# ESCUELA POLITÉCNICA NACIONAL

## FACULTAD DE INGENIERÍA DE SISTEMAS

**Creación de un corpus paralelo para el idioma japonés.**

**TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENCIAS DE LA COMPUTACIÓN.**

**EDUARDO BENJAMIN CARRION ZELAYA**

**eduardo.carrion@epn.edu.ec**

**DIRECTOR: JOSAFA DE JESUS AGUIAR PONTES**

**josafa.aguiar@epn.edu.ec**

**DMQ, 08 2022**

# CERTIFICACIONES

Yo, Eduardo Benjamín Carrión Zelaya declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

**EDUARDO BENJAMIN CARRIÓN ZELAYA**

Certifico que el presente trabajo de integración curricular fue desarrollado por Eduardo Benjamin Carrión Zelaya, bajo mi supervisión.

Digitally signed by Josafá Pontes
DN: cn=Josafá Pontes, o=Escuela Politécnica Nacional, ou=Departamento de Informática y Ciencias de la Computación, email=josafa.aguiar@epn.edu.ec, c=EC
Date: 2022.09.13 23:54:16 -05'00'

**JOSAFA DE JESUS AGUIAR PONTES**

**DIRECTOR**

# DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

EDUARDO BENJAMIN CARRRIÓN ZELAYA

Digitally signed by Josafá Pontes
DN: cn=Josafá Pontes, o=Escuela Politécnica Nacional, ou=Departamento de Informática y Ciencias de la Computación, email=josafa.aguiar@epn.edu.ec, c=EC
Date: 2022.09.13 23:57:00 -05'00'

JOSAFA DE JESUS AGUIAR PONTES

# DEDICATORIA

El presente trabajo dedico a mi Ana Karent y a Amelie que han sido siempre la motivación para siempre continuar y seguir adelante. También a toda mi familia mis padres Wilson y María, mi hermana Raquel que siempre han estado apoyándome. Todos los logros son para ustedes.

# AGRADECIMIENTO

Gracias a Dios que me ha traído por todo este camino. Agradezco el apoyo de todos los que me han acompañado en el transcurso de esta carrera. Agradezco a mi Ana Karent que siempre ha estado a mi lado. De igual manera agradezco a mi familia que no han permitido que me de por vencido. También agradezco la dedicación que han tenido los profesores de preparar sus clases semana a semana para compartir sus conocimientos.

# ÍNDICE DE CONTENIDO

# RESUMEN

Él trabajo que se presenta en este documento tiene como fin la creación de un corpus paralelo constituido por textos en Japones. Este corpus inicia con un conjunto de archivos que pasan por una limpieza inicial para producir el corpus de referencia. En base a este corpus se busca obtener otro corpus paralelo con la inserción de errores de tipo ortográfico. Estos errores son simulados por lo que se conoce como un canal ruidoso. Para la simulación de este canal ruidoso se hace uso de un sistema OCR ya a que los fallos de este sistema representan el canal ruidoso. A continuación del proceso de producción de errores se realiza un alineamiento para poder tener una localización de los errores producidos.

**PALABRAS CLAVE:** Corpus paralelo, OCR, alineación, corpus Japonés, generación de imágenes.

# ABSTRACT

The work presented in this paper aims at creating a parallel corpus of Japanese texts. This corpus starts with a set of files that go through an initial cleaning that produces the reference corpus. Based on this corpus, a parallel corpus is obtained with the insertion of spelling errors. These errors are simulated by what is known as a noisy channel. An OCR system is used to simulate this noisy channel. The errors of this OCR system represent the noisy channel. Following the error production process, an alignment is performed in order to have a localization of the produced errors.



**KEYWORDS:** parallel corpus, OCR, alignment, Japanese corpus, image generation.

# 1. DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El componente desarrollado tiene como fin el crear un corpus paralelo de textos en idioma Japonés. El termino corpus hace mención a una colección de textos almacenados de manera digital. En el presente trabajo se menciona un corpus paralelo constituido por el texto original y un texto degradado (texto que contiene errores ortográficos). Para la degradación de textos se modela un canal ruidoso mediante el uso de un OCR (Optical Character Recognition) el cual introduce errores en el corpus inicial. Los errores introducidos al texto se pueden catalogar como errores de inserción, eliminación y sustitución de caracteres. Una vez obtenido el texto degradado este debe ser alineado al corpus inicial. De este modo se puede comparar y localizar los errores producidos por el canal ruidoso. Creando así un corpus paralelo.

Con el planteamiento descrito se empezó a desarrollar el corpus paralelo. El trabajo inició con un total de 270 archivos en texto plano en idioma Japonés sobre el cual fue necesario realizar una limpieza haciendo uso de expresiones regulares. Dando como resultado un corpus inicial de referencia. En todo el trabajo se aprovechó el lenguaje Perl por su capacidad de procesar texto, se resalta su facilidad para aplicar expresiones regulares al texto que se está procesando.

A continuación, puesto que la entrada del OCR deben ser imágenes, se investigó como transformar el texto plano a imágenes. Para este propósito se hizo el uso del lenguaje LaTex y una herramienta de transformación de PDF a imagen. Para la generación de los PDF se tuvo como desafío el soporte de los caracteres japoneses. Se debe considerar que el sistema de escritura japonés consta de tres tipos de alfabetos: Hiragana (平仮名), Katakana (片仮名) y Kanji (漢字)[1]. Los dos primeros alfabetos contienen 46 caracteres cada uno mientras que los Kanji superan fácilmente los más de 2 mil caracteres. Para el soporte correcto de todos estos caracteres es necesario trabajar con la codificación Unicode. Con esta consideración presente se seleccionó el compilador XeLaTex mismo que permite importar los paquetes con fuentes propias para el japonés.

Con el corpus en PDF se transformó a imagen haciendo uso del programa convert. Ya ingresadas las imágenes en el sistema OCR fue necesario un proceso de prueba y error, en donde se buscó una resolución para las imágenes que produjera un porcentaje de error de 5% a 15%. A continuación, se realizó una normalización de este texto degradado para corregir errores no ortográficos que pueden ser errores en signos de puntuación.

Con el texto ya degradado se realizó una alineación línea a línea entre el corpus original y corpus resultado del OCR. Este paso final se realizó haciendo uso de la herramienta

wdiff. Los textos en los dos corpus se alinean y se calcula la distancia de edición mediante el algoritmo de alineación de cadenas propuesto por Ukkonen. Dando una primera alineación para el corpus paralelo.

## 1.1 Objetivo general

Crear un corpus paralelo de texto Japonés conformado por el texto de referencia y su homólogo OCR.

## 1.2 Objetivos específicos

1. Investigar el soporte de caracteres del idioma Japonés para creación de pdfs.
2. Investigar la generación de imágenes que creen la tasa de errores que se busca en el OCR.
3. Normalizar los textos OCR para reducir errores de tipo no ortográfico.
4. Tokenizar los textos en Japonés mediante la herramienta Mecab.
5. Alinear el texto inicial de referencia con el texto que viene del OCR.

## 1.3 Alcance

Para el componente desarrollado se plantea tener como alcance un corpus paralelo en idioma Japonés. Este corpus paralelo estará constituido por el corpus inicial o de referencia y el mismo corpus degradado que resultará después de pasar por un sistema OCR.

## 1.4 Marco teórico

Para el componente desarrollado fue necesario investigar sobre la codificación para el idioma Japonés. En países occidentales se hace uso de un alfabeto derivado del latín. Idiomas como el español, inglés, portugués, francés, italiano comparten una gran cantidad de letras de sus alfabetos. A medida que las comunicaciones se fueron digitalizando fue necesario tener un estándar para codificar las letras de estos alfabetos.

Con este fin se desarrolló la codificación de caracteres ASCII, (American Standard Code for Information Interchange)[2] la cual asigna un número para todas las letras del alfabeto inglés para que puedan representarse de manera digital. Este estándar soporta todas las letras del alfabeto inglés, sin embargo, para alfabetos de otros idiomas se ha creado una extensión que añada caracteres como la ñ, ç o acentos sobre las vocales. Con está codificación ASCII extendida se puede representar sin problema los idiomas occidentales.

Los idiomas no occidentales han ido desarrollando las codificaciones propias para su lenguaje así dar soporte a los caracteres de sus alfabetos. En este caso el Japonés creo la codificación Shift JIS (Shift Japanese Industrial Standards) para dar soporte a los caracteres de sus alfabetos. Una manera mucho más global de abordar la codificación de caracteres para todos los idiomas fue creada por Unicode[3]. El estándar Unicode define code points para los caracteres de todos los idiomas conocidos. Para hacer una referencia a estos code points se hace uso del estándar de codificación UTF-8 (Unicode Transformation Format) en la mayoría de los casos, aunque es posible usar también las versiones de 16 y 32 bits (UTF-16 y UTF-32). De esta manera, se consigue dotar a todos los caracteres del idioma Japones de una codificación digital para poder mostrarlos en una computadora.

Con la codificación correcta para caracteres se puede empezar a crear un corpus de texto. Un corpus según la definición de la RAE es un “conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación.” [1] Partiendo de un corpus inicial se puede tener uno similar con cambios que pueden ser que el texto este degradado (con errores) o que el texto a sido traducido a un idioma diferente en ambos casos se conoce al corpora (plural de corpus) como corpus paralelo.

Un corpus paralelo tiene gran valor puesto que permite evaluar el comportamiento de los algoritmos de corrección ortográfica. Para la creación de un corpus paralelo se toma como base el modelo de canal ruidoso que fue propuesto en los laboratorios AT&T BELL por Kernighan, M. D., K. W. Church, and W. A. Gale [4]. En el modelo del canal de ruido, imaginamos que la forma de la superficie visible que tenemos de las palabras es en realidad una forma "distorsionada" de la palabra original que pasó por el canal de ruido. Estas distorsiones en el texto se generan por diferentes factores. Para este trabajo se considera como un canal ruidoso un sistema de OCR(Optical Character Recognition). Las fallas de este sistema simulan el comportamiento de un canal ruidoso.

Los tipos de errores resultantes de un canal ruidoso se pueden clasificar en tres tipos básicos, estos pueden ser: de sustitución, inserción y eliminación. Los errores de sustitución se dan cuando una letra es sustituida por otra que puede ser similar. Los errores de inserción pueden darse cuando a la palabra correcta se agrega una letra más. Un error de eliminación se da cuando a una palabra se le ha quitado una letra. En la figura 1 pueden encontrarse ejemplificados estos tipos de errores para una mejor comprensión. Estos errores son generados por un OCR cuando un carácter no es reconocido de manera correcta.

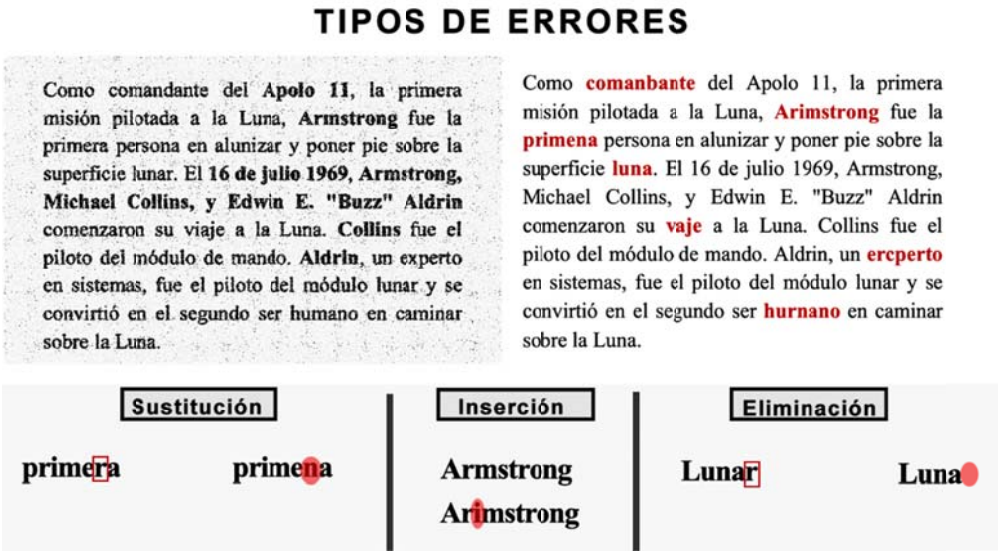


**Figura 1** Tipos de errores localizados en textos degradados.

Esta clasificación de errores es una ayuda para que, posteriormente, se calcule una distancia de edición. En base a esta distancia de edición es posible mostrar una corrección para la palabra. Ya que una forma de determinar si dos textos son similares es calcular sus distancias de edición. Cuanto más larga sea la distancia de edición, más cambios habrá que hacer en el texto de origen para producir el texto de destino. Si dos textos tienen la misma longitud de distancia de edición, son muy similares. Por el contrario, si los dos textos tienen longitudes diferentes de distancia de edición, se puede decir que son muy diferentes. Para este trabajo se utilizó el algoritmo de alineación de cadenas propuesto por Esko Ukkonen[5].

Como un último punto importante, se debe considerar que el sistema de escritura Japonés no contempla el espacio en blanco como un delimitador entre palabras, así como lo hacen varios idiomas occidentales. Una oración en Japones se puede escribir sin espacios en blanco. Varios de los algoritmos encaminados al procesamiento de texto

empiezan el procesamiento de las palabras que no representan un desafío para ser tokenizadas por los espacios en blancos. En el idioma Japones no se tiene la noción de palabras sino de monemas que se comprenden como las unidades mínimas de significado. El parser MeCab permite encontrar estas unidades mínimas del lenguaje que, como hemos dicho, en el caso del Japones se llaman monemas. Esta herramienta es un parser ya que no solo permite la segmentación de texto en monemas, sino que también brinda una descripción del monema segmentado. Con esta tokenización de monemas es posible aplicar los algoritmos de procesamiento de texto bien conocidos.

# 2 METODOLOGÍA

El desarrollo del presente trabajo se puede diferenciar en 4 etapas puntuales, las cuales son:

- Limpieza del corpus.
- Procesamiento del corpus.
- Normalización del corpus.
- Alineación del corpus a nivel de líneas.
- Tokenización del corpus.
- Alineación del corpus a nivel de monemas.

En la figura 2 se puede identificar de manera gráfica la secuencia de estás etapas y como están interconectadas entre si.

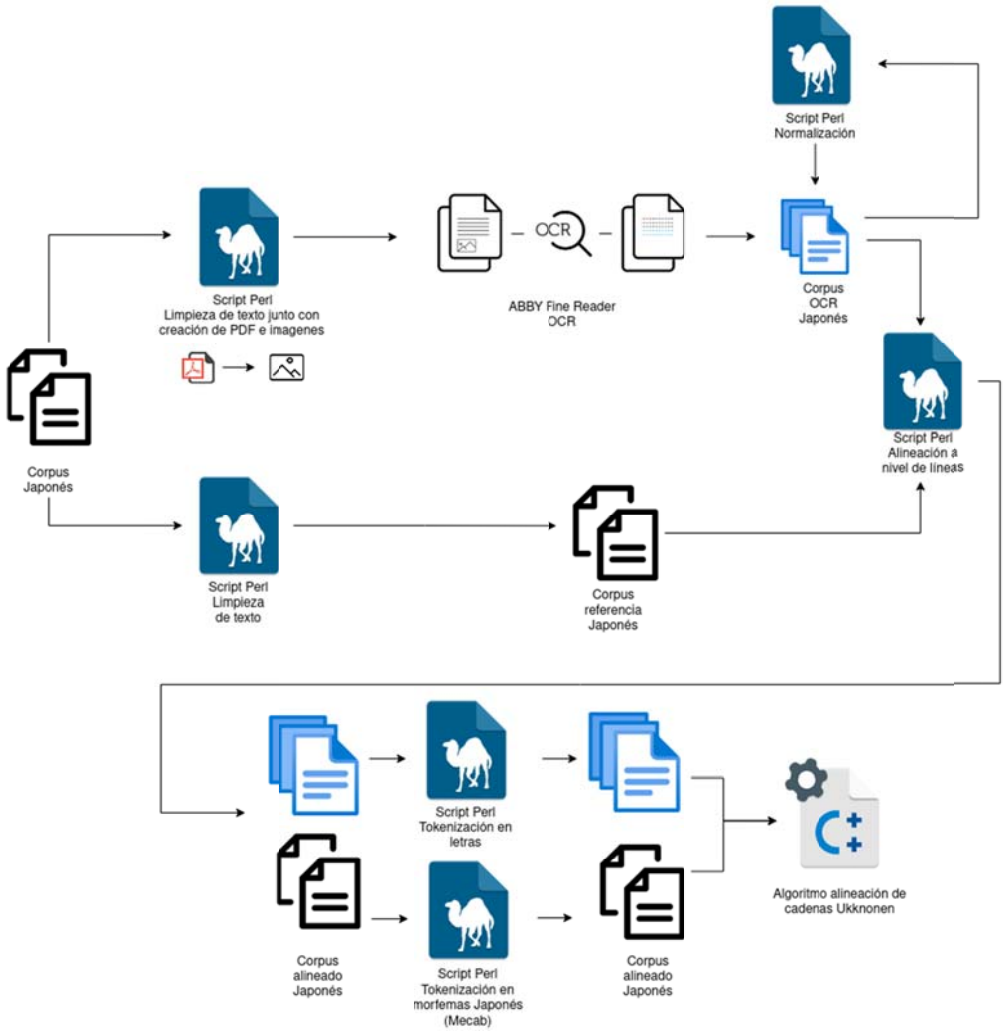


**Figura 2** Diagrama del proceso.

Para el desarrollo de este trabajo se recibieron 230 archivos con textos en Japonés. Estos archivos constituyen el corpus inicial que se encuentra en formato de texto plano y a partir de estos se desarrolló todo el trabajo presentado.

## 2.1 Limpieza del corpus

En un análisis inicial del corpus se encontró que los saltos de línea estaban en el formato de sistema operativo DOS, es decir se componen por los caracteres <CR><LF> (Carriage Return y Line Feed) también denotado como '\r\n'. En vista de que todo el trabajo se desarrolla en el sistema operativo Linux y este solo hace uso del carácter '\n', para los saltos de línea es necesario con una conversión de formato. Para esto se hizo uso del comando `dos2unix *` que permite realizar la conversión requerida.

Con los archivos de texto ya en formato UNIX se hizo la limpieza de estos. En Perl se creó expresiones regulares que permitan la limpieza del texto. Como primera limpieza se eliminó una columna que antecedía a cada línea de texto. En la figura 3 se puede observar la expresión regular y la columna que es eliminada.

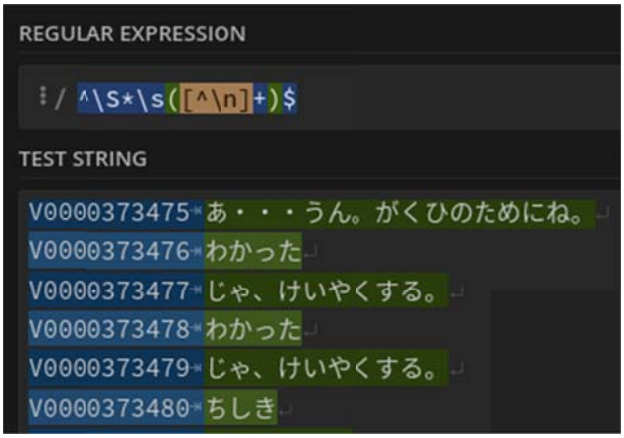


**Figura 3** Limpieza columna de texto.

Seguido a esto, como fue descrito en la sección del marco teórico, el Japonés es un idioma que no hace uso de espacios en blanco, por lo cual todos los espacios en blanco dentro del corpus se eliminan como parte de esta limpieza. Ya que se encontró líneas repetidas, también se realizó la limpieza de estas. En la figura 4 se muestra la expresión regular utilizada para eliminar líneas repetidas.

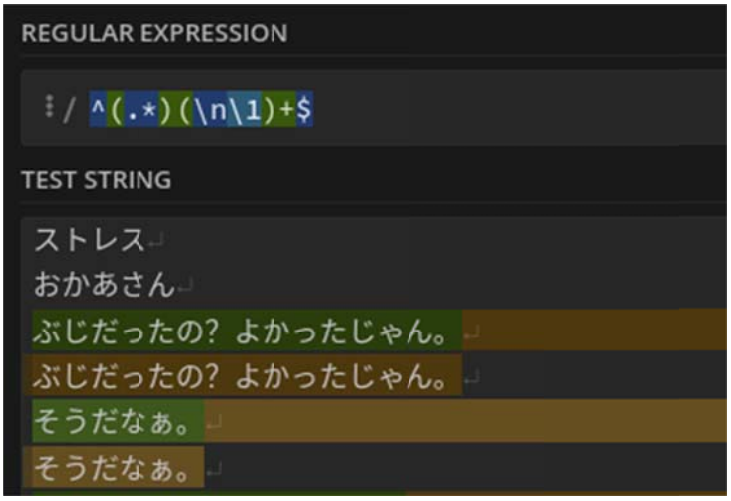


**Figura 4** Limpieza de líneas de texto repetidas.

Otro tipo de líneas que se optó por eliminar fueron las líneas que contienen un solo carácter ya que carecen de contexto para cualquier procesamiento de texto futuro.

Durante esta limpieza se halló ciertos caracteres que se presentan con errores y no pertenecen al Japones por lo que se los elimino del corpus. Un ejemplo de estos caracteres dañados se muestra en la figura 5. Los siguientes son los caracteres que fueron eliminados del corpus ‘ 哦嗷爨幺懋毡耙睨缏 ’.


Figura 5 Caracteres con errores

Con esta limpieza básica se tiene un corpus bien dispuesto sobre el cual se puede trabajar. Dado que se realizó ya una limpieza el texto plano se encuentra en óptimas condiciones para ser transformado a imagen.

## 2.2 Procesamiento del corpus

Para la transformación del texto a imagen se planteó cambiarlo a PDF y luego desde este formato a imagen, dado que existen herramientas que permiten este cometido. Para la parte de transformación a PDF se hace uso del lenguaje LaTex. Una vez los textos se

encuentran en formato PDF se los transforma a imagen mediante el programa `convert` que pertenece al paquete ImageMagick.

En este proceso, para generar PDFs, fue necesario investigar la mejor manera de dar soporte a los caracteres en japones, dado que se encontró que el compilador pdflatex tiene un soporte algo básico para el idioma Japonés. Al momento de generar los PDFs se encontró que varios caracteres eran omitidos y no se mostraban, como solución a este problema se encontró el compilador XeLaTex que cuenta con un soporte nativo para caracteres Unicode. Con este compilador solo fue necesario importar las fuentes de impresión para los caracteres japoneses. Para el presente trabajo se hizo uso de las fuentes Noto Sans Japanese[3] desarrolladas por Google puesto que es uno de los conjuntos de fuentes con el soporte más completo para caracteres japoneses.

Para el proceso de transformar el texto plano a PDF se creó en base a los archivos del corpus archivos `.tex` que se compilan mediante el comando XeLaTex ya mencionado. Para crear estos archivos se consideró que el tamaño de fuente de impresión en el PDF debe ser el menor para que se generen errores en el OCR. También se considera escapar los caracteres propios del lenguaje LaTex. En la figura 6 se puede observar un ejemplo de estos archivos.

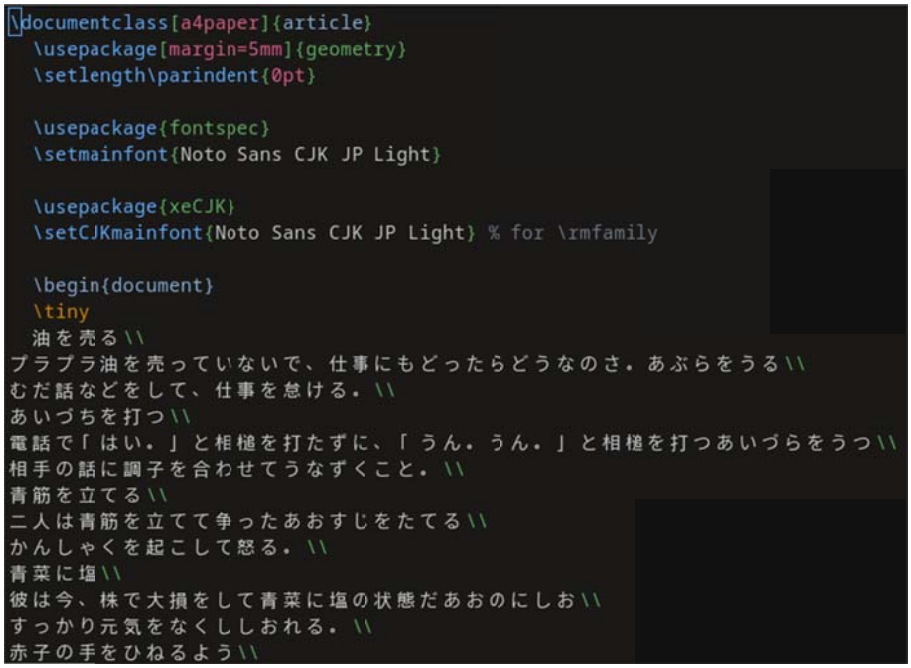


**Figura 6** Ejemplo archivos `.tex`.

A continuación, con los archivos `.tex` generados se compila los PDF. Vale mencionar que algunos archivos con tamaño mayor a 2 MB excedían la capacidad de memoria del compilador por lo que se tomó como estrategia dividir estos archivos haciendo uso del comando `split`. Se optó por dividir estos archivos grandes cada 10 000 líneas, compilar estos fragmentos del archivo inicial y luego volverlos a unir.

Con los archivos que ya están en formato PDF se pasa a convertirlos en imagen mediante el comando ya mencionado, `convert` . Este comando fue utilizado porque permite elegir los DPI (Dots Per Inch) y la compresión de la imagen de salida. Con estos parámetros se puede ir variando la imagen que sirve de entrada para el sistema OCR. Dentro del sistema OCR se encontró que en las imágenes con bajo DPI existen filas que no llegan a ser reconocidas y el sistema termina eliminando estas filas. Por esta razón se configuro el comando con una resolución de 400 DPI.

Cuando ya se tuvo el corpus en formato de imagen, fue posible ingresarlo al sistema OCR. Aquí fue necesario hacer algunos ajustes para el correcto funcionamiento del sistema OCR, dado que el Japonés puede ser escrito de manera horizontal o vertical, es necesario configurar en el sistema OCR de tal modo que asimile que la entrada está en formato vertical y así no cometa errores para reconocer los caracteres. También se debe ajustar para que el sistema OCR siempre inserte el salto de líneas ya que, en algunas ocasiones, cuando se guardar como texto plano, se tienden a borrar los saltos de línea.

## 2.3 Normalización del corpus

El corpus que viene del OCR contiene errores ortográficos, sin embargo, existen errores que no son de ésta tipología, sino que son una variación del caractér dado pues en Japonés se tienen versiones propias de algunos caracteres latinos. En la figura 7 se muestran estas variaciones sobre los caracteres. El OCR tiende a reconocer cara caractr como un caractr latino, por lo cual, el OCR detectaría este caracter como un error, pero no es un error ya que es una variación del mismo.

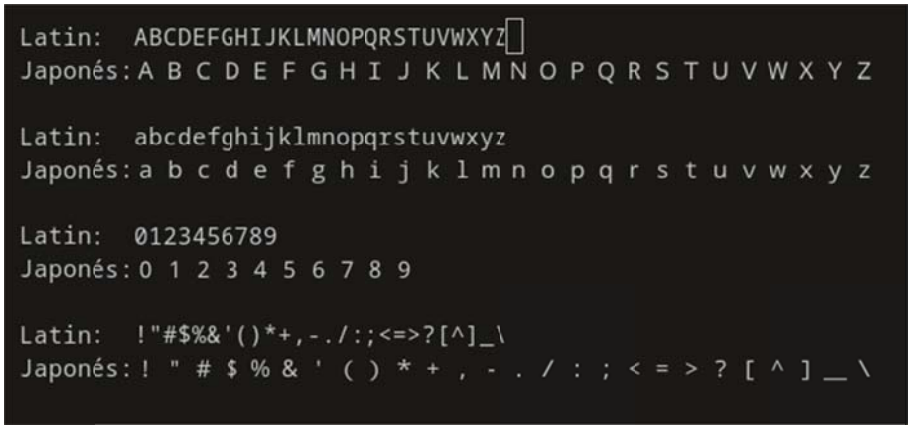


Figura 7 Comparación de caracteres latinos con Japoneses.

En el proceso de normalización se revisan todos estos casos y otros más para dejar un texto consistente que únicamente tenga errores de ortografía, es decir, caracteres japoneses cambiados. En las figuras 8 se muestra una comparación entre el corpus inicial y corpus OCR antes de ser normalizado. Se puede observar que se encuentran diferencias entre los caracteres.

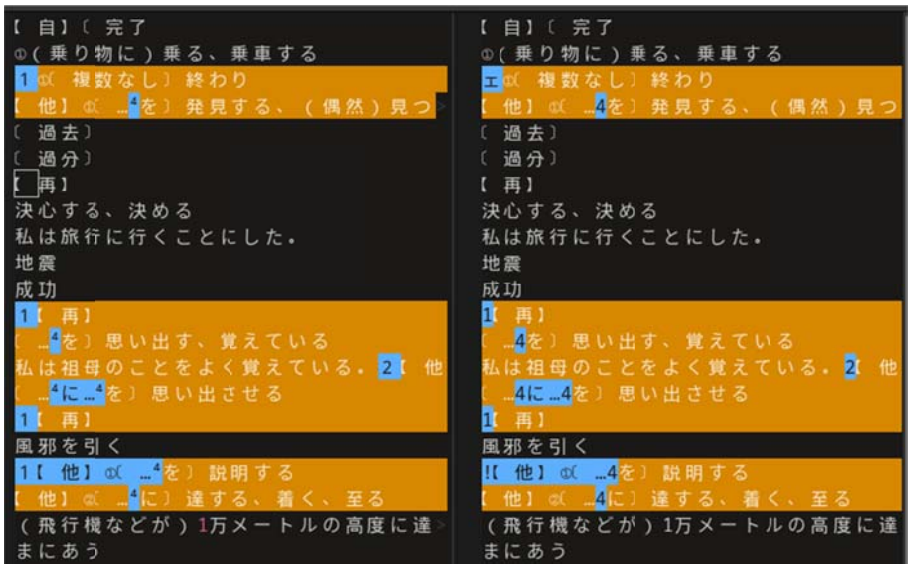


**Figura 8** Texto antes de ser normalizado

## 2.4 Alineación del corpus a nivel de líneas

Con los textos ya normalizados se realiza un proceso de alineación línea por línea. Con este proceso se busca alinear el corpus inicial con el corpus OCR en base a las líneas de estos. En el sistema OCR se encuentra el problema de que algunas líneas no logran ser reconocidas, por lo tanto, se tienen líneas faltantes. Otro error que se encuentra respecto

a las líneas es, que al producirse el archivo PDF, cuando una línea es muy larga, esta ocupa más líneas dentro del PDF, por lo cual, al reconocerse con el sistema OCR, se introducen saltos de línea que deben ser eliminados. En la figura 9 se muestra un ejemplo de está alineación.

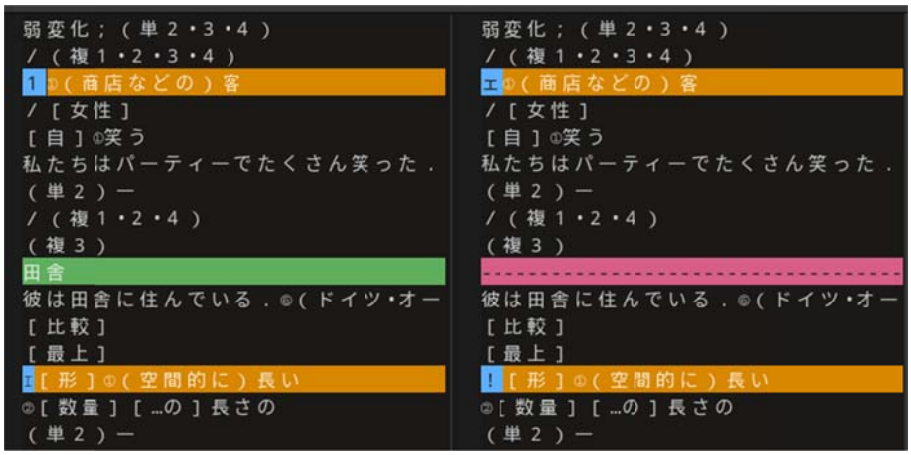


**Figura 9** Ejemplo alineación línea faltante.

Para llevar a cabo esta alineación se utiliza herramientas del sistema GNU, en particular, se utilizó la herramienta wdiff, ya que esta herramienta permite tener una alineación por palabras. Dado que nuestro texto al momento se encuentra sin espacios, una línea completa se interpreta como una palabra, lo cual ayuda para el propósito de alinear líneas. Dentro de un script Perl se llama a este comando wdiff para procesar luego su salida y tener los archivos del corpus paralelo alineados a nivel de línea.

## 2.5 Tokenización del corpus

Con el corpus alineado a nivel de líneas se puede empezar a segmentar o tokenizar por monemas. Este proceso solo se puede realizar en el corpus inicial ya que el corpus OCR, al tener errores, no se identifican de manera correctas las palabras para este. Para este paso se utiliza el parser MeCab el cual es una herramienta bien conocida para la tokenización de cadenas de texto en Japonés. En la figura 10 se muestra la salida de este parser para la cadena “は誰でも編集できるフリー百科事典です” .

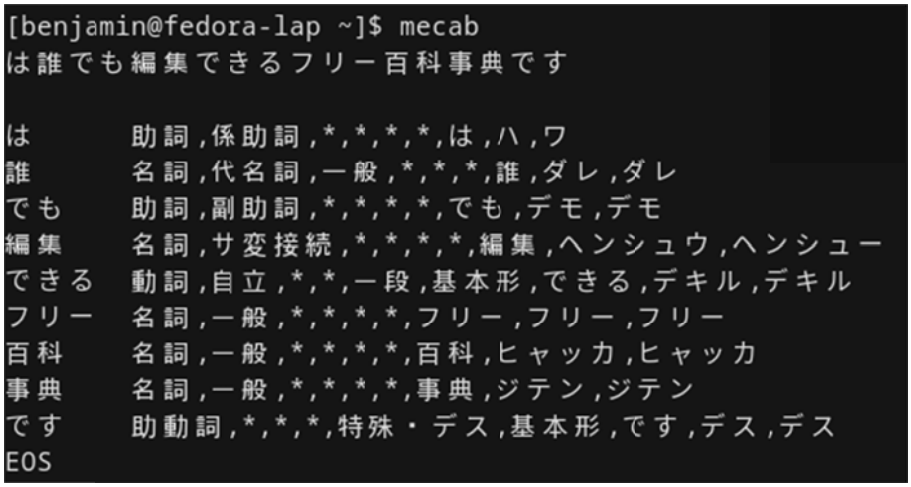


**Figura 10** Salidad parser Mecab.

De esta salida solo necesitamos la primera columna que es en donde está segmentado por monemas la cadena de entrada. De este modo, se tiene el corpus inicial segmentado por espacios.

Para el corpus OCR solo se añaden espacios entre cada caracter para un proceso de alineación a nivel de monemas (palabras)

## 2.6 Alineación del corpus a nivel de monemas

En la fase final de este trabajo se realiza una alineación a nivel de monemas. Para este fin se utiliza el algoritmo de alineación de cadenas propuesto por Esko Ukkonen. Aquí se reutilizó una implementación ya creada de este código en C, aunque fue necesario una modificación para soportar los caracteres en Japonés. Como fue mencionado en el marco teórico, el idioma Japonés necesita de una codificación diferente al estandar ASCII. Las variables char están orientadas a soportar solamente caracteres ASCII de 8 bits. Tampoco es viable el uso de una codificación UTF-8 ya que tiene una longitud variable dependiendo del caracter. Puesto que se necesita realizar una comparación de caracteres, es necesario utilizar las variables de tipo wchar que utilizan 32 bits para representar completamente los code point de Unicode. Cambiando el tipo de variable y algunas funciones fue posible tener un funcionamiento de este código en C para el idioma Japonés. En la figura 11 se analizar una alineación básica que permitiría este algoritmo de Ukkonen

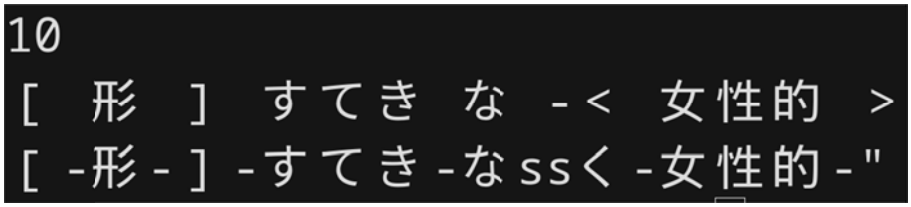


**Figura 11** Alineación de cadenas mediante el algoritmo Ukkonen.

# 3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

## 3.1 Resultados

Como resultado de este trabajo se obtuvieron varios archivos con diferente formato. Cada fase tiene el corpus inicial modificado en alguna manera para producir el corpus paralelo.

Vale mencionar que de los 230 archivos 2 se descartaron. Uno de estos archivos se descartó producto de la limpieza dado que contenía filas con un carácter únicamente. El otro caso que se debió descartar fue porque se encontró 2 archivos con el mismo contenido.

En corpus inicial se diferencia que tiene un total de 1607861 mientras que para el corpus OCR se tiene un total de 2419746. Lo cual indica un 150% de líneas aumentadas por parte del OCR. En esto se observa cierta tasa de cambio que introduce el OCR sobre los archivos.

Para evidenciar el cambio entre archivos, se pueden mirar las siguientes imágenes, en cada una se observa como los archivos van cambiando. En los anexos se muestra detalladamente para entender estos archivos de manera completa.

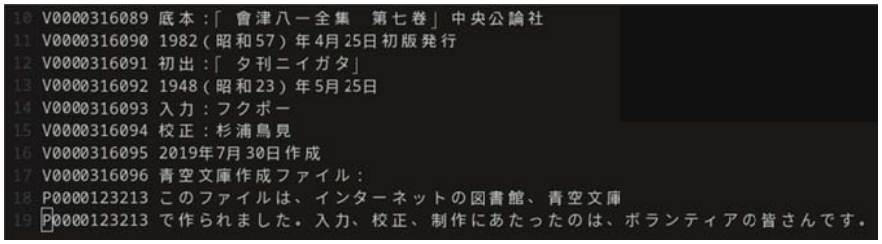


**Figura 12** Corpus Inicial

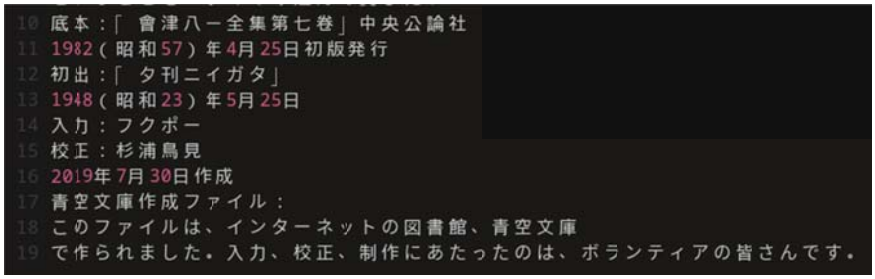


**Figura 13** Corpus limpiado

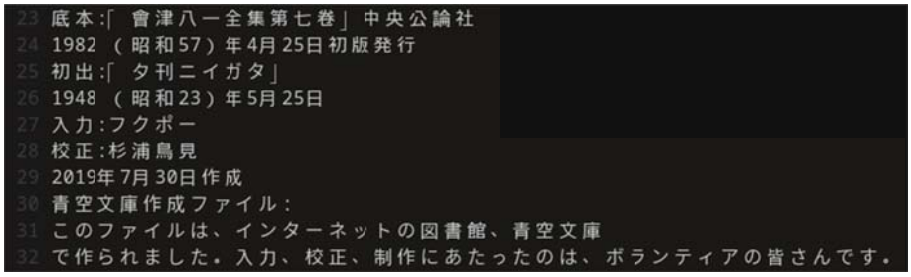


**Figura 14** Corpus OCR

## 3.2 Conclusiones

El presente trabajo considera crear un corpus paralelo de textos en Japonés, este objetivo fue logrado y se detalla a continuación, en las conclusiones relacionadas con la obtención de este objetivo.

1. El primer desafío encontrado en este trabajo fue trabajar con cadenas de texto en Japonés, estas cadenas presentan sus particularidades y al momento de trabajar con las mismas, es necesario estar familiarizado con el conocimiento de cada una de ellas. Para el presente trabajo se debió investigar y comprender el uso correcto de caracteres Unicode para su manejo dentro de los lenguajes LaTex y C. Comprendiendo estos conceptos se logró producir los resultados esperados en cuanto al manejo de este idioma de manera digital.

2. Otro punto que se consideró importante fue el de la creación de las imágenes que funcionen con el sistema OCR. Este sistema tiene un comportamiento propio que fue necesario analizar. Puesto que se utilizó el software ABBY FINE Reader que es de carácter privativo, no se pudo tener acceso a su implementación para conocer con qué tipo de imágenes responden al resultado deseado. Fue necesario un proceso de prueba y error variando las resoluciones de las imágenes para tener un resultado que tuviera errores de un 5 al 15%. Se encontró que las imágenes generadas con una resolución de 400 DPI y 90% de compresión sobre los pdfs son las apropiadas para el trabajo presentado.

3. Un punto nuevo que se debió considerar al trabajar con este idioma Japonés fue que no existen espacios como separador de texto, por lo cual se podría pensar que varios métodos que se tienen para el procesamiento de texto pierden validez. Sin embargo, se encontró que existen tokenizadores que permiten la segmentación de textos para tenerlos separados como en idiomas occidentales.

Así, de este modo se pueden aplicar técnicas de procesamiento de texto que se han aprendido en el transcurso de la carrera.

4. Como último desafío se tuvo la necesidad de que el corpus paralelo se encuentre alineado con el fin de localizar de manera pronta los errores. Aquí no se logró un éxito total puesto que se tiene una alineación muy básica proporcionada por el algoritmo de alineación de cadenas de Ukkonen. Esto puede ser una alineación inicial, pero se necesita una alineación mucho más exhaustiva que nos permita tener bien localizados los errores.

5. Como un factor adicional se menciona que el presente trabajo tenía un alcance mucho mayor que debió ser modificado. Debido a las restricciones de tiempo no fue posible culminar este trabajo con tal alcance. Esto se debe a que al momento de cursar la materia diseño de TIC se contaba con 93 créditos aprobados y los reglamentarios para tomar la materia son 105. Por tal motivo la carga horaria al momento de tomar el Trabajo de Integración Curricular aún era alta, demandando más tiempo.

## 3.3 Recomendaciones

Para el desarrollo de un trabajo similar se recomienda un acercamiento a la codificación del idioma Japonés. Si no se tiene un conocimiento básico de esto se presentan muchas dificultades al momento de manejar este idioma.

Como siguiente paso para este trabajo se recomienda la alineación de monemas que puede realizarse mediante el algoritmo expectation–maximization teniendo en cuenta que se realizó un intento de uso de este algoritmo mediante la herramienta GIZA++ sin mayor éxito por lo cual también se sugiere el uso de los mecanismos de Atención de Bahdanau para una posible solución a esta dificultad.

En la literatura revisada se encontró que la distancia de edición puede no ser útil para la corrección ortográfica en textos para el idioma Japonés.[9] Ya que existen muchas palabras con una longitud de 2 caracteres y la distancia de edición pierde su utilidad. Se sugiere que se empiece la corrección ortográfica por el uso de bi-gramas o tri-gramas.

# 4 REFERENCIAS BIBLIOGRÁFICAS

[1] Kazuko Nakajima (2002). Learning Japanese in the Network Society. University of Calgary Press. p. xii. ISBN 978-1-55238-070-3

[2] C. E. Mackenzie, Coded character sets history and development. Reading, Mass: Addison-Wesley, 1980.

[3] “The unicode® standard: A technical introduction,” [Unicode]. [Online]. Available: https://www.unicode.org/standard/principles.html. [Accessed: 11-Sep-2022].

[4] M. D. Kernighan, K. W. Church, and W. A. Gale, “A spelling correction program based on a noisy channel model,” Proceedings of the 13th conference on Computational linguistics -, 1990.

[5] E. Ukkonen, “Algorithms for approximate string matching,” Information and Control, vol. 64, no. 1-3, pp. 100–118, 1985.

[6] MeCab: Yet another part-of-speech and Morphological Analyzer. [Online]. Available: https://taku910.github.io/mecab/#parse. [Accessed: 11-Sep-2022].

# 5 ANEXOS

Como anexo para este trabajo se presenta la variación uno de los archivos en 3 etapas diferentes para que se pueda observar el cambio a través de cada una de.

# ANEXO I

Archivo inicial: archivo_249_out.txt

# ANEXO II

Archivo referencia: archivo_249_out.ref

# ANEXO III

Archivo OCR: archivo_249_out.ocr