ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA EN SISTEMAS

DESARROLLO DE UN CORPUS PARALELO DE TEXTO E IMÁGENES EN EL IDIOMA JAPONÉS PARA FUTUROS TRABAJOS DE INVESTIGACIÓN EN PROCESAMIENTO DE LENGUAJE NATURAL

GENERACIÓN DE UN CORPUS PARALELO EN EL IDIOMA JAPONÉS OBTENIDO A TRAVÉS DE UN CORPUS REFERENCIAL Y UN CORPUS OCR

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

JHOANN SEBASTIÁN RUEDA VANEGAS

Jhoann.rueda@epn.edu.ec

DIRECTOR: JOSAFÁ DE JESÚS AGUIAR PONTES

josafa.aguiar@epn.edu.ec

DMQ, agosto 2022

CERTIFICACIONES

Yo, JHOANN SEBASTIÁN RUEDA VANEGAS 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.

SEBASTIÁN RUEDA

Certifico que el presente trabajo de integración curricular fue desarrollado por JHOANN SEBASTIÁN RUEDA VANEGAS, bajo mi supervisión.

JOSAFÁ DE JESÚS AGUIAR PONTES

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.

JHOANN SEBASTIÁN RUEDA VANEGAS

JOSAFÁ DE JESÚS AGUIAR PONTES

DEDICATORIA

Dedicado a Dios y a mi familia.

AGRADECIMIENTO

Agradezco profundamente primero a Dios por la oportunidad de educarme, por su gracia inmerecida y por su fidelidad en mi vida, agradezco mis padres Ana y Fernando por su apoyo y acompañamiento, este logro es para ustedes, a Cristián por ser mi ejemplo a seguir y por motivarme a no rendirme, a Camilo por acompañarme en los buenos y malos momentos, a Dani por su ayuda y en general gracias a mis amigos, que al igual que mi familia me han visto sufrir para llegar a este punto y me han apoyado. Agradezco a mis profesores, a mi tutor el PhD Josafá Pontes por compartir sus conocimientos conmigo y por ayudarme en este último tiempo, agradezco al PhD José Lucio por la dedicación y preocupación hacia sus estudiantes. No hubiera llegado hasta aquí de no ser por cada una de las personas que he tenido el privilegio de conocer y de quienes he logrado aprender algo y le han dado alguna enseñanza a mi vida. Por último, me agradezco a mí, porque solo Dios y yo sabemos lo que ha tenido que suceder para llegar hasta aquí.

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo consiste en la investigación sobre el proceso para generar un corpus paralelo dado que se recibieron datos en archivos de texto plano en el idioma japonés, el proyecto consta de la limpieza de los datos para generar un corpus referencial, posterior a ello se busca pasar los datos por una herramienta OCR, con el fin de generar errores ortográficos debido al proceso que implementa internamente la herramienta OCR. Al tener los dos archivos, los de referencia y los generados por la herramienta OCR con errores ortográficos, se normalizan ambos archivos con el fin de compararlos con la herramienta diff para determinar la similitud que existe entre cada archivo referencial y su respectivo archivo OCR. Cuando se alcance el porcentaje de similitud esperado, la siguiente parte es alinear los archivos de referencia y los archivos OCR a nivel de líneas y esto conformará el resultado que es el corpus paralelo.

PALABRAS CLAVE:

Corpus: Conjunto ordenado de datos o textos científicos que pueden ser usados como base para una investigación.

OCR: Reconocimiento óptico de caracteres

HMM: Cadenas Ocultas de Markov

CRF: Campos aleatorios condicionales

ABBY: Herramienta OCR

Diff: Herramienta para la comparación de textos

Mecab: Tokenizador de monemas para el idioma japonés

ABSTRACT

The present work consists of the investigation on the process to generate a parallel corpus given that data were received in plain text files in Japanese language, the project consists of the cleaning of the data to generate a referential corpus, after that it is sought to pass the data through an OCR tool, in order to generate misspellings due to the process that the OCR tool implements internally. Once the two files, the reference files and those generated by the OCR tool with spelling errors, are normalized both files in order to compare them with the diff tool to determine the similarity that exists between each reference file and its respective OCR file. When the expected similarity percentage is reached, the next part is to align the reference files and the OCR files at line level and this will form the result which is the parallel corpus.

KEYWORDS:

Corpus: An ordered set of scientific data or texts that can be used as a basis for research.

OCR: Optical Character Recognition

HMM: Hidden Markov Models

CRF: Conditional Random Field

ABBY: OCR tool

Diff: Text comparison tool

Mecab: Moneme tokenizer for Japanese language

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El componente desarrollado y descrito en el documento presente se basa en la investigación para la preparación, creación de errores, normalización y alineación de los datos de un corpus en el idioma japonés, con el fin de generar un corpus paralelo.

Dichos datos pasan por un proceso general de limpieza de datos, esto con el fin de preparar los datos para que puedan ser utilizados en el futuro desarrollo de modelos probabilísticos y de redes neuronales para la detección de errores ortográficos en el idioma japonés.

El componente se trata de investigar el proceso a seguir para tomar los datos del corpus referencial, dividirlo en dos grupos importantes (datos de entrenamiento y datos de prueba). Luego, por medio de técnicas de procesamiento de lenguaje natural generar errores en el corpus referencial y posterior a ello comparar los datos para determinar el porcentaje de similitud que tienen dichos datos. Los datos con errores generados en base a los datos del corpus de referencia se denominaron datos OCR, que viene de las siglas (Optical Character Recognition). Esto debido a que la técnica que se usó para generar los errores ortográficos se basó en el reconocimiento óptico de caracteres.

La investigación hecha en el trabajo de titulación fue demostrando que para generar estos errores era necesario hacerlo a través de una herramienta OCR, que puede tomar varios tipos de archivos para implementar el reconocimiento óptico y generar los datos con errores ortográficos. En el caso de la investigación hecha se determinó que los archivos que recibirá como entrada la herramienta OCR serán imágenes generadas de los textos en japonés y una vez que se procesen, la salida de la herramienta OCR será un archivo de texto plano correspondiente para cada documento del corpus referencial.

Ya con los dos archivos, de referencia y OCR, la siguiente parte es normalizar los archivos OCR, este proceso se hace cambiando ciertos símbolos que pueden ser reemplazados dentro del texto sin cambiar el sentido del texto.

Luego, se deben comparar los archivos de referencial contra los archivos OCR correspondientes para ver el porcentaje de similitud entre ambos archivos, para esto fue necesario usar la herramienta comparadora de textos wdiff y vimdiff, ambas herramientas se usaron desde la terminal de Linux, con wdiff se usaban algunos comandos para mostrar las líneas que contienen diferencias y aún más importante el wdiff tiene el parámetro –s para ver el porcentaje de similitud entre los archivos. Por otro lado, el vimdiff provee la opción de vista mucho más sencilla, esto es útil debido que es más fácil ver qué símbolos no coinciden en la línea donde se reporte el error. Esta herramienta subraya con colores

las líneas que faltan, las letras que son diferentes y las líneas donde se presentan dichas diferencias.

Este proceso es un proceso repetitivo, pues al hacer la comparación entre los archivos se pueden detectar las diferencias y, por consiguiente, agregar las correcciones correspondientes en la normalización, para verificar nuevamente el porcentaje de similitud. Cuando los archivos tienen una similitud superior al 90%, podemos asegurar que la alineación de los archivos a nivel de líneas estará completa y funcionará satisfactoriamente.

Luego, se tomará la salida de la herramienta diff para conformar el corpus paralelo conformado por los archivos de referencia y los archivos OCR, cumpliendo el objetivo de este trabajo de titulación.

1.1 Objetivo general

Producir un corpus paralelo de texto plano en el idioma japonés conformado por un texto de referencia alineado con el respectivo texto OCR.

1.2 Objetivos específicos

1. Investigar sobre la limpieza de los archivos de referencia.
2. Limpiar los archivos de referencia.
3. Generar imágenes de los archivos de referencia.
4. Pasar las imágenes generadas por un OCR con el fin de generar errores ortográficos.
5. Normalizar los archivos de referencia y OCR.
6. Alinear línea a línea los archivos de referencia y OCR.
7. Producir el corpus paralelo

1.3 Alcance

En el proyecto deberá ser desarrollado a partir de la limpieza de datos en formato de texto plano, generar errores ortográficos por medio de un OCR, esto incluye la limpieza, la normalización y alineación de los textos línea a línea de referencia y OCR. Dentro del alcance se deberá generar un corpus paralelo que supere un porcentaje mayor al 90% de similitud entre archivos.

1.4 Marco teórico

ANTECEDENTES

Los correctores de errores ortográficos son herramientas muy usadas en la corrección de errores ortográficos (Kukich., 1992). Para el caso de las lenguas de origen oriental, en donde las palabras no tienen un delimitador entre ellas, es un problema sin resolver en la computación lingüística (Nagata).

Los procesadores de lenguaje natural, son herramientas de gran impacto en la actualidad, la traducción de textos, la corrección de errores ortográficos son parte de las capacidades que se tiene con el procesamiento del lenguaje natural.

Estas nuevas herramientas generadas a través del uso de inteligencia artificial, requieren de la información necesaria para poder entrenarse. Estos datos son imprescindibles con el fin de alimentar a la inteligencia artificial que se desea desarrollar.

Los correctores ortográficos son altamente usados hoy en día en muchas de las áreas de trabajo en donde personas del día a día desarrollan tareas ligadas al procesamiento de lenguaje natural, actualmente para las computadoras se han desarrollado herramientas que se encargan de corregir los textos.

Algunos algoritmos que se son importantes para estos procesos son: Cadenas ocultad de Markov (HMM), Estimación de parámetros de campos aleatorios condicionales (CRF).

Y existen herramientas que por detrás utilizan algoritmos como los mencionados, al igual que usan gramáticas libres o sujetas al contexto.

Mecab: Es un motor de análisis morfológico de código abierto, desarrollado como un proyecto conjunto entre el departamento de Investigación de la Información de la Universidad de Kioto y el Nippon Telegraph and Telecommunications Communication Science Laboratories. (West, 2014).

Mecab no depende de ningún corpus de lenguaje, esta herramienta es interesante, pues usa estimación de los parámetros de los campos aleatorios condicionales o por sus siglas en inglés CRF, mejorando sobre los modelos ocultos de Markov usado por otros. (West, 2014)

Estimación de Parámetros de Campos Aleatorios Condicionales (CRF): Es un método probabilístico para la predicción estructurada y se usa en varias áreas, como en el procesamiento de lenguaje natural.

Los CRF pueden entenderse como una extensión del clasificador de regresión logística a estructuras gráficas arbitrarias, o como un análogo discriminativo de los modelos generativos de datos estructurados, como los modelos de Markov ocultos. (Sutton & McCallum, 2012).

Modelo oculto de Márkov (HMM): Son una base formal para hacer modelos probabilísticos de problemas de "etiquetado" de secuencias lineales (Rabiner, 1989)**.**

Las cadenas ocultas de Markov están basadas en las cadenas de Makov, en las que dado una información del tiempo presente nos permitirá predecir cuál podría ser el resultado en instantes futuros del tiempo.

Gramática Libre de Contexto: La gramática libre de contexto está sujeta a la siguiente regla V→w, donde V representa a un símbolo no terminal, mientras que w está compuesta por símbolos terminales y no terminales.

V que es un no terminal, puede ser sustituido por un w en cualquier momento, sin tener en cuenta el contexto en el que ocurra.

Gramática dependiente del Contexto: La gramática dependiente del contexto se puede definir como una tupla (V, Σ, P, S)**.** (Vladimir, s.f.)

Donde cada elemento tiene un significado:

- V, alfabeto de símbolos no terminales.
- Σ, Alfabeto de símbolos terminales
- P, Es un conjunto de reglas
- S, Símbolo inicial

2 METODOLOGÍA

El trabajo descrito en este documento implementa una metodología de investigación exploratoria, basada en la búsqueda de información documental sobre el proceso a seguir para la preparación de los datos dado un corpus en el idioma japonés usando técnicas de procesamiento de lenguaje natural.

La investigación está basada en la documentación encontrada relacionada al tema de corrección de errores ortográficos, en el que se consulta la información que los autores de dichas investigaciones compartieron con la comunidad informática, además de algunas ideas sobre las posibles soluciones que tuvieron para resolver el problema de los errores ortográficos y con eso poder plantear una solución alternativa.

El trabajo busca realizar el proceso de investigación del proyecto, además preparar los datos que fueron entregados al estudiante, y que estos en el futuro puedan ser usados por otros estudiantes para implementar los modelos que darán solución a la corrección de los errores ortográficos. La idea es dejar los datos listos para que puedan ser usados en el futuro para el desarrollo de modelos que se encarguen de la corrección de errores ortográficos.

Para el desarrollo del trabajo hubo que usar algunas herramientas que trabajan sobre algoritmos de alineación de texto, que son comúnmente usados para el procesamiento de lenguaje natural, a continuación, se describirán algunos de los algoritmos sobre los que trabajan ciertas herramientas:

2.1 Algoritmo de Levenshtein

Conocido como Distancia de edición Levenshtein, que significa el número de diferencias entre dos palabras llamadas como distancia. La distancia Levenshtein entre dos palabras es el menor número de modificaciones de un solo carácter (inserciones, supresiones o sustituciones) necesarias para transformar una palabra en la otra. (Patel & Ravichandran)

El algoritmo de Levenshtein es una medida de distancia de edición de cadenas que cuantifica la distancia entre las pronunciaciones de las palabras correspondientes en diferentes dialectos o lenguas estrechamente relacionadas. (Beijering, Gooskens, & Heeringa, s.f.)

El algoritmo de Levenshtein consiste de dos partes, la primera consiste en formar la matriz del cruce de letras de las palabras, y luego se otorga el valor de cada celda según lo que

el algoritmo determina para cada caso y por último se usa una técnica de backtracking para avisar la operación que se debe hacer.

El algoritmo de Levenshtein calcula el número más corto de edición para una serie. Esto se hace usando técnicas de programación dinámica. Y la matriz se llena desde izquierda a derecho y de arriba hacia abajo.

En el segundo paso se debe decidir las operaciones que necesitan ser ejecutadas para hacer que las cadenas de caracteres comparadas tengan la misma forma.

El algoritmo lo que hace es revisar las tres celdas localizadas a la parte de arriba, a la izquierda y en diagonal hacia la izquierda y arriba, el valor mínimo entre las tres celdas es el valor hacia donde se moverá el cursor.

Si el valor mínimo está en la celda de la izquierda, podemos concluir que nuestra operación debe estar dispuesta como la letra corriente de borrado que se controla frente a otras. Si el valor mínimo está en una celda superior, podemos deducir que la operación necesaria para la solución adecuada es la inserción. Por último, si la celda diagonal tiene el menor valor, significa que debe incluirse una operación de sustitución en la lista de operaciones necesarias. (Patel & Ravichandran)

2.2 Diff

Es una herramienta para la comparación de textos línea a línea, también puede comparar el contenido entre directorios. Para los archivos que son idénticos, diff normalmente no produce ninguna salida; para los archivos binarios (no de texto), diff normalmente sólo informa de que son diferentes. (MacKenzie, Eggert, & Stallman, 2021).

Esta herramienta es usada en diferentes implementaciones, las usadas en este trabajo son las siguientes:

- **Wdiff:** El programa GNU wdiff es un front end de diff para comparar archivos palabra por palabra. (Pinard & Gagern, 2016)
- **VimDiff:** Esta herramienta presenta cada archivo en su propia pantalla y su característica principal es que las diferencias entre los archivos se resaltan, esto ayuda a detectar de manera más rápida las diferencias entre los archivos.

2.3 Ukkonen

Es un algoritmo de comparación aproximada de cadenas, que habla de que la distancia entre una cadena caracteres de tamaño m y otra de tamaño n es el costo mínimo de una secuencia de pasos de edición (inserciones, eliminaciones y cambios) (UKKONEN, 1985)

Es un algoritmo con complejidad O(mn), es decir que es efectivo en cadenas cortas, ya que para cadenas demasiado largas el algoritmo es ineficiente.

Dada una secuencia A y una secuencia B, el algoritmo de Ukkonen lo que hace es calcular la cantidad de operaciones de edición que se deben hacer para convertir la cadena A en la cadena B. (UKKONEN, 1985)

El algoritmo de Ukkonen recibe algún texto T de tamaño m y se le entrega una cadena desconocida S con un tamaño n. Luego, la lógica lo que propone es entregar la información sobre la ocurrencia de S en T, puede determinar si S se encuentra en T y hace la búsqueda en el caso que se encuentre. Es un algoritmo que usa técnicas de programación dinámica para recolectar la información almacenada sobre la comparación entre las cadenas.

2.4 OCR

En el trabajo desarrollado, el sistema operativo de Windows se usó para la instalación de una herramienta llamada ABBYY, dicha herramienta es un OCR que tiene 5 etapas para procesar la entrada y entregar una salida de archivos en texto plano.

Importar la imagen: ABBYY permite recibir imágenes de diferentes fuentes, los formatos de imágenes que admite el OCR son los siguientes BMP, DCX, DjVu, JBIG2, JPEG, JPEG 2000, PNG, PDF, TIFF, PCX, GIF, multi-page TIFF

Pre procesamiento de la imagen: ABBYY ejecuta una variedad de funciones para mejorar la calidad del documento para el proceso de reconocimiento. Algunas de las funciones que se ejecutan son las siguientes Escalado de imágenes, Recorte de imágenes, Recorte de imágenes, Creación de vistas previas, Rotación de imágenes (90, 180 y 270 grados) Enderezamiento de líneas, Reflejo e inversión, Eliminación de ruido, Mejora del contraste local. (ABBYY, imageProcessing, 2022)

Análisis de documentos: El paso de análisis de un documento analiza su estructura lógica. Identifica la primera y la última página del documento y reconoce elementos de formato como notas al pie, encabezados, pies de página y tablas.

Adicional, se reconoce el diseño de las páginas individuales, cada página se divide en objetos individuales, es decir, como bloques de texto, imágenes, tablas y celdas de tablas, etc. Se reconoce la orientación de la página, la extensión de la página, detectan texto vertical u horizontal y definen áreas de la página para el proceso de OCR.

Esto permite especificar las áreas de texto y campos a reconocer y qué áreas de la página (como imágenes y gráficos) deberían mantenerse originales. Al mismo tiempo, se obtiene información sobre la estructura lógica del documento (incluido el formato) que se utilizará al final del proceso de OCR cuando el documento se reconstruya con precisión.

Los resultados de este análisis se utilizan para obtener la estructura y el diseño del documento cuando se procesa para su posterior reutilización. En otras palabras, el documento debe ser reconstruido exactamente. Todas las imágenes y diagramas se conservan en su formato original y no se reconoce el texto dentro de las imágenes y logotipos. (ABBYY, ABBYY, 2022)

Reconocimiento: Para el reconocimiento ABBYY ofrece diferentes tecnologías de reconocimiento que son OCR, ICR y OBR. En el caso específico de este trabajo la tecnología sobre la que es importante trabajar es sobre el OCR. Que en ABBY ofrece soporte para los idiomas no europeos, entre los cuales se pueden ver el chino, japonés, coreano, árabe, farsi, tailandés y vietnamita.

Exportar el texto y reconstrucción de la documentación: el ABBYY tiene varias opciones para la exportar los datos, entre ellos se encuentran una variedad de extensiones, pero para el caso del trabajo a entregar es necesario que la salida de los archivos sean de tipo texto plano.

2.5 LaTex y Compiladore de LaTex

Al iniciar el trabajo, lo primero es conocer el problema sobre el que nace el proyecto y cada uno de los componentes que lo constituyen, además de cómo se podrían resolver las fases de dichos componentes. De manera más específica el presente trabajo propone para este módulo del proyecto en base al corpus en japonés entregado por el tutor de la materia, hacer el procesamiento de datos. Es decir, el trabajo consiste en investigar la información necesaria para la limpieza de datos, luego hacer dicha limpieza, pasar a la normalización y alineación de los datos. Esto debido a que es un proceso sin implementar, sobre un idioma desconocido y poco enseñado en países latinoamericanos. Es por esto que fue

necesario hacer la investigación sobre las diferencias entre los lenguajes orientales con respecto a los lenguajes occidentales y las herramientas que están disponibles para poder preparar los datos que se encuentran en el idioma japonés.

Se determina que el proceso a seguir inicia con la limpieza de datos. Es necesario mencionar que todo el modulo está desarrollado en el lenguaje de programación Perl, se eligió hacer uso del sistema operativo Linux, con la distribución Fedora 34, esto con el objetivo de instalar ciertas herramientas para el procesamiento de los datos y que esto ayude con la limpieza y normalización de los mismos. También fue necesario el uso del sistema operativo Windows 10, con la finalidad de instalar una herramienta OCR llamada ABBYY que ayudará a generar los errores en los documentos, para generar el corpus que se comparará con el corpus de referencia y dejar los datos preparados para que los futuros proyectos hacer la corrección ortográfica.

La razón de usar el lenguaje de programación Perl es porque es útil para la creación de modelos de inteligencia artificial, las expresiones regulares son fáciles de procesar y aportaba un enfoque diferente que usar el lenguaje de programación Python.

Las herramientas que se necesitaron para desarrollar este trabajo son las siguientes:

Instalar LaTex en el sistema operativo Linux, para esta parte del módulo es necesario hacer la instalación de las librerías de LaTex con soporte para el idioma japonés, en un inicio se instaló la librería con soporte para el japonés *texlive-collection-langcjk*. Luego de correr las respectivas pruebas se determinó que a pesar de tener instalada esta librería, no se mostraban todos los caracteres del idioma japonés, al parecer la librería no los soporta todos. Por lo que fue a ser necesario buscar una solución al problema, el primer intento fue instalar todo el paquete de LaTex el cual es textlive-scheme-full, pero el error persistía. Luego se agregaron fuentes de google, específicamente las fuentes de google de la familia Noto y Noto Sans, los cuales, si mostraban parte de dichos caracteres que en un principio no se mostraban al compilar el archivo LaTex, esto debido a que soportaba diferentes tipos de caracteres de UTF-8.

Luego se encontró que el problema principal sobre los caracteres que no se detectaban se encontraba en el uso del compilador de LaTex que se estaba usando, la recomendación era usar la librería CJK, la cual es compatible para el compilador pdfLaTex que fue el sugerido al inicio del proceso por el tutor. Se abordaron durante el curso varias soluciones, en el caso particular de la solución desarrollada en esta parte del componente, se optó por la búsqueda de un compilador LaTex que tuviera un soporte más completo, es así como se implementó el compilador de xeLaTex que es soportado por la librería xeCJK.

La idea de generar archivos en formato LaTex es que desde un compilador de LaTex va a ser posible convertir a pdf el archivo de entrada que recibió y posterior a ello convertir el pdf en una imagen, que será la entrada para la herramienta OCR. Hay que tener en cuenta que al momento de generar archivos latex hay caracteres especiales que usa latex como parte de los comandos, como lo son: las llaves {}, [], $, #, &, \, ~, (), etc. Esto es importante mencionarlo debido a que en el proceso se reflejó que es necesario escapar todos estos caracteres para que se puedan generar los archivos pdf, al inicio no funcionaba bien la ejecución para la creación de los archivos pdf, esto debido a que al querer compilar los archivos mostraba un error en los símbolos que no se escapaban y por ello no se generaban los archivos pdf.

En el caso de la creación del archivo LaTex era necesario agregar al nuevo archivo que llevará extensión .tex es que se debe generar una cabecera con las configuraciones que debe tener el archivo latex para poder compilar, algunas de estos atributos son: tamaño de letra del archivo, tamaño del formato del archivo, los valores del margen del documento, quitar la identación al inicio del documento, cambiar el interlineado del archivo, usar la fuente de letra, al igual que el lenguaje en que está configurado y el compilador que tendrá cada archivo LaTex.

Para la cuestión del compilador es donde hubo algunas novedades, y ahí fue necesario investigar sobre los compiladores latex para generar un pdf. Al inicio lo recomendado fue usar CJK, que son las siglas de la librería que contiene el soporte para chino, japonés y koreano, y la librería para el soporte de UTF 8. Pero al encontrarse con archivos en los que había caracteres especiales la ejecución paraba y no se podían generar los archivos.

2.6 Creación de imágenes con el comando Ghostscript (gs)

El comando GhostScript es usado en el presente trabajo para crear las imágenes, esto se hace leyendo el directorio donde se encuentran almacenados los archivos en formato pdf y convertirlo a imágenes, esto se logró con el siguiente comando:

```
gs -dQuiet -dSAFER -dNOPROMT -dMaxBitmap=500000000 -dJPEGQ=80 -dAlignToPixels=0 -dGridFitTT=2 -sDEVICE=jpeg -r400x400 -dNOPAUSE -dBATCH -sDEVICE=jpeg -dDEVICEHEIGHT=500 -dTextAlphaBits=4 -dGraphicsAlphaBits=4 -sOutputFile=$directory$file_name%03d.jpg $file -c quit
```

Y a continuación se explicarán los parámetros más importantes usados en el comando (Software, 2022):

Tabla 2.1 Parámetros más importantes usados para GhostScript

dQuiet	Suprime los mensajes normales de inicio.
dSAFER:	Activa el control de archivos.
dMaxBitmap	Usada para controlar cuando usar la lista de visualización
sDEVICE	selecciona el dispositivo de salida que Ghostscript debe utilizar.
r400x400	Resolución de la imagen
dNOPAUSE	Desactiva la consola y pausa al final de cada página.
dBATCH	Hace que Ghostscript se salga luego de procesar todos los archivos nombrados en la línea de comandos.
sOutputFile	Dirección del lugar donde se almacenarán las imágenes creadas
c quit	Es equivalente a quit.ps

Los datos son archivos de texto plano, un total de 117.1MB al inicio del proceso, en donde se decidió separar el 75% de los datos para ser usados como los datos de entrenamiento y el otro 25% como los datos de prueba que se usarán para los modelos que se desarrollarán en el futuro.

La estructura inicial del corpus entregado por el tutor se basa en que cada archivo está constituido por dos columnas, la primera son unos valores que representan los identificadores de la línea y a continuación separados por un tabulador están las líneas con el texto en japonés.

La estructura del proceso completo a seguir en el documento es la descrita a continuación:

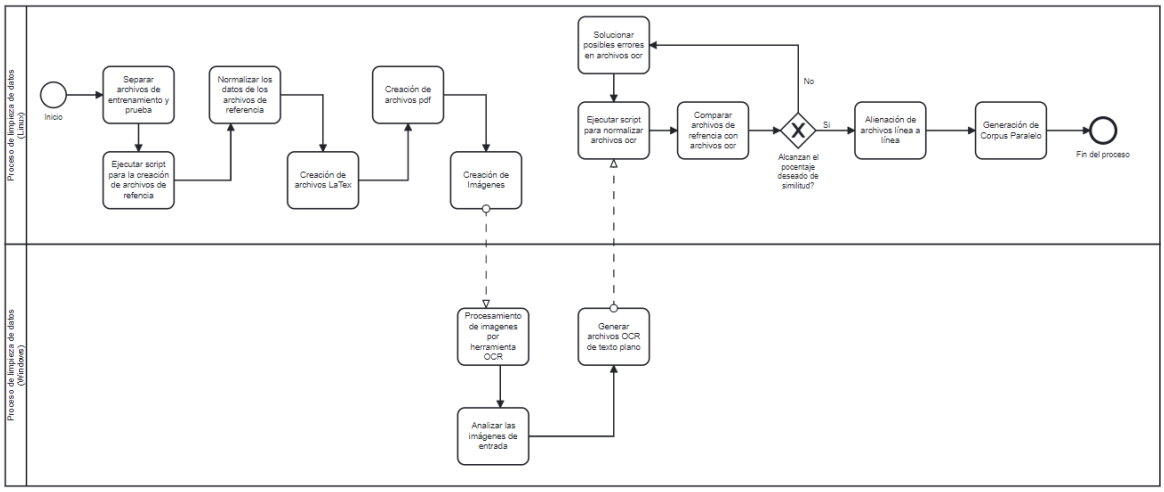


Figure 2.1 Proceso completo para preparación de datos

Para explicar esta parte de una manera más fácil de comprender, se dividirá el proceso en tres partes y en cada una habrá una descripción de cada proceso. Los procesos están separados debido a los saltos que hubo que hacer de un sistema operativo a otro.

Primera parte del proceso:

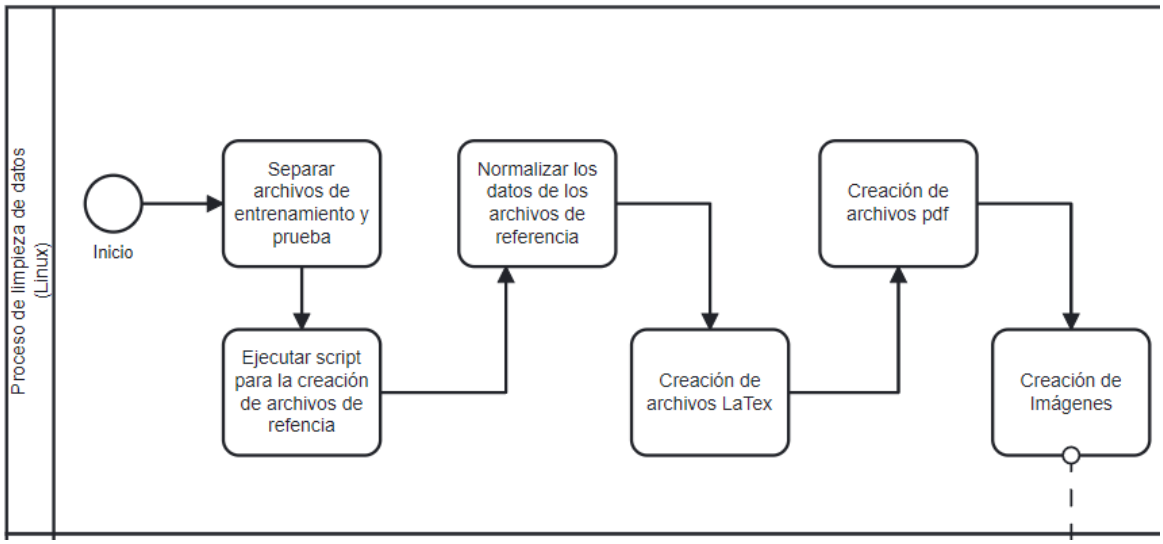


Figure 2.2 Primera parte del proceso de preparación de los datos

En esta primera parte se inicia el proceso de separación de los archivos, para esto se generó un repositorio en GitHub con la siguiente estructura de carpetas: **Trabajo_titulacion:**

01**_jp_corpus**

Test_data

Training_data

02**_LaTex_created_files**

Test_data

Training_data

03_pdf_created_files

Test_data

Training_data

05_parallel _files

Test_data

Training_data

Normilized_files

Test_data

Training_data

referenceFiles

Test_data

Training_data

ocrFiles

Test_data

Training_data

01_CreateReferenceFiiles.pl

02_NormalizeFiles.pl

03_CreateImage.p

04_move_ocrfiles.sh

05_wdiff_ref_ocr.pl

Con dicha estructura lo que se hace es ejecutar los archivos con extensión .pl en el siguiente orden:

1. CreateReferenceFiles.pl
2. NormalizeFiles.pl (para archivos de referencia)
3. CreateImage.pl
4. NormalizeFiles.pl (para archivos OCR)

En el primer script lo que hace el código es ejecutar la función *create_reference_file* y recibe como primer parámetro el directorio fuente y como segundo parámetro el directorio de destino, en este caso el script está configurado para que se ejecute esta función tanto para los datos de entrenamiento, como para los datos de prueba.

El proceso interno de dicha función es recorrer el directorio fuente en busca de todos los archivos que están allí, se lee el archivo fuente y se toma el texto línea por línea y se procede a quitar la primera columna de cada línea del corpus y el tabulador, para así solo dejar el texto en japonés del archivo, también se buscan los espacios en blanco que existen en el documento y se eliminan, posterior a esto, se procede a insertar un salto de línea en el final de cada línea del archivo de referencia y también se eliminan las líneas que están repetidas en el archivo, estas líneas repetidas son las líneas que se encuentren una seguida de la otra.

Y al final se procede a guardar estos cambios en el nuevo archivo, en el directorio de destino con el nombre de: “archivo_xxx_out.ref”, en donde las xxx son los números correspondiente a cada archivo y se guardan con la extensión “.ref” debido a que se necesitará luego diferenciar cada archivo de referencia contra su respectivo archivo OCR.

Cuando se ha terminado esta parte del proceso, fue necesario ejecutar el script normalize.pl para los archivos de referencia creados, en este punto el objetivo de normalizar los datos es reemplazar algunos símbolos que no logran ser detectados por el compilador de LaTex. El proceso de la normalización se detallará más adelante en el texto.

Una vez que el corpus de referencia ha sido normalizado, se almacenan los archivos en el directorio “./Normilized_files”. Posterior a esto se crean debe ejecutar el script CreateImage.pl

Este script inicia el proceso ejecutando la función *write_latex* Esto lo que hace es que recibe los paths de los directorios fuente y destino, almacena los archivos que se encuentran en dicho directorio dentro de un array, luego recorre los archivos y almacena todo el contenido dentro de una variable llamada “*file_content”* luego se inserta al final de cada línea los símbolos “\\\\”. En latex el salto de línea se debe hacer con dos barras invertidas, pero como es un carácter especial, por lo que es necesario escaparlo y ese proceso se hace usando de igual manera las barras invertidas, es por esto que se agregan cuatro barras invertidas.

También se reemplazan ciertos caracteres especiales como el corchete que abre “ [“ y el que cierra “] “.

Y luego se genera el archivo destino que será el archivo con la extensión “.tex” que corresponde a los archivos LaTex.

Los archivos LaTex se componen de tres partes importantes, la cabecera del archivo, el cuerpo del archivo, y el cierre del archivo. Para la cabecera fue necesario agregar las librerías que iban a permitir que el compilador de LaTex detectara el idioma japonés.

Tabla 2.2 Headers for LaTex files

documentclass[10pt]{extreport}
enewcommand{\\baselinestretch}{2}
usepackage{geometry}
usepackage[T1]{fontenc}
usepackage[utf8x]{inputenc}
geometry{a4paper,left=08mm,top=10mm,right=08mm,bottom=10mm}
usepackage{xeCJK}
setmainfont{Noto Serif} setCJKmainfont{Noto Serif CJK KR} setCJKsansfont{Noto Sans CJK KR} setCJKmonofont{Noto Sans Mono CJK KR}
begin{document}

setlength{\\parindent}{0cm}
pagenumbering{gobble}

Una vez creado el archivo LaTex la siguiente función que se ejecuta es *“write_pdf”* , lo que esta función hace es recibir como parámetro el directorio donde se encuentran ubicados los archivos LaTex y empieza a procesar uno a uno los archivos, la ejecución que se hace dentro de esta función es la de llamar al compilador de LaTex que se usa para este proceso se ejecuta el comando xelatex –output-directory=”DIRECTORIO_DESTINO” archivo_latex

Y adicional se ejecuta también un comando para eliminar unos archivos adicionales que se crean junto con el archivo pdf, estos archivos tienen la extensión “.aux” y “.log”.

Por último, en el script se ejecuta la función “*create_image”*. Aquí se toma como entrada el directorio en el que se encuentran almacenados los archivos pdf, lee el directorio, toma uno a uno los archivos y se ejecuta el comando de la herramienta Ghostscript descrito anteriormente en el documento. Aquí es válido mencionar que fue necesario subir la calidad de la imagen, por defecto el comando ghostscript al procesar los archivos pdf y convertirlos en imagen, estás se generan con una calidad de 70/100, en el trabajo se subió a 80/100.

La característica de este comando es que toma un archivo formato pdf y cada hoja del archivo la convierte en una página, por lo que al momento de generar la salida se determinó que se van a crear por cada archivo un directorio contenedor de las imágenes correspondientes a cada archivo.

La ubicación de las imágenes generadas se almacenó dentro de una carpeta compartida entre la máquina virtual de Linux y la maquina local de Windows. Esto debido a que las imágenes sirven como entrada para la herramienta OCR.

Segunda parte del proceso:

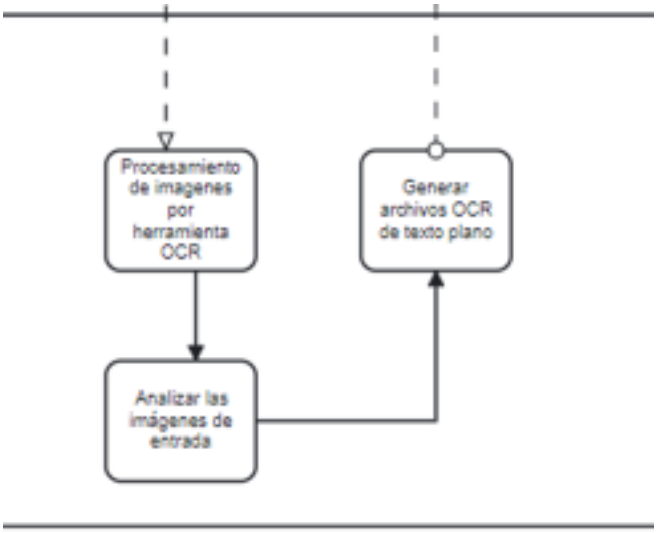


Figura 2.3 Segundo subproceso

Como se mencionó anteriormente, la segunda parte del proceso requiere de hacer uso de la maquina local de Windows, esto debido a que para ese sistema operativo hay disponible una herramienta OCR, llamada ABBYY. Dicha herramienta utiliza como entrada las imágenes generadas.

Dentro de ABBYY OCR editor, es posible generar ciertos procesos para la detección del contenido en japonés de las imágenes.

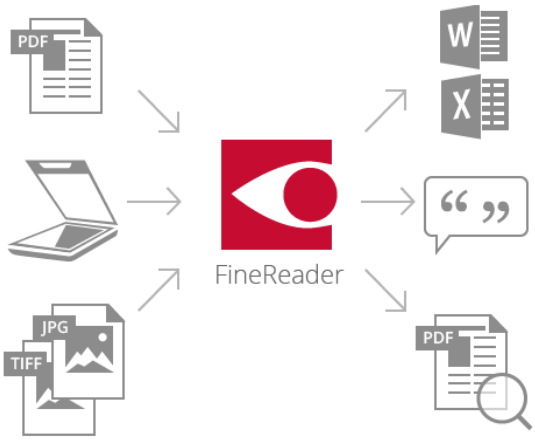


Figura 2.4 Flujo de ABBYY OCR

La tiene la opción de crear tareas automatizadas, es decir, se pueden generar una serie de pasos que se van a ejecutar automáticamente, esto sirve de ayuda para el procesamiento de las imágenes, debido a que existen muchos archivos con gran tamaño de información, por lo que al generar las imágenes estos directorios generar muchos archivos que procesar.

Actualmente dentro de la herramienta ABBYY, se creó una tarea automática que consta de cinco pasos que la herramienta va a seguir para generar como salida un archivo de texto plano en el idioma japonés.

Los cinco pasos son:

- Crear un nuevo proyecto OCR: Se determina el idioma que va a reconocer el OCR, a parte, esta sección permite configurar ciertas características para que se generen los archivos OCR.
- Abrir la imagen o pdf: Aquí se determina el directorio desde el que el archivo va a ser cargado.
- Analizar el texto de las imágenes: Aquí es el trabajo que tiene la herramienta, para este punto se utiliza un template previamente creado, que es la selección del área que la herramienta va a identificar.
- OCR: En este punto es donde el programa está ejecutando la lógica que permite identificar el texto.
- Guardar la salida en un archivo de texto plano: Al terminar el proceso la herramienta genera un archivo de texto plano, que se almacena dentro de un directorio llamado OCRs.

Este proceso descrito se debe hacer para cada uno de los archivos dentro del directorio de imágenes, este proceso de seleccionar cada uno de los archivos que se deben ejecutar es exhaustiva debido a que no se logró encontrar alguna manera de pasarle un directorio y que lo hiciera directamente para todos los archivos dentro de un directorio y luego generar un archivo de texto plano para cada uno de los directorios con imágenes de los documentos.

Cuando todos los archivos OCRs han sido creados, se inicia el siguiente subproceso del trabajo de titulación:

Tercera parte del proceso:

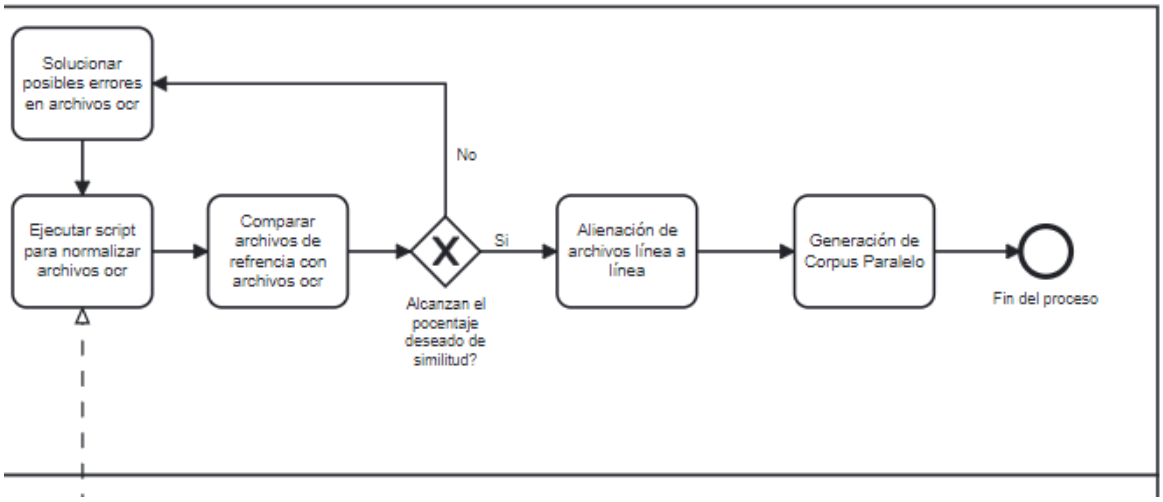


Figura 2.5 subproceso de normalización de datos

Aquí es necesario ejecutar el script normalize.pl para los archivos OCR, una vez que se han normalizado los archivos OCR, es necesario hacer una comparación entre los archivos de referencia y OCR, para ello se hace uso de la herramienta diff, usando específicamente el comando wdiff, que como se explicó antes, nos sirve como una interfaz para poder detectar las partes del texto donde hay diferencia.

El comando que se usa para esto es:

wdiff -s $ref_file_name $ocr_file_name --no-common

Esto con el fin de poder identificar el porcentaje de similitud entre los dos archivos, el OCR debe arrojar un valor mayor al 90%.

Cuando la comparación entre los dos archivos no alcanza el porcentaje esperado, hacemos uso de la herramienta vimdiff con el siguiente comando:

vimdiff $ref_file_name $ocr_file_name

Este comando permite revisar de manera más amigable las diferencias entre los archivos, un ejemplo de estos se encuentra en la figura 3.1.10, en la sección de resultados.

Con la revisión se puede distinguir específicamente la línea e incluso las secciones donde los caracteres no coinciden.

En todo el trabajo se ha mencionado que los cambios posibles que se presentarán dentro del documento pueden ser tres:

- Eliminación de un carácter o símbolo
- Añadir un carácter o símbolo
- Reemplazo de un carácter o símbolo por otro

Lo que se hace es identificar qué tipo de cambios es el que sufrieron las líneas que presentan una diferencia, y así poder decidir qué cambios serán necesarios en el script de normalización.

Por ejemplo, en el proceso de normalización se decidió cambiar todos números y letras que no pertenecen al japonés como los siguientes: “0123456789” que se cambiaron por “０１２３４５６７８９”, que son las representaciones numéricas en el japonés.

Aunque las letras o los números mostrados representan lo mismo, se identifican como diferentes valores, por lo que no es correcto mezclar los símbolos del lenguaje en japonés con los usados en lenguajes como el español. La decisión es pasar los caracteres que no correspondan al lenguaje japonés a sus respectivos homólogos en dicho lenguaje. Es así que se presenta la siguiente tabla con los homólogos usados en este trabajo de titulación

Tabla 2.3 Normalización de caracteres

Español	Japonés
0123456789	０１２３４５６７８９
ABCDEFGHIJKLMNOPQRSTUVWXYZ	ＡＢＣＤＥＦＧＨＩＪＫＬＭＮＯＰＱＲＳＴＵＶＷＸＹＺ
~～～	～
" ‟″ ˝«»	″
`	’
‚、 ’	’
[〖 〔	「
]〗 }	」
(《	（
)》	）
?	？

Es proceso es cíclico, y es necesario hacerlo varias veces, hasta lograr completar en la mayoría de los archivos el porcentaje esperado de similutid.

Para obtener el valor del porcentaje de similitud real de los archivos, hay que hacer un promedio ponderado, esto debido a que los datos están separados en archivos, pero dichos archivos no contienen valores similares en MB, así como den Número de Líneas.

Es por esto que se hace el siguiente cálculo.

$$\text{PP} = \sum(\frac{NL}{TL}) * PS$$

Ecuación 2.1. Formula promedio ponderado

- **Donde PP es el Promedio Ponderado**
- **NL es el número de líneas del archivo**
- **TL es el total de líneas en el repositorio**
- **PS Porcentaje de Similitud salido del comando wdiff**

Está formula aplica para todos los archivos dentro de la carpeta donde se encuentran almacenados los datos.

Con esto completo, el siguiente paso es la alineación de los textos a nivel de líneas, para lo cual nos ayudamos con la herramienta wdiff para recibir los dos archivos, un nos entregue una salida con las comparaciones de los archivos. La salida de la herramienta será el archivo OCR alineado con respecto al archivo referencial correspondiente, este proceso completa la generación del corpus paralelo que se busca como objetivo, pero la alineación no se logró al 100% debido a que el archivo OCR que resulta al pasar por la herramienta ABBY en ciertas cadenas de texto reemplaza los saltos de línea por un espacio en blanco, y en algunos casos donde encuentra diferencia añade un salto de línea. Se intentó hallar una manera para poder restablecer los saltos de línea perdidos porque son los que representan el mayor problema en la alineación.

Esto es debido a que no se puede calcular el número de saltos de línea que existen en un bloque de texto del texto de referencia y al c

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

Como se esperaba se logró hacer la limpieza del corpus recibido por parte del tutor, lo que nos permitió generar archivos únicamente con texto en el idioma japonés.

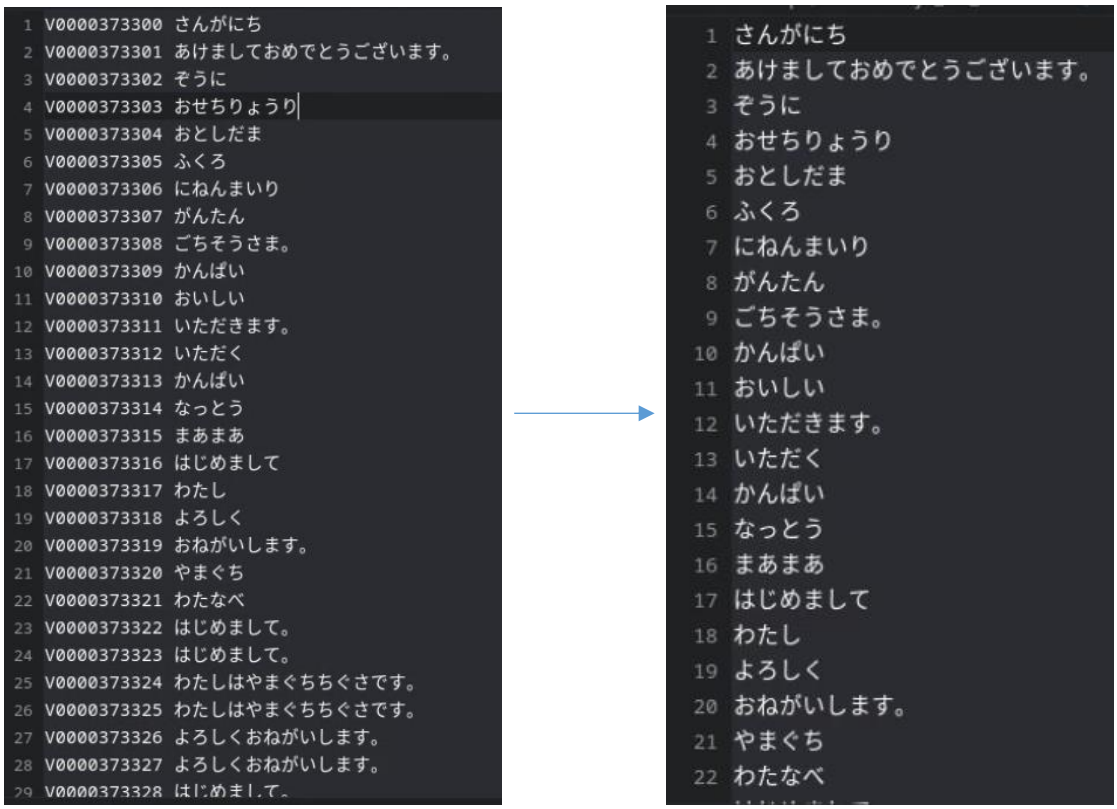


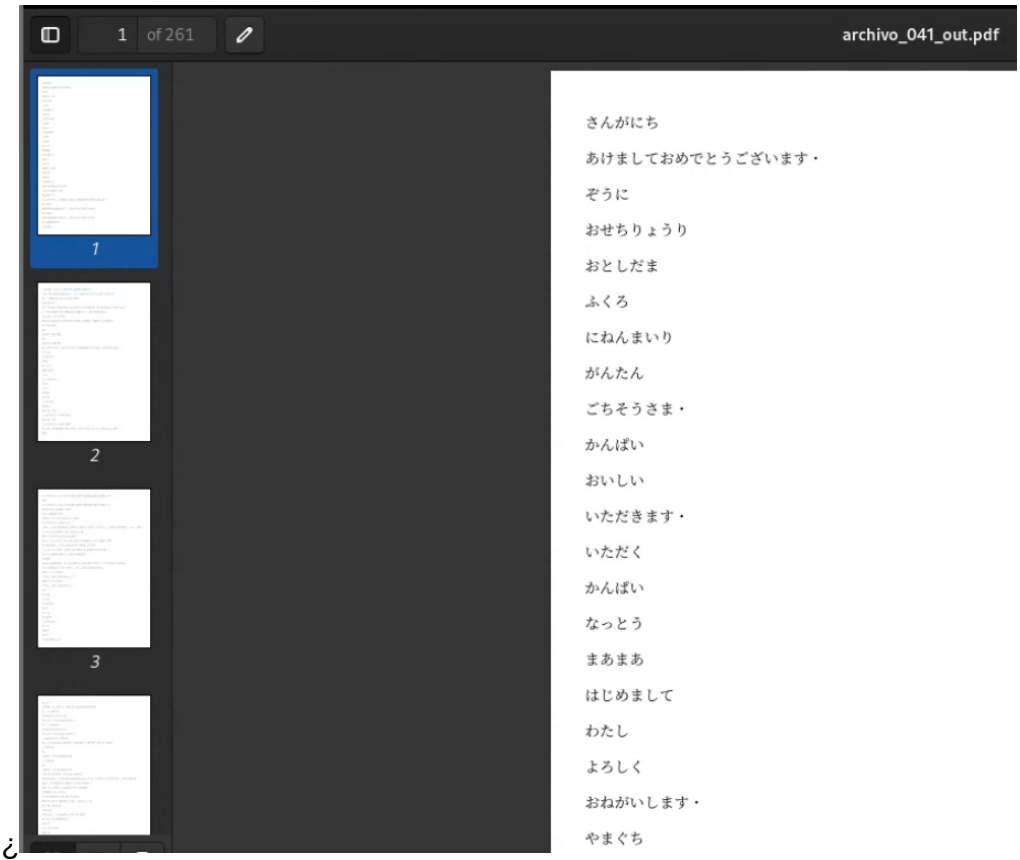
Figura 3.1.1 Limpieza de datos para creación de archivo_041_out de referencia

También se logró generar los archivos latex por medio de las librerías usadas y en el ambiente Linux, luego de haber normalizado el corpus de referencia generado previamente.

```latex
\documentclass[10pt]{extreport}
\renewcommand{\baselinestretch}{2}
\usepackage{geometry}
\usepackage[T1]{fontenc}
\usepackage[utf8x]{inputenc}
\geometry{a4paper,left=08mm,top=10mm,right=08mm,bottom=10mm}
\usepackage{xeCJK}
\setmainfont{Noto Serif}
\setCJKmainfont{Noto Serif CJK KR}
\setCJKsansfont{Noto Sans CJK KR}
\setCJKmonofont{Noto Sans Mono CJK KR}
\begin{document}
\setlength{\parindent}{0cm}
\pagenumbering{gobble}
さんがにち\\
あけましておめでとうございます・\\
ぞうに\\
おせちりょうり\\
おとしだま\\
ふくろ\\
にねんまいり\\
がんたん\\
ごちそうさま・\\
かんぱい\\
```

Figura 3.1.2 Archivo_041_out LaTeX generado

También se logró generar los archivos pdf por medio del compilador de latex usado y descrito previamente en la sección anterior. Todos los archivos fueron generados de manera exitosa y puestos en el respectivo directorio según el tipo de datos que son, es decir, datos de entrenamiento o datos de prueba.



¿

Figura 3.1.3 Archivo_041_out en formato pdf

Las imágenes generadas se almacenaron de manera exitosa por directorio, según el nombre del archivo al que corresponden, las imágenes almacenadas abarcan un gran espacio de memoria, es por esto que no se compartirán las imágenes generadas, pero se podrían generar en caso de necesitarlo, ejecutando el script para la generación de imágenes.

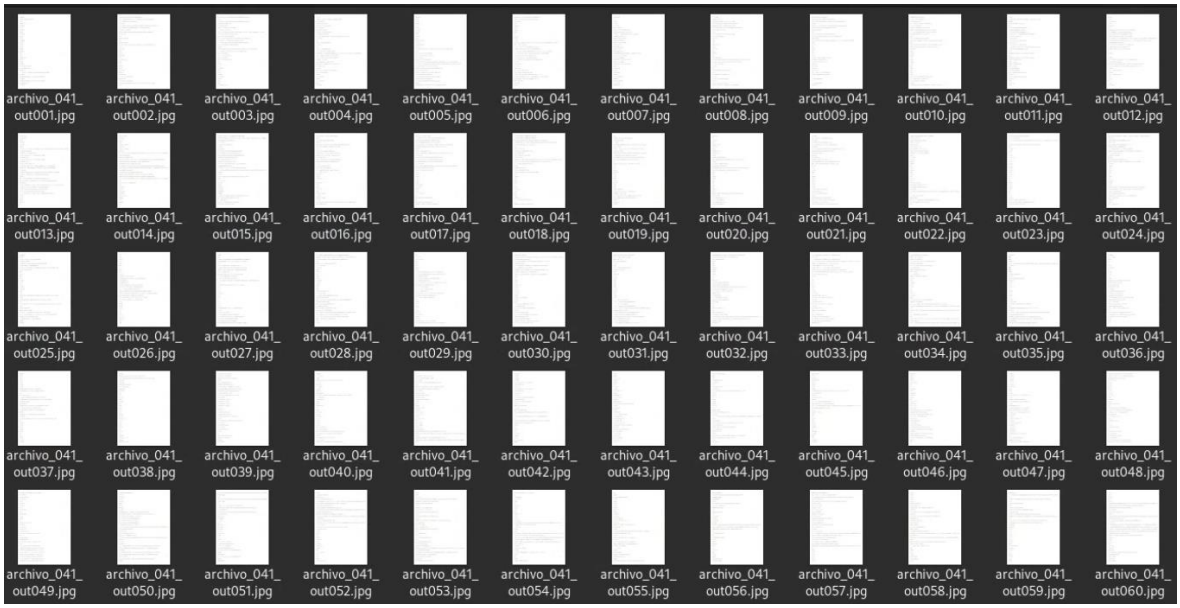


Figura 3.1.4 archivo_041_out en imágenes

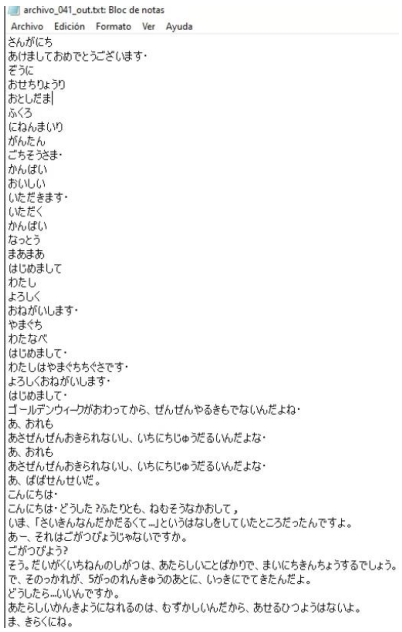

archivo_041_out.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
さんがにち
あけましておめでとうございます・
そうに
おせちりょうり
おとしだま
ふくろ
にねんまいり
がんたん
ごちそうさま・
かんぱい
おいしい
いただきます・
いただく
かんぱい
なっとう
まあまあ
はじめまして
わたし
よろしく
おねがいします・
やまぐち
わたなべ
はじめまして・
わたしはやまぐちちぐさです・
よろしくおねがいします・
はじめまして・
ゴールデンウィークがおわってから、ぜんぜんやるきもでないんだよね・
あ、おれも
あさぜんぜんおきられないし、いちにちじゅうだるいんだよな・
あ、おれも
あさぜんぜんおきられないし、いちにちじゅうだるいんだよな・
あ、ばばせんせいだ。
こんにちは・
こんにちは・どうした？ふたりとも、ねむそうなかおして，
いま、「さいきんなんだかだるくて…」というはなしをしていたところだったんですよ。
あー、それはごがつびょうじゃないですか。
ごがつびょう？
そう。だいがくいちねんのしがつは、あたらしいことばかりで、まいにちきんちょうするでしょう。
で、そのっかれが、5がつのれんきゅうのあとに、いっきにでてきたんだよ。
どうしたら…いいんですか。
あたらしいかんきょうになれるのは、むずかしいんだから、あせるひつようはないよ。
ま、きらくにね。

Figura 3.1.5 Archivo OCR del Archivo_041_out

A continuación, se mostrará los resultados obtenidos en el proceso de normalización de archivos, como se puede observar en las primeras instancias, el porcentaje alcanzado es bastante bajo para el ejemplo del archivo que hemos seleccionado, pero luego, al agregar reglas en la normalización de los datos se va incrementando dicho valor de similitud entre los archivos comparados.

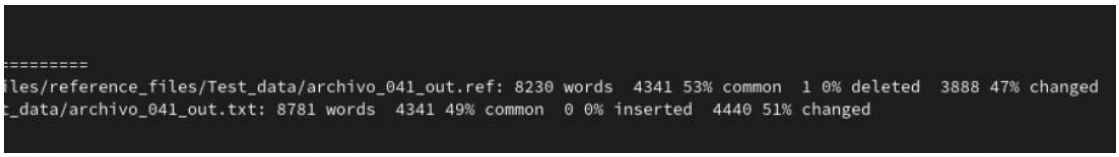


Figura 3.1.6 Porcentaje de similitud entre archivos sin normalizar

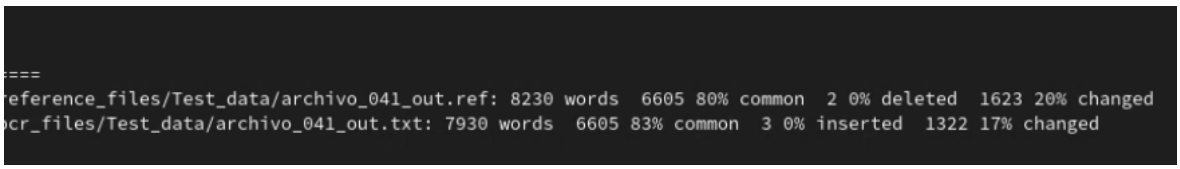


Figura 3.1.7 Porcentaje de similitud entre archivos al normalizar archivos

A continuación, se muestran algunos de los porcentajes alcanzados para la data de prueba, como se puede observar el valor alcanzado no es del 90% pero es un valor cercano.

	N de líneas	Porcentaje Peso	Porcentaje Similitud	Promedio
41	7930	0.020171496	90	0.01815435
42	7866	0.020008699	85	0.01700739
43	7763	0.019746699	90	0.01777203
44	4886	0.01242849	95	0.01180707
45	13701	0.034851156	99	0.03450264
46	7580	0.019281203	91	0.01754589
47	6868	0.017470093	90	0.01572308
48	10732	0.027298927	80	0.02183914
49	10732	0.027298927	85	0.02320409
50	12432	0.031623208	88	0.02782842
51	6495	0.016521295	90	0.01486917
52	11166	0.028402891	85	0.02414246
53	7950	0.02022237	90	0.01820013
54	7136	0.018151803	95	0.01724421

Figura 3.1.8. Cálculo de porcentaje de similitud real 1

258	388	0.000986953	80	0.00078956
259	388	0.000986953	85	0.00083891
260	66	0.000167884	88	0.00014774
261	38	9.66604E-05	90	8.6994E-05
262	30	7.63108E-05	85	6.4864E-05
263	81	0.000206039	90	0.00018544
264	33	8.39419E-05	95	7.9745E-05
265	46	0.00011701	99	0.00011584
266	30	7.63108E-05	91	6.9443E-05
267	1	2.54369E-06	90	2.2893E-06
268	24	6.10487E-05	80	4.8839E-05
269	0	0	85	0
270	1	2.54369E-06	75	1.9078E-06
Total	**393129**			**0.89189014**

Figura 3.1.9. Cálculo de porcentaje de similitud real 2

La alineación final de los archivos muestra como la gran mayoría de las líneas están de los archivos de referencia a la izquierda están alineadas con su correspondiente línea a los archivos OCR que se encuentran a la derecha de la imagen.

Como se puede observar, el color morado muestra la línea en la que hay una diferencia y el color naranja muestra en qué parte de la línea es donde existe la diferencia.

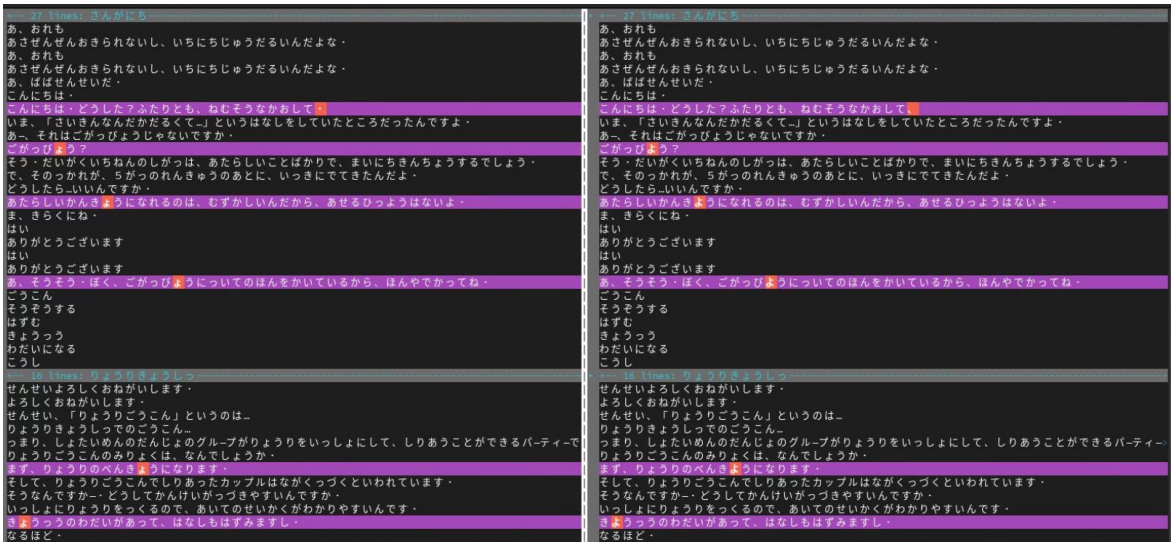


Figura 3.1.11 Ejemplo del corpus paralelo obtenido

3.2 Conclusiones

En el presente trabajo se explicó el proceso para alistar los datos recibidos en un corpus del idioma japonés, la creación de errores por medio de una herramienta OCR, la normalización y la alineación del corpus paralelo a nivel de líneas.

Además, se intentó alinear a nivel de tokens las cadenas de texto, pero no fue posible debido a que el algoritmo de ukkonen no usa gramática sujeta al contexto, y a parte, hace la comparación token por token y esto genera un problema al procesar los tokens de las líneas que tiene una relación diferente a la relación “uno a uno”.

Se concluye que las herramientas usadas para generar los errores OCR, como la herramienta ABBYY son bastante útiles, pues permiten la lectura de textos en diferentes idiomas y también por la facilidad del manejo de archivos soporta.

Las herramientas como diff o sus derivados como wdiff y vimdiff son una excelente forma de poder comparar los textos.

No fue posible avanzar en el proceso de alineación de token a token porque la opción de algoritmo de ukkonen que se había propuesto no logra satisfacer la necesidad del problema, debido a las diferentes relaciones que pueden existir entre las cadenas homologas del corpus de referencia y del corpus OCR.

En el trabajo de titulación se investigó el uso del algoritmo de Comparación aproximada de cadenas de Ukkonen, con el fin de alinear a nivel de token las líneas de cada uno de los archivos de referencia y OCR.

Aquí encontramos algunos problemas, esto debido a que la alineación de las cadenas con dicho algoritmo se hace token a token, además que es un algoritmo que trabaja libre de contexto, es decir, que no conoce cuál es el contexto y esto no permite que pueda existir una buena relación entre las cadenas que tienen errores y las cadenas candidatas a solución. Por lo que se planteó usar la herramienta Giza, que al igual que el algoritmo de Ukkonen es una herramienta libre de contexto, pero que sí permite cubrir el caso de muchos a uno, pero no al revés. Por esta razón la herramienta Giza no funciona para la alineación de token a token.

Por último, no se obtuvo el resultado esperado de alineación, a pesar de estar muy cerca de conseguir completamente la alineación línea a línea entre los archivos.

El proyecto de titulación tenía un alcance distinto al iniciar el curso, este proponía la creación de un modelo probabilístico y uno modelo de redes neuronales que pudieran

hacer la corrección ortográfica, ambos modelos entrenados con los datos del corpus paralelo. Pero no fue posible realizar el alcance propuesto debido a que la extensión del problema era demasiado grande para el tiempo límite para realizar el trabajo, por esto se decidió hacer el recorte del proyecto a la mitad, es decir, hasta crear el modelo probabilístico. Pero tuvimos que hacer un reajuste al alcance nuevamente debido a que el cambio de carrera que sufrimos nos obligó a tomar siete materias adicionales en los últimos dos semestres, entre los cuales con la cantidad de 90 créditos aproximadamente tuvimos que tomar la materia de DISEÑO DE TRABAJO DE INTEGRACIÓN CURRICULAR, al presente semestre de acuerdo al reglamento institucional, debemos tomar obligatoriamente la materia de TRABAJO DE INTEGRACIÓN CURRICULAR a la par que tomamos 5 materias más que demandan demasiado tiempo para poder cumplir con todos los objetivos planeados, es por esto que de acuerdo a lo acordado con el tutor de la materia, y con la sub decana de la facultad de ingeniería en sistemas se determinó que el alcance llegará hasta la preparación de los datos, para que los posteriores estudiantes puedan usarlos y preparar los modelos probabilísticos y de red neuronal.

3.3 Recomendaciones

Se recomienda para los futuros trabajos usar alguna técnica de procesamiento de lenguaje natural que permita el uso de la gramática sujeta al contexto, para que la relación entre las cadenas comparadas pueda soportar la relación que existe, no solo de token a token, sino que cubra todos los casos que son:

- Uno a uno
- Uno a muchos
- Muchos a uno
- Muchos a muchos

Para esto se recomienda el uso de técnicas de inteligencia artificial como, por ejemplo, los mecanismos de atención. Uno que podría ayudar a solucionar es el mecanismo de atención de Bahdanau. Esto debido a que este algoritmo si permite cubrir todas las relaciones que existirán entre algunas cadenas del corpus referencial y el corpus OCR.

El algoritmo se basa en usar redes neuronales recurrentes bidireccionales como un codificador y decodificador. Es usado muchas veces para la traducción de textos. Una de las deficiencias de las RNN normales es que sólo utilizan el contexto anterior. La BiRNN puede entrenarse utilizando toda la información de entrada disponible en el pasado y el futuro de un marco temporal específico. (Niu, Zhong, & Yu, 2021)

4 REFERENCIAS BIBLIOGRÁFICAS

ABBYY. (2022). *ABBYY.* Obtenido de ABBYY: https://www.abbyy.com/ocr-sdk/how-it-works/document-analysis/

ABBYY. (2022). *imageProcessing.* Obtenido de https://www.abbyy.com/ocr-sdk/features/image-processing/

Beijering, K., Gooskens, C., & Heeringa, W. (s.f.). *wjheeringa.nl*. Obtenido de wjheeringa.nl: http://www.wjheeringa.nl/papers/lin08.pdf

Carvajal, L. (2006). *Metodología de la Investgación Científica. Curso general y aplicado* (28 ed.). Santiago de Cali: U.S.C.

Kukich., K. (1 de Diciembre de 1992). *arxiv.org*. Recuperado el 25 de 08 de 2022, de https://doi.org/10.1145/146370.146380

MacKenzie, D., Eggert, P., & Stallman, R. (02 de 01 de 2021). *gnu.org*. Obtenido de gnu.org: https://www.gnu.org/software/diffutils/

Nagata, M. (s.f.). *https://aclanthology.org.* Obtenido de Context-Based Spelling Correction for Japanese OCR: https://aclanthology.org/C96-2136.pdf

Niu, Z., Zhong, G., & Yu, H. (04 de abril de 2021). *Neurocomputing*. Obtenido de Neurocomputing: https://www.sciencedirect.com/science/article/abs/pii/S092523122100477X?via%3Dihub

Patel, H., & Ravichandran, G. (s.f.).

Pinard, F., & Gagern, M. v. (11 de 06 de 2016). *gnu.org*. Obtenido de gnu.org: https://www.gnu.org/software/wdiff/

Rabiner, L. (Febrero de 1989). ieee.org. *ieee*, 257-286. Obtenido de ieee.org.

Software, A. (29 de marzo de 2022). *ghostscript.com*. Obtenido de ghostscript.com: https://www.ghostscript.com/doc/current/Use.htm

Sutton, C., & McCallum, A. (2012). *nowpublishers.* Obtenido de nowpublishers: https://www.nowpublishers.com/article/Details/MAL-013

UKKONEN, E. (1985). *Algorithms for Approximate String Matching*. Obtenido de Algorithms for Approximate String Matching: https://www.sciencedirect.com/science/article/pii/S0019995885800462

Vladimir, I. (s.f.). *https://ivanvladimir.gitlab.io*. Obtenido de https://ivanvladimir.gitlab.io: https://ivanvladimir.gitlab.io/lfya_book/docs/06dependedelcontexto/07gramáticadependientedelcontexto/

West, J. (07 de 03 de 2014). *Github*. Obtenido de Github: https://github.com/jordwest/mecab-docs-en