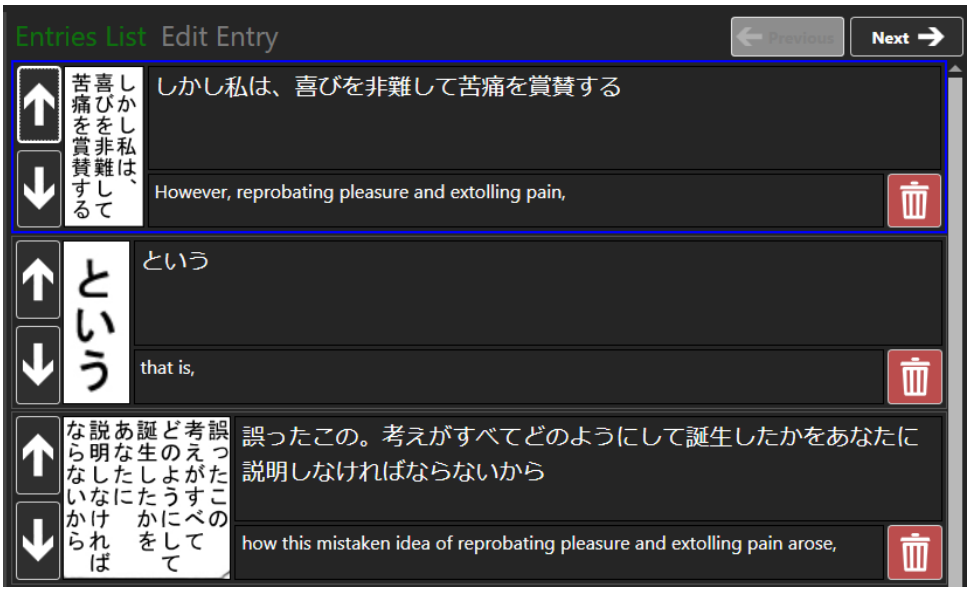

Figura 4.5: Lista de entradas de texto de Miharu.

Cada entrada incluye controles para modificar la posición de la lista en la que aparece y un botón para eliminar la entrada (por defecto se muestra una advertencia al pulsar este botón para evitar borrados accidentales). La entrada activa se resalta con un borde de color azul.

Vista de edición de entradas de texto

Esta vista muestra y permite editar todos los datos de la entrada de texto activa y está organizada verticalmente en tres secciones como se observa en la figura 4.6.

Figura 4.6: Edición de entradas de texto en Miharu.

La sección superior muestra todo lo relacionado con el texto en japonés de la entrada. En la parte izquierda se muestra el recorte de la imagen de la página que se utiliza para el OCR. En la parte superior un control permite cambiar la orientación del texto que espera el OCR (vertical u horizontal). En la parte superior derecha un botón que permite realizar el OCR sobre la imagen de nuevo (Miharu realiza el OCR automáticamente cuando se crea una nueva entrada de texto).

El resto de está sección contiene un cuadro de texto en el que se muestra el texto en japonés extraído mediante OCR, que puede ser editado por el usuario. Este cuadro tiene a la derecha un control que al pulsarlo muestra un desplegable que permite introducir kanji japoneses en el cuadro de texto mediante el método de selección por radicales (ver figura 4.7).

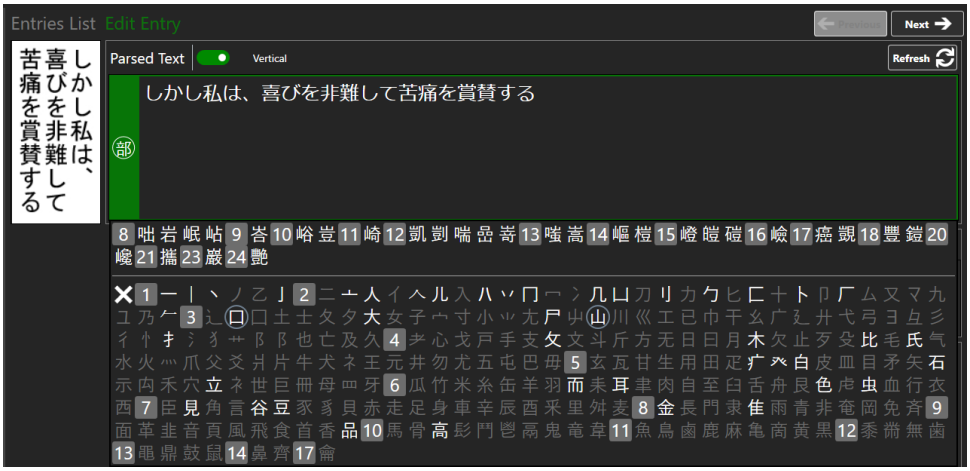

Figura 4.7: Sección superior con entrada de kanji por radicales desplegada.

La sección central, titulada *Translation* (traducción), contiene únicamente un cuadro de texto en el que el usuario puede introducir la traducción final de la entrada de texto (ver figura 4.8). El texto que se introduzca aquí será el que aparezca al exportar.

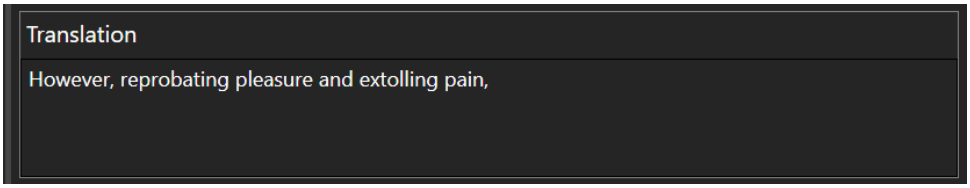

Figura 4.8: Traducción final del usuario.

La sección inferior contiene cuatro pestañas con herramientas de ayuda para la traducción y para el trabajo en equipo.

La pestaña *WebTL*, abreviatura de *Web Translation* (traducción web), agrupa varios servicios de traducción en línea como se muestra en la figura 4.9. Por defecto al crear una entrada de texto, el texto en japonés obtenido mediante OCR se manda a estos servicios de traducción y se presetan aquí los resultados automáticamente. Este comportamiento se puede desactivar en las preferencias, y también se pueden desactivar servicios de traducción de forma individual. En la parte superior un botón permite realizar todas las traducciones en línea de nuevo, útil cuando se cambia el texto japonés. Adicionalmente cada servicio tiene su propio botón para realizar la traducción de nuevo, utilizando únicamente ese servicio.

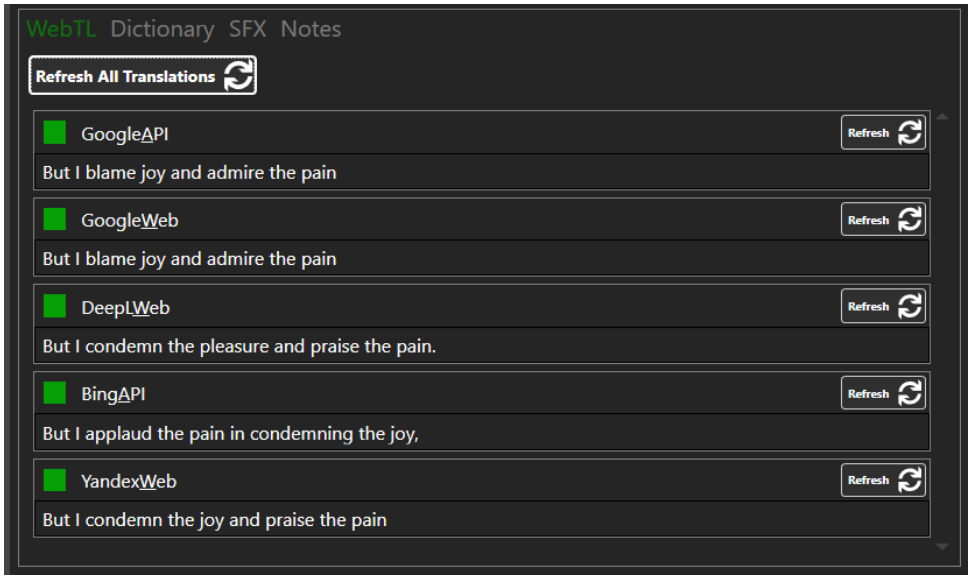

Figura 4.9: Agregador de herramientas de traducción en línea.

La pestaña *Dictionary* (diccionario) permite ver los resultados del diccionario en línea https://jisho.org/ para el texto japonés. La pestaña incluye en la parte superior el enlace web que lleva a la página de la que se ha obtenido la información así como un botón para realizar la búsqueda de nuevo. El resto de la pestaña muestra los resultados de la misma forma que se muestran en *Jisho*, permitiendo seleccionar las distintas palabras que forman la frase y mostrando a continuación sus entradas en el diccionario (ver figura 4.10).

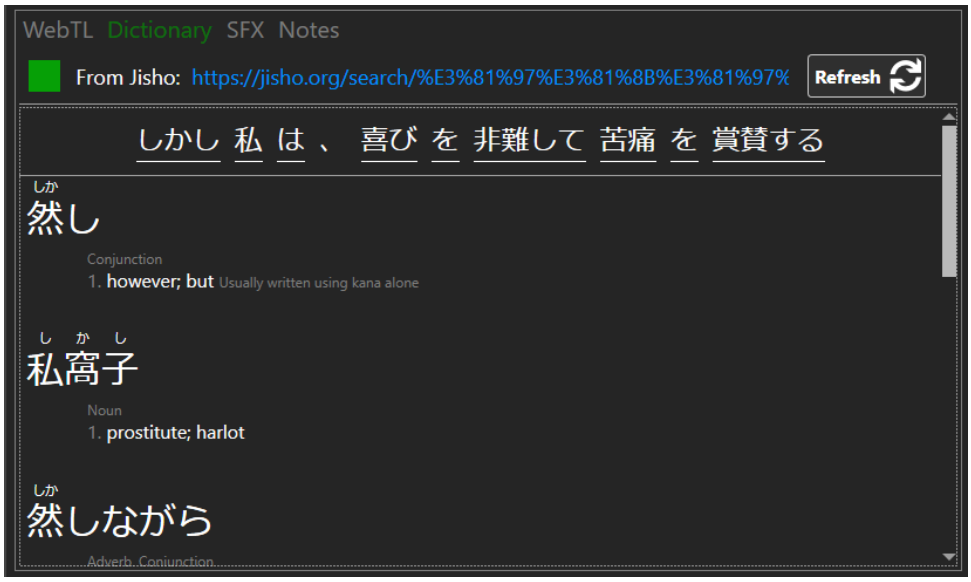

Figura 4.10: Resultados de búsqueda en https://jisho.org/.

La pestaña *SFX*, abreviatura de *Sound Effects* (efectos de sonido), utiliza el servicio de http://thejadednetwork.com/sfx para buscar posibles traducciones del texto japonés como onomatopeyas. En la parte superior derecha se encuentra un botón para realizar la búsqueda de nuevo. Debajo se muestran los resultados de la búsqueda de la misma forma que se muestran en el servicio original, separados en cuatro columnas: texto en japonés, texto en japonés romanizado (romaji), onomatopeyas equivalentes en inglés y explicaciones (ver figura 4.11).

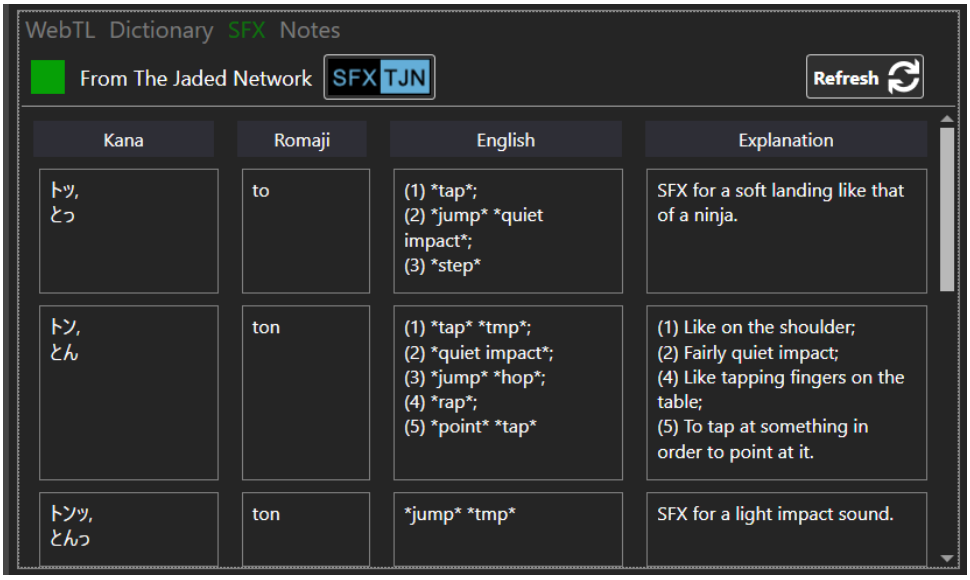

Figura 4.11: Resultados de búsqueda en thejadednetwork.com/sfx/.

Finalmente, la pestaña *Notes* (notas) contiene cuadros de texto en los que el usuario puede escribir notas o comentarios sobre la entrada de texto, la traducción, etc. (ver figura 4.12). Esto ayuda a la hora de compartir los archivos *.scan* al trabajar en equipo. Al introducir texto en el último cuadro vacío, un nuevo cuadro aparece permitiendo añadir un número arbitrario de anotaciones ordenadas.

Figura 4.12: Anotaciones de una entrada de texto.

Diálogo de exportar

Este diálogo, accesible desde las opciones de la barra de menús *File* → *Export As* → *More...*, permite personalizar el contenido del archivo de texto al que se va a exportar el capítulo. Como se observa en la figura 4.13, en la parte superior se muestra un control estándar para establecer la ruta del archivo al que se va a exportar.

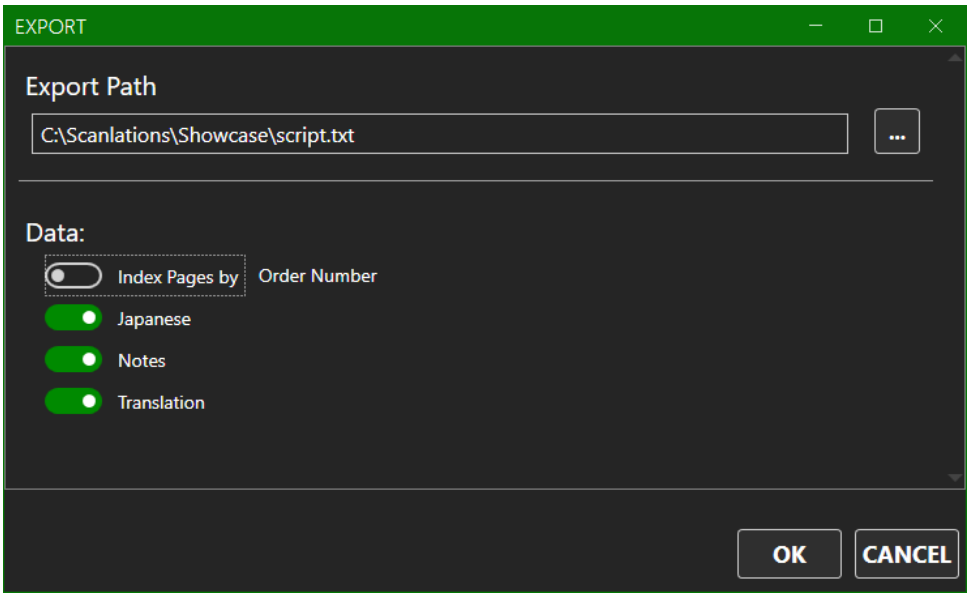

Figura 4.13: Diálogo de exportar.

De las cuatro opciones que aparecen a continuación, la primera, *Index Pages By* (indexar páginas por), permite cambiar la forma en la que se indexan las páginas: bien por el orden en el que aparecen (*Order Number*) o bien por el nombre del archivo de imagen (*File Name*). El resto de opciones están relacionadas con el texto en japonés (*Japanese*), las anotaciones (*Notes*) y la traducción final (*Translation*); y permiten controlar si se incluirán los datos correspondientes en el archivo de texto.

Diálogo para editar páginas

Este diálogo, al que se accede desde las opciones de la barra de menús *Edit* → *Chapter Pages...*, muestra la lista de páginas que forman el capítulo. Mediante los controles de la parte inferior del diálogo, se puede alterar el orden de las páginas del capítulo (ordenadas alfabeticamente por defecto), así como añadir y eliminar páginas (ver figura 4.14).

Al intentar eliminar una página, si esta contiene entradas de texto se muestra un diálogo estándar para confirmar la eliminación. Esto previene la eliminación accidental de datos.

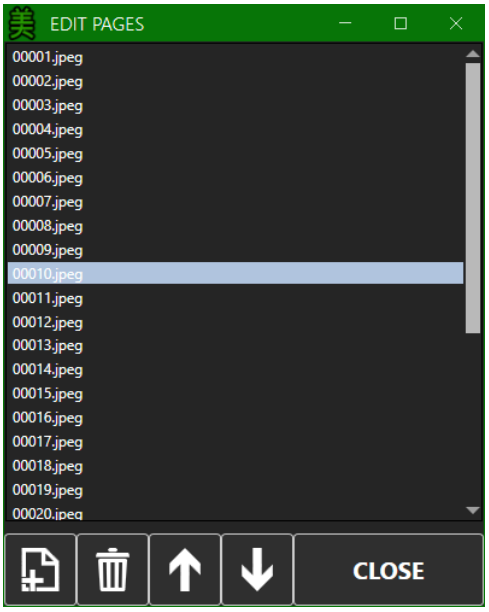

Figura 4.14: Diálogo para editar páginas del capítulo.

Diálogo de preferencias

Este diálogo, accesible a través de las opciones de la barra de menús *Edit* → *Preferences...*, permite al usuario modificar de forma persistente las opciones generales de la herramienta. Como se observa en la figura 4.15, las opciones están organizadas verticalmente en cinco secciones:

- *Tesseract Path* (Ruta de Tesseract), esta sección contiene un control estándar para determinar la ruta al archivo ejecutable de Tesseract. Por defecto Miharu redistribuye una copia de Tesseract (licencia Apache 2.0) preconfigurada para el uso con Miharu, pero permite al usuario usar su propia copia con su propia configuración.
- *Theme* (Tema), en esta sección se presentan controles para modificar la apariencia de Miharu. En concreto, permite alternar entre un color base claro u oscuro, así como alterar el color de acento de la aplicación.
- *Use Screen DPI* (usar DPI de la pantalla). Esta opción permite modificar la forma en la que Miharu interpreta las imágenes. Debido a inconsistencias en distintas configuraciones hardware, es posible que Miharu muestre los rectángulos de las entradas de texto de forma errónea, en cuyo caso activar está opción solventará el problema.
- *Warn on text entry deletion* (avisar al eliminar entradas de texto), esta opción permite desactivar los diálogos de confirmación al eliminar entradas de texto (ver sección Vista de Lista de Entradas de Texto).
- *Auto Translate New Text Entries* (traducir automáticamente nuevas entradas de texto). Esta sección presenta un interruptor global para desactivar el uso automático de los sistemas de traducción en línea al crear una nueva entrada de texto. Si se desactiva este control, se deberán ejecutar los servicios de traducción en línea de forma manual. Adicionalmente aparecen controles para cada uno de los servicios de

traducción en línea accesibles desde Miharu, que permiten desactivar servicios de traducción de forma concreta.

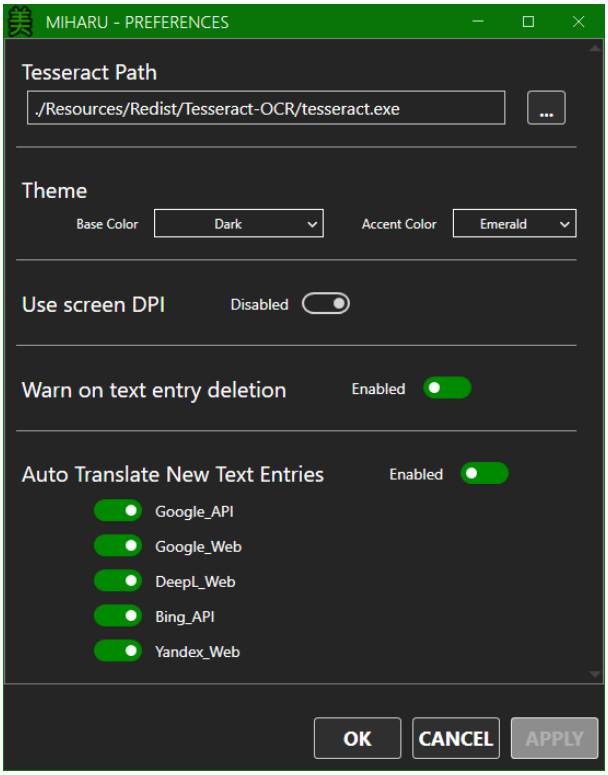

Figura 4.15: Diálogo de preferencias.

Diálogo "Acerca de"

Este diálogo, visible en la figura 4.16 muestra información sobre la aplicación:

- *Licenses* (licencias) muestra la licencia de Miharu, así como las licencias de todas las bibliotecas utilizadas por la aplicación. Adicionalmente muestra las licencias del software redistribuido junto a Miharu.
- *Translation Sources* (fuentes de traducción) muestra enlaces a todos los servicios de traducción accesibles a través de Miharu.
- *Kanji/Radical Dictionary* (Diccionario de kanji/radicales) esta sección muestra información sobre el uso de los archivos de datos proporcionados por *Electronic Dictionary Research and Development Group* de acuerdo a su licencia.

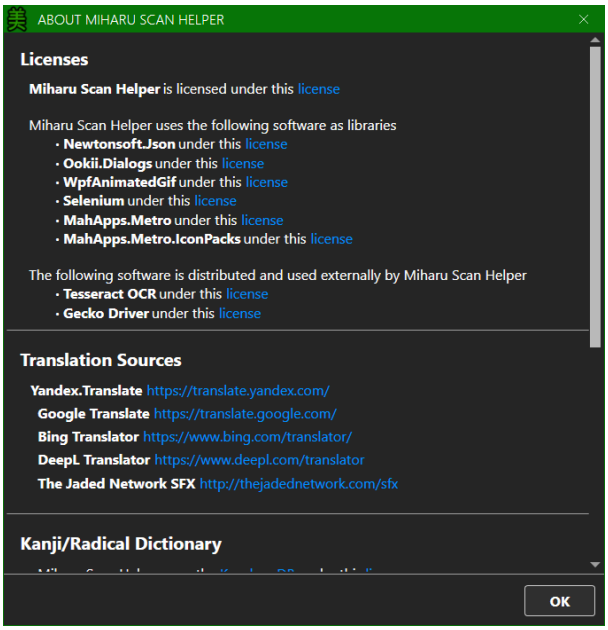

Figura 4.16: Diálogo “Acerca de”.

Capítulo 5

Conclusiones y trabajo futuro

Uno de los principales objetivos de la aplicación era agilizar y proporcionar una ayuda a la traducción de *mangas* japoneses, proporcionando al usuario herramientas visuales y automatizadas que presentaran la información de forma simple y compacta, evitando así la pérdida de tiempo que surge al utilizar múltiples herramientas distintas al mismo tiempo. Este objetivo se ha alcanzado de forma satisfactoria, incluso más allá de lo esperado. La única deficiencia en este aspecto ha sido la limitada capacidad de manejar el sistema de OCR al utilizarlo de forma externa en lugar de a través de una biblioteca.

En cuanto al segundo objetivo principal de la aplicación, proporcionar sistemas para el uso de los datos generados por la aplicación en sistemas de automatización para la composición tipográfica, así como una guía visual para llevar a cabo la composición tipográfica: no solo se ha cumplido sino que se ha llevado más allá, demostrando ser un sistema útil para otros usos además de la composición tipográfica, permitiendo a usuarios crear archivos de texto plano fáciles de comprender y compartir.

En general, se puede decir que Miharu ha resultado ser una herramienta relativamente exitosa. Como se puede observar en la figura 5.1, en un periodo de 30 días, se tradujeron un total aproximado de 21.770 caracteres. Es difícil estimar el número de usuarios debido a los múltiples usos de la aplicación y a la posibilidad de la existencia de usuarios que no utilicen el sistema de traducción de Bing. Pero desde el punto de vista de la traducción de *manga*, teniendo en cuenta que un bocadillo suele contener una media de 15 caracteres, una página suele contener una media de 7 bocadillos y un capítulo una media de 20 páginas, 21.770 caracteres suponen alrededor de 10 capítulos de *manga* traducidos en un mes. Esto sumado al *feedback* recibido durante el desarrollo (detallado en el Capítulo 1) implica un uso relativamente grande en el pequeño círculo de las *scanlations*.

En el ámbito personal, este proyecto ha supuesto la adquisición de conocimiento y experiencia en el uso de herramientas modernas para el desarrollo de interfaces de usuario como WPF así como en métodos de parseo de HTML y web scraping (extracción de datos web).

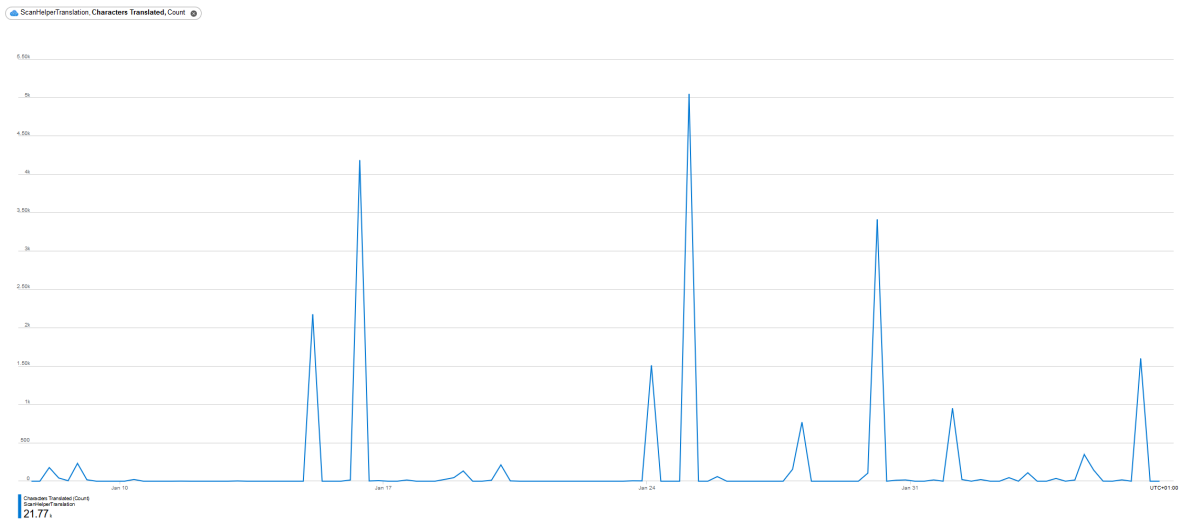

Figura 5.1: Estadísticas de uso del servicio de traducción de Bing (Caracteres traducidos en 30 días).

Trabajo futuro

Aunque Miharu es una herramienta completamente funcional, existen aún características (*features*) adicionales cuya implementación mejoraría la aplicación y permitiría un uso más extendido.

OCR

El uso actual de Tesseract de forma externa permite poca manejabilidad de los resultados. La implementación del uso de una biblioteca permitiría mostrar los resultados de formas más efectiva, así como proporcionar posibles resultados alternativos, lo que mejoraría en gran medida el servicio.

Idiomas

Actualmente Miharu solo soporta la traducción de *mangas* japoneses. Más usuarios podrían beneficiarse de la herramienta si se pudiera utilizar para cómics escritos en otros idiomas, como se puede desprender de la "*issue*" 26 del repositorio de GitHub (https://github.com/Ynscription/MiharuScanHelper/issues/26) en la que un usuario de GitHub propone que añadir la capacidad de utilizar Miharu para la traducción de cualquier idioma mejoraría la aplicación.

Multiplataforma

Miharu es una herramienta desarrollada únicamente para Windows. Con el lanzamiento de .NET5 uno de los principales objetivos para el futuro es portar la herramienta a dicho entorno, permitiendo así la compatibilidad multiplataforma. A través de comunicaciones

personales, se sabe que existen personas interesadas en la herramienta que utilizan MacOS o GNU/Linux y que actualmente no pueden utilizar Miharu.

Scripts de usuarios

Para el análisis sintáctico de HTML y web scraping (extracción de datos web), Miharu utiliza algoritmos específicos para cada servicio y solamente incluye algunos servicios. Una de las extensiones más ambiciosas que añadir al proyecto sería la capacidad de utilizar scripts proporcionados por usuarios para la obtención de datos de traducción de servicios arbitrarios. Esto se podría realizar creando una API que permitiera interactuar con los sistemas de traducción de Miharu a través de un lenguaje de scripting como Lua [28].

Conclusions and Future Work

One of the main objectives of the application was to speed up and help in the translation of Japanese *manga*, through visual and automated tools that display the information neatly and in a simple way, thus avoiding the loss of time incurred by the use of multiple tools simultaneously. This objective has been successfully reached, even beyond expectations. The only shortcoming in this aspect was the limitation in driving the OCR engine by using it externally instead of using a library.

In terms of the second objective, the capability to use the data generated through the application in automated systems for typesetting and to provide a visual guide during the typesetting process: not only has it been reached, it has been taken further than anticipated, as it has proven to be a useful system for other uses beyond typesetting, allowing users to create plain text files that are easy to read and share.

In general, it can be said that Miharu has proven to be a relatively successful tool. As seen on figure 5.2, in a 30 day period, a total of 21,770 characters were translated. It is not easy to assess the amount of users from this data due to the multiple uses of the application and the possibility of using it without the Bing translation service. However, from the *manga* translation perspective, given that a text bubble usually contains an average of 15 characters, with pages containing an average of 7 bubbles and a chapter being 20 pages long on average, 21,770 translated characters mean around 10 *manga* chapters being translated in a month. Knowing this and taking into account the feedback received during development (detailed in Chapter 1), it can be said that Miharu is relatively widely used in the small community of scanlations.

Personally, this project has entailed acquiring knowledge and experience in the use of modern tools for graphical user interface development as WPF as well as in mechanisms for parsing HTML and web scraping.

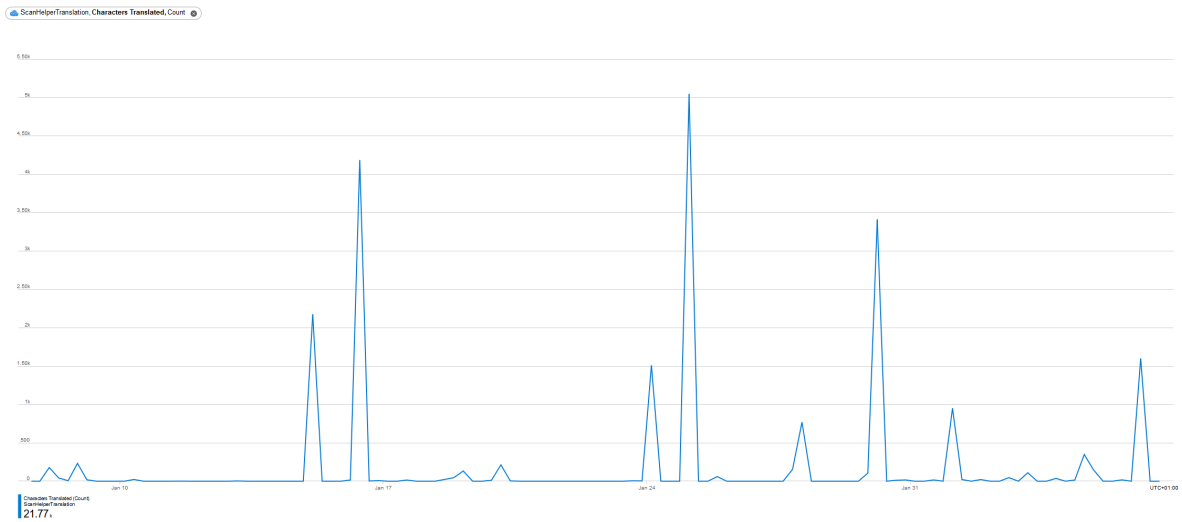

Figura 5.2: Usage stats of Bing translation service (Translated characters in 30 days).

Future Work

Even though Miharu is a completely functional tool, there are still features that would improve the application and allow to reach more users if implemented.

OCR

Using Tesseract externally gives little control over the obtained results. Implementing it's use through a proper library would allow for a better display of the results as well as the capability to provide alternative results, improving the system greatly.

Languages

Currently, Miharu only supports the translation of Japanese *manga*. More people would be able to use the tool if it could be used on comics written on other languages, as is apparent from the GitHub issue 26 (https://github.com/Ynscription/MiharuScanHelper/issues/26) where a GitHub user proposes that the capability to use Miharu for translation from any language would improve the application.

Cross Platform

Miharu is a tool only developed for use on Windows. With the recent release of .NET 5 one of the main objectives for the future would be to port the tool to said framework, which would allow for cross platform compatibility. There are people interested on Miharu that use MacOS or GNU/Linux that can't currently use the application and have contacted me about this.

User Scripts

For the syntactical analysis of HTML and web scraping, Miharu uses algorithms specific to each service and only for some services. One of the most ambitious extensions to the project would be to add user scripting to allow for the collection of data from arbitrary online translation services. This could be achieved by providing an API to interact with the translation systems of Miharu through a scripting language such as Lua [28].

Índice de figuras

Bibliografía

[1] Adobe Photoshop. *Photoshop User Guide*. Automation of tasks and actions in Photoshop. URL: https://helpx.adobe.com/photoshop/user-guide.html?topic=/photoshop/morehelp/automation.ug.js.

[2] Krita Foundation. *Krita 4.4 Manual*. Python Scripting. URL: https://docs.krita.org/en/user_manual/python_scripting.html.

[3] The GIMP Documentation Team. *GNU Image Manipulation Program User Manual*. The GIMP Team. Chapter 13, Scripting. URL: https://docs.gimp.org/en/gimp-scripting.html.

[4] Ministerio de educación de Japón. *Tabla de kanji de uso común (常用漢字表, japonés romanizado: jōyō kanji hyō)*, 2010.

[5] Wikimedia Foundation Inc. Jōyō kanji. URL: https://en.wikipedia.org/wiki/J%C5%8Dy%C5%8D_kanji.

[6] Microsoft. *C# programming guide*. URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/.

[7] Andrew Troelsen and Phil Japikse. *Pro C# 7: With .NET and .NET Core*. Apress, 2017.

[8] Microsoft. *.NET Framework documentation*. What is .NET Framework? URL: https://docs.microsoft.com/en-us/dotnet/framework/get-started/#what-is-net-framework.

[9] Microsoft. *.NET Programming Languages*. URL: https://dotnet.microsoft.com/languages.

[10] Wikimedia Foundation Inc. List of cli languages. URL: https://en.wikipedia.org/wiki/List_of_CLI_languages.

[11] .Net Foundation. *Mono Documentation*. About Mono. URL: https://www.mono-project.com/docs/about-mono/.

[12] Richard Lander. Announcing .net 5.0. 2020. URL: https://devblogs.microsoft.com/dotnet/announcing-net-5-0/.

[13] Microsoft. *Windows Presentation Foundation (WPF) documentation*. URL: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/.

[14] Microsoft. *Windows Forms documentation*. URL: https://docs.microsoft.com/en-us/dotnet/desktop/winforms.

[15] AvaloniaUI OÜ. *Avalonia Docs*. URL: http://avaloniaui.net/docs.

[16] Google. *Tesseract OCR*. README. URL: https://github.com/tesseract-ocr/tesseract/blob/master/README.md.

[17] ABBYY. *AI-powered OCR SDK for Windows, Linux Mac OS - ABBYY OCR API*. URL: https://www.abbyy.com/ocr-sdk/.

[18] a9t9 software GmbH. *Free OCR API*. URL: http://ocr.space/OCRAPI.

[19] Software Freedom Conservancy. *The Selenium Browser Automation Project*. URL: https://www.selenium.dev/documentation/en/.

[20] Cypress.io. *Cypress Documentation*. URL: https://docs.cypress.io/guides/overview/why-cypress.html.

[21] Intuit Inc. *Test Automation Made Simple*. README. URL: https://github.com/intuit/karate/blob/master/README.md.

[22] Enrique Ávila Rodríguez. Kozakura db - a kanji by radicals sqlite database. README. URL: https://github.com/Ynscription/KozakuraDB/blob/master/README.md.

[23] ELECTRONIC DICTIONARY RESEARCH AND DEVELOPMENT GROUP. *RADKFILE/KRADFILE*. URL: http://www.edrdg.org/krad/kradinf.html.

[24] Kodansha International. *Kodansha's Essential Kanji Dictionary*. Kodansha USA, 2012.

[25] Kim Ahlström, Miwa Ahlström, and Andrew Plummer. *About - Jisho*. URL: https://jisho.org/about.

[26] Jaered Koichi Croes. How to look up and read kanji you don't know. Physical Kanji Dictionary. URL: https://www.tofugu.com/japanese/look-up-kanji/.

[27] The SQLite Consortium. *SQLite Documentation*. URL: https://www.sqlite.org/docs.html.

[28] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. *Lua 5.4 Reference Manual*. URL: https://www.lua.org/manual/5.4/manual.html.

Sobre TEFLONX

Teflon X(cc0 1.0(documentación) MIT(código))es una plantilla de LaTeX creada por David Pacios Izquierdo con fecha de Enero de 2018. Con atribuciones de uso CC0.

Esta plantilla fue desarrollada para facilitar la creación de documentación profesional para Trabajos de Fin de Grado, Trabajos de Fin de Máster o Doctorados. La versión usada es la X

V:X Overleaf V2 with XeLaTeX, margin 1in, bib

Contacto
Autor: David Pacios Izquiero
Correo: dpacios@ucm.es
ASCII: ascii@ucm.es
Despacho 110 - Facultad de Informática

Enrique Ávila Rodríguez
Febrero 2021
Ult. actualización 9 de febrero de 2021
LaTeX lic. LPPL & powered by TEFLON CC-ZERO