ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

ELABORACIÓN, EVALUACIÓN Y OPTIMIZACIÓN DE MODELOS OCULTOS DE MARKOV PARA LA TRANSLITERACIÓN DE PALABRAS DEL INGLÉS A KATAKANA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

JUAN JOSÉ MORALES BANDA

juan.morales01@epn.edu.ec

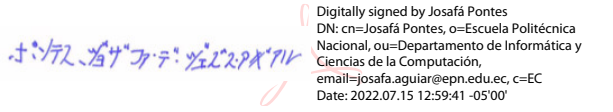
DIRECTOR: PhD. JOSAFÁ PONTES

josafa.aguiar@epn.edu.ec

QUITO, JULIO 2022

AVAL DEL DIRECTOR

Certifico que el presente trabajo fue realizado por Juan José Morales Banda bajo mi supervisión.



PhD. JOSAFÁ DE JESÚS AGUIAR PONTES

Director del Trabajo de Titulación

DECLARACIÓN DE AUTORIA

Yo JUAN JOSÉ MORALES BANDA, declaro bajo juramento que el trabajo 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.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

JUAN JOSÉ MORALES BANDA

AGRADECIMIENTO

Agradezco a mis padres, a mi hermano y a mi familia en general, por siempre apoyarme a lo largo de mi vida y por formarme en quien soy ahora. Es gracias a ellos que he podido salir adelante en cada nuevo reto que se me presenta. Mención especial a mi perrito Rufo II, quien me acompañó durante casi 17 años.

Gracias a la Escuela Politécnica Nacional, por haberme instruido con docentes de gran nivel profesional y calidad humana. Agradecimiento especial a mi director de tesis, quien me ha brindado su guía y cooperación total durante el desarrollo de este trabajo.

A todas mis amistades, pues el camino recorrido no hubiese sido el mismo de no haber contado con ellos.

Tabla de contenido

INDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto integrador propone la utilización de Modelos Ocultos de Markov (HMM) para realizar la transliteración de palabras provenientes del inglés hacia el silabario katakana de escritura japonés. Dicha transliteración será la secuencia de caracteres katakana más probable, dada una secuencia de observaciones de sílabas fonéticas del inglés. Estas sílabas fonéticas han sido obtenidas utilizando las palabras en inglés del diccionario EDICT2 dentro de las conversiones *Grapheme-to-Phoneme* (G2P) de *Festival*. Adicionalmente, el lenguaje de programación PERL y el uso de expresiones regulares permitirán la programación de los HMM.

Antes de elaborar los HMM es necesario poseer una alineación entre las sílabas fonéticas con sus respectivas secuencias katakana, la herramienta MGIZA es la encargada de realizar dicha alineación. Tras ello, la generación de las probabilidades de inicio, transición y emisión, pertenecientes a los HMM, se realiza de manera probabilística por medio de conteos en función de la alineación antes mencionada. Estas probabilidades serán las que permiten encontrar la cadena de caracteres katakana más probable por medio del algoritmo de Viterbi. A través de este proceso, se podrán evaluar y optimizar los HMM para obtener transliteraciones aceptables.

Debido a restricciones, los resultados obtenidos se comparan parcialmente con otras herramientas ya existentes que también realicen la transliteración del inglés al katakana. Esto otorga una idea de cuán satisfactorios fueron, para así evaluar la eficacia del enfoque de los HMM basados en sílabas fonéticas.

Palabras clave: Modelos Ocultos de Markov, HMM G2P, MGIZA, PERL, EDICT2, Katakana, transliteración, decodificador Viterbi.

ABSTRACT

This integrative Project proposes the use of Hidden Markov Models (HMM) to carry out the transliteration of English words into the Japanese writing katakana syllabary. The transliteration will be the most probable sequence of katakana characters, given a sequence of observations of phonetic English syllables. These phonetic syllables have been obtained using the English words from the EDICT2 dictionary within Festival Grapheme-to-Phoneme (G2P) conversions. Additionally, the PERL programming language and the use of regular expressions will allow the programming of HMMs.

To elaborate the HMM, first is necessary to have and alignment between the phonetic syllables with their respective katakana sequences, the MGIZA tool is responsible to carry out such an alignment. After that, the generation of the initial vector, transition and emission probabilities, belonging to the HMMs, are carried out probabilistically using counts based on the aforementioned alignment. These probabilities will allow to find the most probable sequence of katakana characters using the Viterbi algorithm. Through this process, HMMs can be evaluated and optimized for acceptable transliterations.

Due to restrictions, the obtained results are partially compared with other existing tools that also perform the transliteration from English to katakana. This gives an idea of how satisfactory were in order to assess the effectiveness of the phonetic HMM syllable-based approach.

Keywords: Hidden Markov Models, HMM, G2P, MGIZA, PERL, EDICT2, Katakana, transliteration, Viterbi decoder.

1. INTRODUCCIÓN

1.1. Problema

El japonés es un lenguaje complejo ya que posee tres sistemas de escritura: Kanji, Hiragana y Katakana, en una misma oración es posible utilizar sus tres sistemas e incluso no se emplean espacios en blanco para diferenciar palabras [1] [2] cómo sí ocurren en el inglés y el español. El sistema de escritura Katakana es usado principalmente para palabras y nombres extranjeros, onomatopeyas, nombres científicos y a veces para enfatizar algo [3], no se trata de una traducción de palabras sino de la transliteración de caracteres, por ejemplo, la palabra Ecuador escrito en Katakana es エクアドル (E/ku/a/do/ru). Sin embargo, no siempre es sencillo identificar los caracteres correspondientes, esto se evidencia de mejor manera con la palabra smartphone cuya transliteración a Katakana es スマ―トホン(su/ma/to/ho/n).

En la actualidad existen sistemas basados en el uso de grafemas, fonemas o ambos (híbridos) para realizar la transliteración de palabras. En [4] se realizó una comparación de este tipo de sistemas y se concluyó que la transliteración híbrida es la más efectiva. Este tipo de transliteración interpola ambos modelos de transliteración por grafemas y fonemas para obtener una transliteración que tenga mayor probabilidad de correspondencia a la cadena original.

1.2. Propuesta

Con el escenario descrito anteriormente, la presente propuesta de trabajo consiste en la elaboración, evaluación y optimización de Modelos Ocultos de Markov (HMM) para realizar la transliteración de caracteres provenientes de palabras en español o inglés a caracteres de Katakana. Para lograrlo es necesario convertir las palabras a grafemas, por ejemplo, para la palabra Quito sus grafemas son <qu, i, t, o>, los cuales tienen su correspondencia en los fonemas /k/ /i/ /t/ /o/ y sus dos sílabas fonéticas son /ki/to/. Con estos datos por medio de un HMM se inferirá de manera probabilística los caracteres Katakana correspondientes. En este caso los caracteres más apropiados serían キト (ki/to) [5]. En la Figura 1 se muestra un diagrama de flujo con el proceso propuesto.

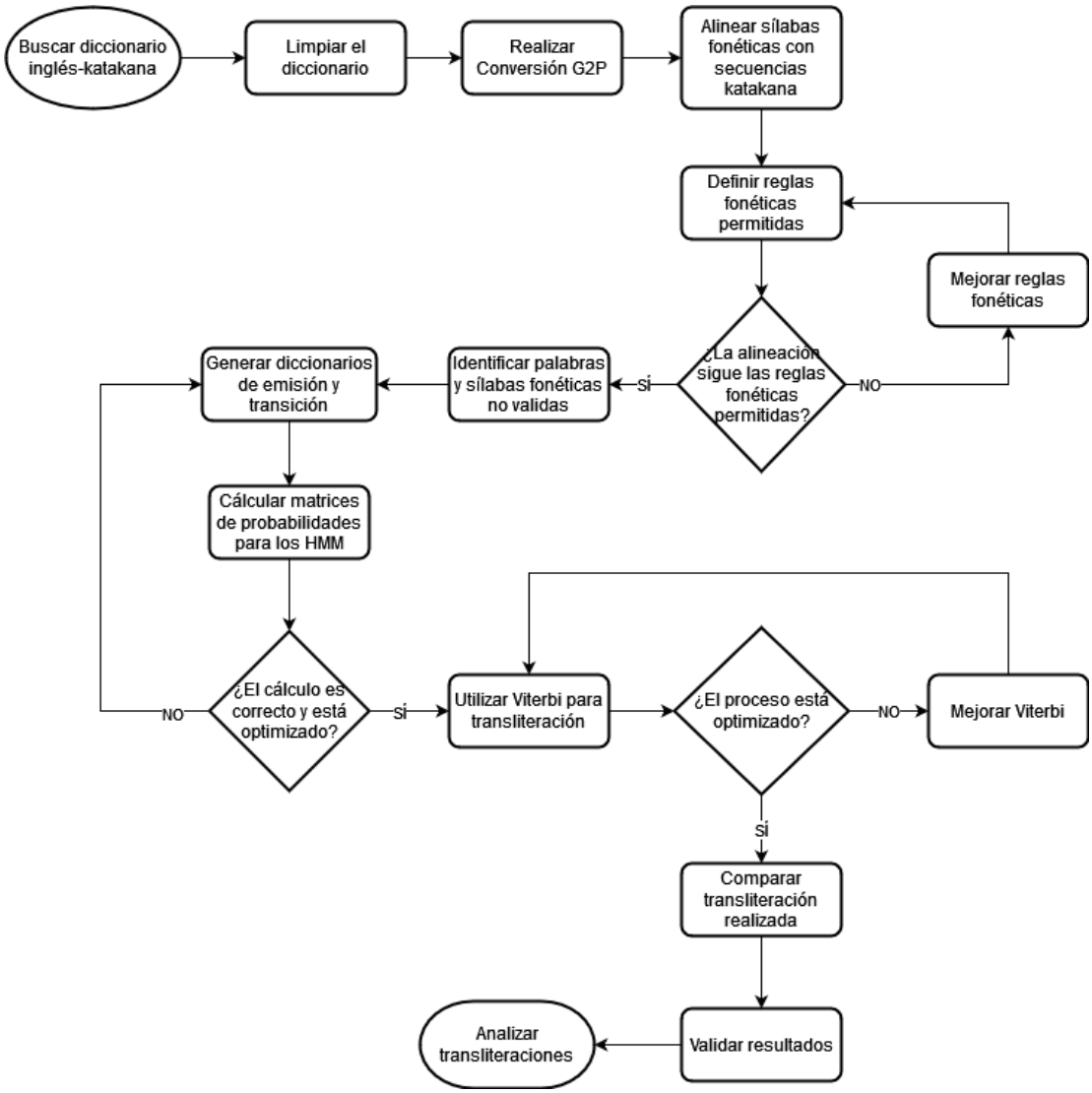


Figura 1. Diagrama de flujo de la propuesta

1.3. Objetivos

1.3.1. Objetivo General

Transliterar palabras del inglés al katakana.

1.3.2. Objetivos Específicos

- Recopilar diccionarios de inglés a katakana.
- Convertir grafemas a fonemas de palabras del idioma inglés.
- Silabificar automáticamente las cadenas fonéticas.
- Alinear las sílabas fonéticas de palabras provenientes del inglés hacia el katakana.

- Programar matrices de probabilidades para los Modelos Ocultos de Markov a partir de diccionarios.
- Realizar experimentos para la optimización de los HMM.

2. MARCO TEÓRICO

2.1. Conversión Grapheme to Phoneme (G2P)

Un grafema es una secuencia de una o más letras que representan un solo sonido, por ejemplo, la secuencia “Qu” que representa el sonido /k/ en la palabra Quito, o la “i” que resulta en la pronunciación del sonido /i/ en la misma palabra [6]. Por otro lado, un fonema es la unidad más pequeña del lenguaje oral, en otras palabras, el sonido de una letra [6]. Una conversión G2P cual es un proceso de generar la pronunciación de las palabras basándose en su forma escrita. Esta es comúnmente empleada en los sistemas de síntesis de texto a voz y los sistemas de reconocimiento automático de voz [7].

Esta conversión se realiza por medio de algoritmos que poseen una serie de reglas aplicadas de manera ordenada. Estas reglas pueden o no depender de un contexto para ser aplicadas. Generalmente se mapean dos conjuntos de símbolos con la finalidad de reescribir uno en función del otro [8]. Como ejemplo básico, se puede mapear que el grafema <c> tenga una correspondencia con los fonemas /k/ y /s/, aunque esto dependerá de las reglas que se utilicen.

Con el tiempo, estos algoritmos han pasado de poseer reglas escritas a mano con validaciones y análisis manuales, a tener reglas generadas por medio de un gran conjunto de palabras. En [9] se menciona que un conjunto de reglas eficiente debe ser:

- Riguroso. Es decir, poseer restricciones de orden para poder añadir nuevas reglas de manera responsable.
- Completo. Se espera que exista una gran cantidad de reglas que cubran gran cantidad de información.
- Analizado de manera óptima para utilizar las reglas de manera relevante.

En la Sección 2.1.2 se menciona el conjunto de reglas a ser utilizado dentro del presente trabajo para la conversión G2P.

2.1.1. Festvox / Festival

Festvox es un proyecto de la Universidad Carnegie Mellon cuyo principal objetivo es ofrecer un entorno de desarrollo para la síntesis del habla por la comunidad académica internacional. Busca que la construcción de voces sintéticas sea un proceso más sistemático y mejor documentado que permita que la creación de una nueva voz sea accesible para cualquiera [10].

El trabajo de *Festvox* se lo realiza en conjunto con *Festival Speech Synthesis Systems.* Este es un framework desarrollado por la Universidad de Edimburgo el cual permite la síntesis de texto a voz en los idiomas inglés (acento británico y americano), español y galés, siendo el primero el que más desarrollo posee en la actualidad [11].

2.1.2. Letter-to-sound rules

Escribir las reglas de conversión G2P puede ser un proceso difícil en varios idiomas. Por ello, *Festvox* ofrece un método automático con casi nula participación manual en función de un léxico (vocabulario/glosario) de pronunciaciones denominado reglas *letter-to.sound.* Este ya ha sido usado para el inglés, francés y alemán de manera exitosa, aunque su uso depende de cada lenguaje. El proceso detallado en [12] sigue los siguientes pasos:

- Preprocesamiento del léxico hacia una data de entrenamiento adecuada.
- Definición del conjunto permitido de letras a fonemas (allowables / permitidos).
- Construcción de las probabilidades para cada par de letra-fonema.
- Alineación de letras con fonemas o espacios vacios (_epsilons_).
- Extracción de las letras aptas para el entrenamiento.
- Construcción de árboles de decisión CART para la predicción de fonemas a partir de letras y contexto.
- De ser necesario, construcción de un diccionario adicional de asignación en función del léxico original (addenda).

Todo este proceso es automático excepto por los dos primeros pasos. Es importante indicar que como base para el segundo paso se tienen las reglas permitidas del *Oxford Advanced Learner's Dictionary (OALD)* el cual es un diccionario para aprender inglés. Estas reglas se las observa en la Figura 2 y su formato es letra-símboloVacío-fonemasPermitidos para cada una de las letras.

```
((a _epsilon_ ei aa a e@ @ oo au o i ou ai uh e)
 (b _epsilon_ b )
 (c _epsilon_ k s ch sh @-k s t-s)
 (d _epsilon_ d dh t jh)
 (e _epsilon_ @ ii e e@ i @@ i@ uu y-uu ou ei aa oi y y-u@ o)
 (f _epsilon_ f v )
 (g _epsilon_ g jh zh th f ng k t)
 (h _epsilon_ h @ )
 (i _epsilon_ i@ i @ ii ai @@ y ai-@ aa a)
 (j _epsilon_ h zh jh i y )
 (k _epsilon_ k ch )
 (l _epsilon_ l @-l l-l)
 (m _epsilon_ m @-m n)
 (n _epsilon_ n ng n-y )
 (o _epsilon_ @ ou o oo uu u au oi i @@ e uh w u@ w-uh y-@)
 (p _epsilon_ f p v )
 (q _epsilon_ k )
 (r _epsilon_ r @@ @-r)
 (s _epsilon_ z s sh zh )
 (t _epsilon_ t th sh dh ch d )
 (u _epsilon_ uu @ w @@ u uh y-uu u@ y-u@ y-u i y-uh y-@ e)
 (v _epsilon_ v f )
 (w _epsilon_ w uu v f u)
 (x _epsilon_ k-s g-z sh z k-sh z g-zh )
 (y _epsilon_ i ii i@ ai uh y @ ai-@)
 (z _epsilon_ z t-s s zh )
 (# #)
```

Figura 2. Reglas permitidas básicas OALD

2.1.3. Árboles de decisión CART

Una de las herramientas que *Festival* posee son los Árboles de Clasificación y Regresión (CART) [13]. Estos realizan preguntas de si/no en función de ciertas características para obtener una probabilidad de distribución en casos donde se predigan valores categóricos (árbol de clasificación), o una desviación media y estándar cuando se predigan valores continuos (árbol de regresión) [14]. Como se mencionó anteriormente, estos se utilizan para la predicción de los fonemas en función de letras y un contexto.

La estructura CART utilizada en *Festvox/Festival* se encuentra en la Figura 3. Dentro del CART se encuentra un nodo de pregunta (QUESTION-NODE) y un nodo de respuesta (ANSWER-NODE). Con respecto al nodo de pregunta, en su interior se detalla una pregunta (QUESTION) que será respondida por sí o no, y dependiendo de ello se accede al nodo afirmativo (YES-NODE) o negativo (NO-NODE). Ambos nodos poseen otro CART con la misma estructura. Las preguntas utilizadas en el CART están

divididas en seis opciones, las cuales toman como base una característica (FEATURE) [15] y son:

- ¿La característica está en una lista (LIST)?
- ¿La característica es una cadena de texto (STRVALUE)?
- ¿La característica es igual a un número (NUMVALUE)?
- ¿La característica es mayor que un número (NUMVALUE)?
- ¿La característica es menor que un número (NUMVALUE)?
- ¿La característica coincide con una expresión regular (REGEX)?

Dentro del ANSWER-NODE se tienen otras dos opciones de nodo, por un lado, el nodo de clasificación (CLASS-ANSWER) y por otro, el nodo de regresión (REGRESS-ANSWER). Para el primer nodo se detallan los valores (VALUE0, 1, …, N) que pueden tener las respuestas de las preguntas, junto con su probabilidad (PROB), y también se indica cuál es el valor más probable como respuesta (MOST-PROB-VALUE). Para el nodo de regresión se obtiene la desviación estándar (STANDARD-DEVIATION) y la media (MEAN) de los valores.

```
CART ::= QUESTION-NODE || ANSWER-NODE
QUESTION-NODE ::= ( QUESTION YES-NODE NO-NODE )
YES-NODE ::= CART
NO-NODE ::= CART
QUESTION ::= ( FEATURE in LIST )
QUESTION ::= ( FEATURE is STRVALUE )
QUESTION ::= ( FEATURE = NUMVALUE )
QUESTION ::= ( FEATURE > NUMVALUE )
QUESTION ::= ( FEATURE < NUMVALUE )
QUESTION ::= ( FEATURE matches REGEX )
ANSWER-NODE ::= CLASS-ANSWER || REGRESS-ANSWER
CLASS-ANSWER ::= ( (VALUE0 PROB) (VALUE1 PROB) ... MOST-PROB-VALUE )
REGRESS-ANSWER ::= ( ( STANDARD-DEVIATION MEAN ) )
```

Figura 3. Estructura CART en Festvox [14]

En la Figura 4 se detalla un ejemplo básico [16] de como un árbol CART puede ser utilizado dentro de la conversión G2P. En la parte superior se observan cinco casos o “reglas” que sirven de base para el árbol. A partir de estas reglas se realizan preguntas de sí/no para crear el árbol. La primera pregunta corresponde a sí la letra actual (A) es igual al grafema “C”, de ser una respuesta positiva se avanza al nodo de la derecha y en caso negativo se sigue en el nodo izquierdo, en este ejemplo, el nodo de la izquierda podría contener nodos referentes a otro grafema que dependerán de nuevas reglas.

En el nodo de la derecha se encuentra otra pregunta donde se quiere conocer si la letra siguiente (S) corresponde al grafema “O”. De sí serlo, se concluye que el fonema elegido es /k/, pero de no ser así, se sigue el camino de la izquierda. En dicho nodo se encuentra otro CART en el que las preguntas a realizar estén relacionadas con las letras siguientes

"I" o "E", quizás existan preguntas con las letras faltantes "A" y "U", o incluso haya preguntas con otras consonantes. Una muestra del primer caso puede ser la palabra *economy,* mientras que del segundo caso pueden ser palabras como *Cinderella, aplication* o *clock,* todo dependerá del análisis adicional que no se encuentra en la figura. Como dato adicional, la letra previa (P) otorga contexto para realizar la clasificación.

Letra Previa (P)	Letra Actual (A)	Letra Siguiente (S)	Fonema (F)
_	C	I	/s/
_	C	O	/k/
I	C	E	/s/
O	C	O	/k/
E	C	O	/k/

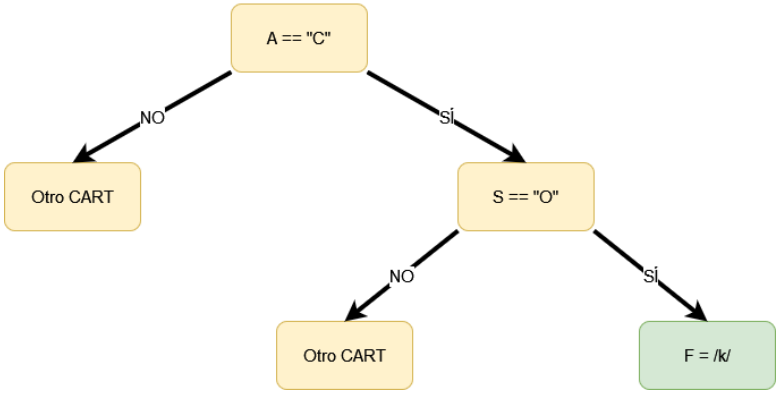


Figura 4. Ejemplo básico de árbol CART

2.2. GIZA++

En 1999 el Centro para Procesamiento de Lenguaje y Habla de la Universidad Johns-Hopkins de Estados Unidos creó un kit de herramientas desarrollado por su equipo de Traducción Automática Estadística [17]. Dentro de este kit se encuentra GIZA, el cual es un programa de entrenamiento que aprende modelos estadísticos de traducción a partir de un corpus bilingüe [17].

GIZA++ es una extensión escrita por Franz Josef Och [18] que incluye características tales como la alineación de modelos en función de clases de palabras, la implementación de Modelos Ocultos de Markov (HMM) relacionados con el entrenamiento Baum-Weich, el algoritmo *Forward-Backward,* mapear palabras vacías, entre otros [19].

La compilación de GIZA++ requiere de un compilador GNU 2.95 o superior, así como un ensamblador y enlazador que no posea restricciones con respecto a la longitud de

los caracteres a utilizar [19]. En el presente trabajo, GIZA++ funciona para realizar la alineación en base a conteos y probabilidad estadística entre secuencias de caracteres katakana y sílabas fonéticas del inglés.

2.2.1. Algoritmo de Expectation-Maximization (EM)

Este algoritmo de Esperanza-Maximización (por su nombre en español) consiste en un proceso iterativo con el propósito de maximizar la verosimilitud cuando se trabaja con datos faltantes [20] [21]. Al tratarse de un proceso iterativo, en cada ocasión se repiten dos pasos:

- Paso E. En este paso se busca estimar las variables faltantes de la data con la que se está trabajando. Para calcular el valor esperado de la verosimilitud se emplea el promedio sobre los datos faltantes junto con una aproximación de la solución.
- Paso M. Aquí se maximizan los parámetros del modelo en conjunto con la data. En otras palabras, se obtiene una nueva aproximación al maximizar la cantidad resultante del paso anterior.

GIZA++ emplea este algoritmo para realizar la alineación final de palabras [22] luego de que se hayan realizado los conteos respectivos y se haya entrenado el modelo en función de los dos diccionarios de entrada que van a ser alineados. Dicha alineación se la observará en la Sección 3.1.5.

2.2.2. Modelos de alineación IBM 1-5

Los modelos de alineación IBM [23] son una secuencia de modelos complejos que se utilizan en la Traducción Automática Estadística (SMT por sus siglas en inglés) con la finalidad de entrenar modelos de traducción y modelos de alineación. GIZA++ emplea estos modelos para realizar la alineación de cadenas. La definición [24] de cada modelo partiendo del inglés es:

- Modelo 1 para la traducción léxica. El proceso de traducción se divide en pasos más pequeños.
- El Modelo 2 incluye un modelo de alineación absoluta. Las alineaciones se dan de una palabra del inglés en posición i, hacia una palabra de otro idioma en posición j.
- Con el Modelo 3 se añade un modelo de fertilidad, la cual es la cantidad de palabras en inglés generadas por la palabra del otro idioma.

- Gracias al Modelo 4 se suma una alineación relativa en función de los dos modelos anteriores.
- Y en Modelo 5 se solucionan problemas de deficiencia, por ejemplo, generación de traducciones imposibles, colocación de múltiples palabras en una misma posición, etc.

2.3. Modelos Ocultos de Markov (HMM)

Los HMM [25] [26] son autómatas probabilísticos los cuales permiten el modelado de procesos estocásticos. Estos procesos son una colección de variables aleatorias $\{X_t : t \in T\}$ que están parametrizadas por un conjunto *T* (espacio temporal), en donde las variables toman valores de un conjunto *S* (espacio de estados) [27]. Las transiciones entre los estados están asociadas con las probabilidades encontradas a partir de la data de entrenamiento de un HMM. En estos existen dos tipos de estados: los ocultos y los observados.

Los primeros corresponden a aquellos que se quieren predecir y por ello no se los puede ver, esto se debe a que la data de entrada no se encuentra en el mismo dominio de lo que se desea inferir. Por otro lado, los estados observados contienen información disponible en el dominio de la cadena de entrada.

En primer lugar, se realiza una alineación de los datos de ambos dominios para empezar con los conteos respectivos. Por medio de la alineación es posible obtener la distribución de probabilidades de emisión que se asocia al dominio observado (sílabas fonéticas del español/inglés) con el dominio oculto (Katakanas). La distribución de probabilidades de transición interna al dominio oculto se obtiene a partir de probabilidades condicionadas a las frecuencias de un carácter Katakana dado un n-grama que lo precede. La distribución de estas probabilidades se obtiene a partir de un corpus.

Con la distribución de probabilidades de transición entre Katakanas, la distribución de probabilidades de emisión y dada una secuencia de observaciones, se podrán inferir la secuencia de estados ocultos más probables [28]. Para ello, dada una secuencia de sílabas fonéticas en inglés (estados observados) se buscará encontrar la secuencia de caracteres Katakanas (estados ocultos) con mayor probabilidad de representar a la secuencia de sílabas de la cadena original.

Para conseguir las sílabas fonéticas es necesario utilizar la conversión G2P que, en este contexto, tendrá como función principal la obtención de los fonemas que servirán como

elementos de entrada para estimar las probabilidades de emisión de los modelos HMM que permitirán la transliteración de caracteres.

2.3.1. Algoritmo de Viterbi

El pseudocódigo de la Figura 5 muestra tres bloques dentro del algoritmo de Viterbi. Cabe señalar que recibe una secuencia de observaciones como entrada. En el primer bloque se encuentra un bucle *for* donde se calcula la probabilidad en tiempo 0. Por otro lado, el segundo bloque presenta un doble bucle *for* que permite calcular las probabilidades del tiempo 1 en adelante, es decir, este proceso se repite hasta el final de las observaciones de entrada. Este proceso se lo realiza considerando la observación de entrada y el estado anterior de la predicción. Por último, el tercer bloque calcula la probabilidad final de Viterbi junto con el camino que maximiza la secuencia de probabilidades entre los estados ocultos que mejor explican la secuencia de observaciones.

Viterbi Algorithm

```
function VITERBI(O, S, Π, Y, A, B) : X
    for each state i = 1, 2, ..., K do
        T_1[i, 1] ← π_i · B_{i y_1}
        T_2[i, 1] ← 0
    end for
    for each observation j = 2, 3, ..., T do
        for each state i = 1, 2, ..., K do
            T_1[i, j] ← max_k (T_1[k, j − 1] · A_ki · B_{i y_j})
            T_2[i, j] ← arg max_k (T_1[k, j − 1] · A_ki · B_{i y_j})
        end for
    end for
    z_T ← arg max_k (T_1[k, T])
    x_T ← s_{z_T}
    for j = T, T − 1, ..., 2 do
        z_{j−1} ← T_2[z_j, j]
        x_{j−1} ← s_{z_{j−1}}
    end for
    return X
end function
```

Figura 5. Algoritmo de Viterbi

En el siguiente ejemplo se desea conocer la secuencia más probable del clima, dada una serie de acciones de una persona. Se tienen dos estados referentes al clima en el dominio oculto, “soleado” y “lluvioso”. Mientras que los estados en el dominio observable son tres: “limpiar”, “comprar” y “caminar”.

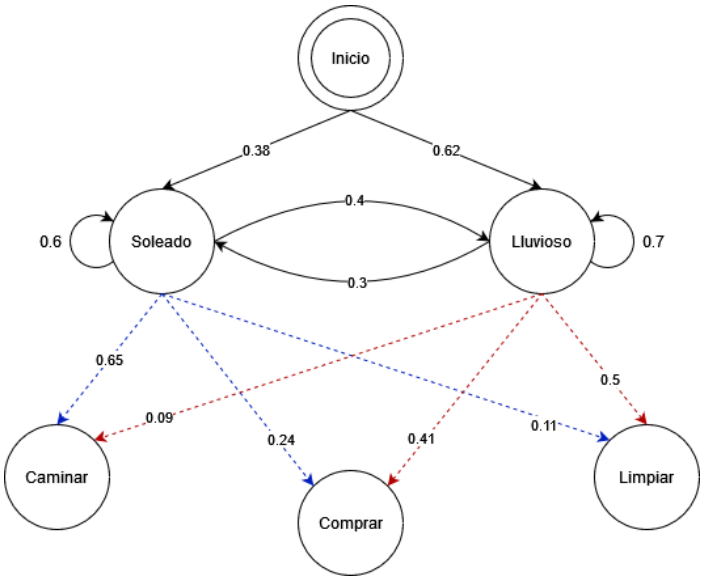


Figura 6. Ejemplo de probabilidades en Viterbi

Las probabilidades para utilizar se observan en la Figura 6. Primero se tiene una probabilidad de inicial de con que estado del dominio oculto podría empezar la secuencia a predecir. Sigue la probabilidad de transición que permite saber que tan probable es transicionar entre los estados del dominio oculto, es decir, pasar de “soleado” a “lluvioso” y sus diferentes combinaciones. Por su lado, la probabilidad de emisión permite conocer la probabilidad de llegar a un estado del dominio oculto dado un estado del dominio observable, en otras palabras, cual es la probabilidad de que el clima sea “soleado” dado que la acción de la persona fue “caminar”.

Como ejemplo, se desea conocer el clima más probable que había, dado que una persona realizó tres acciones en el siguiente orden “caminar”, “comprar” y “limpiar”. Siguiendo el pseudocódigo de la Figura 5, la resolución de este ejemplo es la siguiente. Primero se calcula en el tiempo cero (t=0) con el estado “caminar” y con ambos estados ocultos. La probabilidad de que el clima sea soleado dado que se caminó es igual a 0.38*0.65 = 0.247. Por otro lado, la probabilidad de que el clima sea lluvioso dado que se caminó es igual a 0.62*0.09 = 0.0558. Así pues, es más probable que el clima sea “soleado”.

En la siguiente iteración se tiene de observación a la acción “comprar”, sin embargo, a partir de t=1, ya se toman en cuenta a las probabilidades de transición entre estados ocultos. Primero se calcula la probabilidad considerando que el clima en t=1 será “soleado”, esto indica que las transiciones pudieron ser “soleado-soleado” y “lluvioso-soleado”. Para la primera combinación la probabilidad es igual al valor del estado “soleado” en función del tiempo anterior t=0 multiplicado por la transición de estados “soleado-soleado”, es decir, 0.247*0.6 = 0.1482. La segunda combinación multiplica el

valor del estado “lluvioso” en t=0 con la transición de “lluvioso-soleado” dando así 0.0558*0.3 = 0.01674. Viterbi entonces elige el valor más alto y lo multiplica por la probabilidad de emisión de la acción “comprar” considerando que el clima en t=1 esta vez es “soleado”, siendo así 0.1482*0.24 = 0.035568.

Siguiendo en t=1, ahora se calcula la probabilidad considerando que el clima será “lluvioso” y se repite el procedimiento descrito anteriormente, en esta ocasión las transiciones podrán ser “soleado-lluvioso” y “lluvioso-lluvioso”. De igual manera se usan las probabilidades del tiempo anterior t=0. Para “soleado-lluvioso” el cálculo es 0.247*0.4 = 0.0988 mientras que para “lluvioso-lluvioso” el valor es 0.0558*0.7 = 0.03906. Una vez más Viterbi elige el mayor valor y lo multiplica esta vez por la emisión de “comprar” con el clima “lluvioso” en t=1 con resultado igual a 0.0988*0.41 = 0.040508. Se comparan los dos valores finales (0.035568 y 0.040508) donde se concluye que el estado del clima más probable en t=1 es “lluvioso” con probabilidad de 0.040508.

Para las probabilidades en t=2 el procedimiento es el mismo que en t=1 tal como se ve en el seudocódigo. Por ende, ahora se trabaja a la observación “limpiar” con las probabilidades del clima en función de t=1 y se supone primero que el clima en t=2 será “soleado” con transiciones “soleado-soleado” y “lluvioso-soleado”. Los cálculos son 0.035568*0.6 = 0.02134 en la primera transición y 0.040508*0.3 = 0.01215. Viterbi elige el valor más alto y lo multiplica por la probabilidad de emisión de “limpiar” en función del clima “soleado” de valor 0.02134*0.11 = 0.00234.

Ahora con t=2 suponiendo que el clima será “lluvioso”, las transiciones son “soleado-lluvioso” cuyo calculo es 0.035568*0.4 = 0.014227, y la segunda transición “lluvioso-lluvioso” 0.040508*0.7 = 0.028355. De igual manera, Viterbi usa el mayor valor y lo multiplica por la probabilidad de emisión de “limpiar” dado que el clima es “lluvioso” dando como resultado 0.028355*0.5 = 0.01417. Una vez más se comparan ambos valores finales (0.00234 y 0.01417) y se obtiene como clima más probable al estado “lluvioso” con probabilidad de 0.01417.

Al no tener más observaciones que predecir, se ha llegado al final del algoritmo y la secuencia de clima más probable para la observación “caminar-comprar-limpiar” es “soleado-lluvioso-lluvioso” con un valor probable máximo igual a 0.01417. En la Tabla 2 se observa un resumen de cálculos.

Tabla 1. Probabilidades ejemplo predicción de clima

INICIO	
SOLEADO	0,38
LLUVIOSO	0,62

TRANSICIÓN		
	SOLEADO	**LLUVIOSO**
SOLEADO	0,6	0,4
LLUVIOSO	0,3	0,7

EMISIÓN			
	CAMINAR	**COMPRAR**	**LIMPIAR**
SOLEADO	0,65	0,24	0,11
LLUVIOSO	0,09	0,41	0,5

Tabla 2. Ejecución de Viterbi en ejemplo de clima

			INICIO	**t=0**	**t=1**				**t=2**			
		P[SOL]	0,38	**0,247**			**P[SOL]**	0,035568			**P[SOL]**	0,00234
		P[LLUV]	0,62	0,0558			**P[LLUV]**	**0,040508**			**P[LLUV]**	**0,01417**
TRANSICION												
PREVIO	**ACTUAL**	**P[T]**			**P[T]*P[SOL]**	**MAYOR**	***P[E] (SOL)**		**P[T]*P[SOL]**	**MAYOR**	***P[E] (SOL)**	
SOLEADO	SOLEADO	0,6	-	-	0,1482	x	0,035568		0,02134	x	0,00234	
LLUVIOSO	SOLEADO	0,3	-	-	0,01674				0,01215			
TRANSICION												
PREVIO	**ACTUAL**	**P[T]**			**P[T]*P[LLUV]**	**MAYOR**	***P[EM](LLUV)**		**P[T]*P[LLUV]**	**MAYOR**	***P[E](LLUV)**	
SOLEADO	LLUVIOSO	0,4	-	-	0,0988	x	0,040508		0,014227			
LLUVIOSO	LLUVIOSO	0,7	-	-	0,03906				0,028355	x	0,01417	

EMISIÓN (E)
TRANSICIÓN (T)

2.4. PERL

Para la programación se utiliza el lenguaje de programación PERL [29] . Originalmente era un lenguaje optimizado para la extracción de información de archivos de texto y la generación de reportes a partir de dicha información. Gradualmente fue evolucionando a un lenguaje de programación con propósitos generales con la intención principal de generar código práctico sin darle tanto protagonismo al apartado estético [30].

Se utiliza PERL precisamente gracias a sus bases en manejo de texto y en el amplio catálogo de opciones para la manipulación de archivos, procesos y cadenas de caracteres, además del uso de librerías pensadas para trabajar con expresiones regulares.

2.5. Distancia de Levenshtein

Esta distancia propuesta por el científico ruso Vladimir Levenshtein en 1965 consiste en calcular la diferencia de dos cadenas de caracteres [31]. Su valor es el número mínimo de ediciones de un único carácter que ocasiona que una cadena cambie por otra. Dichas ediciones pueden ser la inserción, eliminación y sustitución de caracteres [32]. Por ejemplo, la distancia entre “pala” y “pasta” es de dos. Las ediciones en cuestión son la sustitución de “l” por “s” y la inserción de “t” luego de la nueva “s”.

Para el contexto actual, la distancia de Levenshtein será de utilidad al momento de comparar los resultados que el programa de transliteración arroje junto con herramientas ya existentes de transliteración en la Sección 5.1.

3. METODOLOGÍA

Es viable utilizar un HMM para la transliteración de palabras. En [33] se presenta un modelo para transliterar una cadena en inglés hacia cinco diferentes lenguajes: tamil, hindi, ruso, chino y canarés. Para ello a partir de datos de prueba y aprendizaje se definió un HMM capaz de derivar subcadenas con un estado finito probabilístico con las cuales tras el entrenamiento respectivo conseguían inferir la cadena de caracteres correspondientes a cada lenguaje.

Para la elaboración y optimización del HMM, se ha optado por adoptar un enfoque iterativo basado en el desarrollo en espiral [34]. El proceso de obtención de diccionarios y generación de los modelos base para el HMM puede realizarse de manera secuencial.

Por otro lado, el cálculo de las probabilidades y la optimización del HMM emplearán el método en espiral para mejorar con cada nueva iteración. A medida que se encuentre la parametrización adecuada para el modelo, se podrá demostrar en el proceso que los HMM son una alternativa a considerar que sirva como base o punto de partida cuando se analicen ciertos problemas de transliteración.

3.1. DATA

Para la elaboración de un HMM se necesita como base un diccionario de emisión y un diccionario de transición. Estos se obtienen a partir de una data tratada con anterioridad que permita obtener las respectivas probabilidades de emisión y transición. Para el diccionario de emisión se requiere poseer una alineación entre una secuencia de sílabas fonéticas del inglés con una secuencia de caracteres katakana.

Por otro lado, para el diccionario de transición se precisa de una alineación entre dos secuencias de caracteres katakana que se preceden una de otra, en función de la palabra o frase de donde provienen. En ambos diccionarios se incluirán dos caracteres, uno que represente el inicio de cadena (<s>) y otro el fin de dicha cadena (</s>) para identificar cada una de las entradas.

La data surge a partir de un diccionario conocido como EDICT2 que posee palabras en inglés con su respectiva transliteración al katakana.

3.1.1. EDICT2

El proyecto EDICT tiene como objetivo la producción de un diccionario japonés/inglés que posea un formato legible y esté disponible de manera gratuita con la finalidad de ser utilizado en una gran variedad de proyectos [35]. El archivo EDICT2 tiene una estructura simple de una entrada por línea cuyo contenido es la traducción de palabras y frases provenientes del japonés hacia el inglés. Estas traducciones siguen los requerimientos del léxico japonés entre las que se encuentran diferentes formas de escritura, variaciones ortográficas, variantes en otros sistemas de escritura como el kanji y el hiragana, etc. [35]. La parte donde EDICT2 posee traducciones del katakana al inglés será el corpus base con el que se trabajará la data. Cabe recalcar que es necesario usar EDICT2 con formato de codificación UTF-8.

```
8130 アウトフィット /(n) outfit/EntL2155500X/
8131 アウトフォーカス;アウト・フォーカス /(n) soft focus (photography, film) (wasei: out focus)/out-of-focus
     technique/EntL2622390/
8132 アウトフロー /(n) outflow/EntL2430940X/
8133 アウトブリーディング /(n) (See アウトブリード) outbreeding/EntL2689070/
8134 アウトブリード /(n,vs) outbreeding/EntL2491420/
8135 アウトブレイク;アウトゥブレイク /(n) outbreak (of war or disease)/EntL2129120/
8136 アウトプット /(n,vs) (See インプット) output/EntL1014850X/
8137 アウトプットポート;アウトプット・ポート /(n) {comp} output port/EntL2278570X/
8138 アウトプレースメント /(n) outplacement/EntL2455580/
8139 アウトボクシング;アウト・ボクシング /(n) out boxing/EntL1014860X/
8140 アウトボックス /(n) {comp} out-box/EntL2278580X/
8141 アウトポケット;アウト・ポケット /(n) (abbr) (See アウトサイドポケット) outside pocket (wasei: out pocket)/patch
     pocket/EntL1014870X/
8142 アウトライト取引 [アウトライトとりひき] /(n) outright transaction/EntL1014880X/
```

Figura 7. Muestra de contenido de EDICT2

Por la naturaleza de EDICT2 es necesario realizar una limpieza de este. En la Figura 7 se observa la estructura del corpus y se detalla que cada nueva entrada posee elementos separados entre sí por el carácter /. Tomando de ejemplo la línea 8132 se encuentra la palabra katakana “アウトフロー” seguida de su transliteración al inglés que es el sustantivo *outflow* y a este lo sigue lo que parece ser un identificador para dicha entrada. Sin embargo, la entrada de la línea 8141 es más compleja pues aquí se tienen dos secuencias katakana correctas. A su vez, la transliteración indica que se trata de una abreviación de la palabra *outside pocket* y que la secuencia katakana corresponde a la expresión wasei (construcciones inglesas usadas principalmente en Japón) *out pocket.* En otras entradas aparecen además caracteres kanji o hiragana.

EDICT2 posee 192 800 entradas, de las cuales, 42 788 siguen la estructura antes mencionada y se puede divisar una muestra en la Figura 8. Esta data seguirá con un proceso de limpieza hasta llegar a un diccionario limpio donde solo se considere una secuencia katakana y una transliteración al inglés.

1	アーウィン	(n) {comp} ERwin	
2	アーカイバ;アーカイバー	(n) {comp} archiver	
3	アーカイバル	(adj-f) archival	
4	アーカイバルファイル;アーカイバル・ファイル	(n) {comp} archival file	
5	アーカイバルメモリ;アーカイバル・メモリ	(n) {comp} archival memory	
6	アーカイビング	(n) {comp} archiving	
7	アーカイブ;アーカイヴ	(n) archive	
8	アーカイブサーバ;アーカイブサーバー;アーカイブ	(n) {comp} archive server	
9	アーカイブス;アーカイブズ	(n) (See アーカイブ) archives	
10	アーカイブデータセット;アーカイブ・データセッ	(n) {comp} archive data set	
11	アーカイブビット;アーカイブ・ビット	(n) {comp} archive bit	
12	アーカイブファイル;アーカイブ・ファイル	(n) {comp} archive file	
13	アーカイブライブラリ;アーカイブ・ライブラリ	(n) {comp} archive library	
14	アーカイブ先 [アーカイブさき]	(n) {comp} archiving destination	
15	アーカイブ属性 [アーカイブぞくせい]	(n) {comp} archive attribute	
16	アーカイブ対象ファイル [アーカイブたいしょうフ	(n) {comp} archived file	
17	アーカンソー	(n) Arkansas	
18	アーガイル;アーガル(ik)	(n) argyle	
19	アーガイルチェック;アーガイル・チェック;アーガ	(n) argyle check	

Figura 8. Punto de partida para limpieza de EDICT2

3.1.2. Limpieza de EDICT2

Partiendo de los 42 788 registros, se procede a eliminar aquellos que posean caracteres en hiragana o en kanji. Esto se realiza en función del bloque Unicode de los caracteres katakana, el cual es desde U+30A0 hasta U+30FF [36]. De este modo quedan 33 500 registros con una nueva estructura, donde solo existen las secuencias katakana y la transliteración en inglés.

La expresión regular utilizada es: *^[\x{30A0}-\x{30FF};\(\)\[\]\s\w]*\s*\(.*$* y a simple vista se observa que en la Figura 9 ya han desaparecido ciertos registros que tenían kanji como el 15 y 16 de la Figura 8. Sin embargo, en varias filas como la 2 o la 7, se identifican más de una secuencia katakana válidas para una misma palabra en inglés y las cuales están separadas por punto y coma (;). Mientras que en la sección en inglés tampoco se tiene una transliteración limpia, sino que existen datos adicionales que toca extraer. La limpieza ahora se dividirá en dos, por un lado, se trabaja solo con las secuencias katakana, y por otro solo con las transliteraciones. En ambos casos se usan expresiones regulares.

1	アーウィン	(n) {comp} ERwin
2	アーカイバ;アーカイバー	(n) {comp} archiver
3	アーカイバル	(adj-f) archival
4	アーカイバルファイル;アーカイバル・ファイル	(n) {comp} archival file
5	アーカイバルメモリ;アーカイバル・メモリ	(n) {comp} archival memory
6	アーカイビング	(n) {comp} archiving
7	アーカイブ;アーカイヴ	(n) archive
8	アーカイブサーバ;アーカイブサーバー;アーカイブ	(n) {comp} archive server
9	アーカイブス;アーカイブズ	(n) (See アーカイブ) archives
10	アーカイブデータセット;アーカイブ・データセッ	(n) {comp} archive data set
11	アーカイブビット;アーカイブ・ビット	(n) {comp} archive bit
12	アーカイブファイル;アーカイブ・ファイル	(n) {comp} archive file
13	アーカイブライブラリ;アーカイブ・ライブラリ	(n) {comp} archive library
14	アーカンソー	(n) Arkansas
15	アーガイル;アーガル(ik)	(n) argyle
16	アーガイルチェック;アーガイル・チェック;アーガ	(n) argyle check
17	アーガット	(n) ergot
18	アーキオロジー;アーケオロジー	(n) archaeology
19	アーキタイプ	(n) archetype

Figura 9. EDICT2 sin kanji

Con respecto al katakana, se sigue un proceso secuencial donde cada vez se eliminan cosas innecesarias. En la Figura 10 se pueden identificar cinco columnas que van de la letra A hasta la letra E. En la columna A están las secuencias katakanas válidas del anterior paso, por lo que en la segunda columna B se elimina todo lo que se encuentre después del primer punto y coma, dejando así sólo la primera secuencia katakana. En la columna C en cambio se elimina lo que se encuentre después de un paréntesis de entrada como es el caso de la fila 21. Para la columna D se han eliminado caracteres del alfabeto latino y por último en la columna E se extrae el espacio en blanco del final de cada fila.

Para la limpieza de las palabras en inglés se han utilizado más expresiones regulares. En la Figura 11 se ven las columnas con dichas expresiones que indican aquello que se va a eliminar de las expresiones en inglés. De igual manera que con el katakana, en la columna A están las expresiones tras la primera limpieza. Para las columnas B y C se han eliminado lo que está entre paréntesis al inicio de las filas. En la columna D se elimina los datos entre llaves. Una vez más en las columnas E, F y G se han eliminado los paréntesis del inicio que pudieron aparecer tras ejecutar el paso anterior. Para la columna H se ha realizado una validación manual en busca de datos entre paréntesis al inicio de la fila y por último en la columna I se eliminan los paréntesis del final de cada registro.

	A	B	C	D	E
1	**ORIGINAL**	**;.***	**\(.***	**[a-zA-Z]**	**QUITAR ESPACIO EN BLANCO FINAL**
2	アーウィン	アーウィン	アーウィン	アーウィン	アーウィン
3	アーカイバ;アーカイバー	アーカイバ	アーカイバ	アーカイバ	アーカイバ
4	アーカイバル	アーカイバル	アーカイバル	アーカイバル	アーカイバル
5	アーカイバルファイル;ア	アーカイバルファイル	アーカイバルファイル	アーカイバルファイル	アーカイバルファイル
6	アーカイバルメモリ;アー	アーカイバルメモリ	アーカイバルメモリ	アーカイバルメモリ	アーカイバルメモリ
7	アーカイビング	アーカイビング	アーカイビング	アーカイビング	アーカイビング
8	アーカイブ;アーカイヴ	アーカイブ	アーカイブ	アーカイブ	アーカイブ
9	アーカイブサーバ;アーカ	アーカイブサーバ	アーカイブサーバ	アーカイブサーバ	アーカイブサーバ
10	アーカイブス;アーカイブ	アーカイブス	アーカイブス	アーカイブス	アーカイブス
11	アーカイブデータセット	アーカイブデータセット	アーカイブデータセット	アーカイブデータセット	アーカイブデータセット
12	アーカイブビット;アーカ	アーカイブビット	アーカイブビット	アーカイブビット	アーカイブビット
13	アーカイブファイル;アー	アーカイブファイル	アーカイブファイル	アーカイブファイル	アーカイブファイル
14	アーカイブライブラリ;ア	アーカイブライブラリ	アーカイブライブラリ	アーカイブライブラリ	アーカイブライブラリ
15	アーカンソー	アーカンソー	アーカンソー	アーカンソー	アーカンソー
16	アーガイル;アーガル(ik)	アーガイル	アーガイル	アーガイル	アーガイル
17	アーガイルチェック;アー	アーガイルチェック	アーガイルチェック	アーガイルチェック	アーガイルチェック
18	アーガット	アーガット	アーガット	アーガット	アーガット
19	アーキオロジー;アーケオ	アーキオロジー	アーキオロジー	アーキオロジー	アーキオロジー
20	アーキタイプ	アーキタイプ	アーキタイプ	アーキタイプ	アーキタイプ
21	アーキテクチャ(P);アーキ	アーキテクチャ(P)	アーキテクチャ	アーキテクチャ	アーキテクチャ

Figura 10. Limpieza de secuencias katakana

	A	B	C	D	E	F	G	H	I
1	**ORIGINAL**	**^\([-"':;. ・ ,\w\s]*\)\s**	**^\([-"':;. ・ ,\w\s]*\)\s**	**^\([-"':;. ・ ,\w\s]*\)\s**	**^\([-"':;. ・ ,\w\s]*\)\s**	**^\([-"':;. ・ 、 ,\w\s]*\)\s**	**^\([-"':;+. ・ 、 ,\w\s]*\)\s**	**^\(.* MANUAL**	**\s\(.***
2	(n) {comp} ERwin	{comp} ERwin	{comp} ERwin	ERwin	ERwin	ERwin	ERwin	ERwin	Erwin
3	(n) {comp} archiver	{comp} archiver	{comp} archiver	archiver	archiver	archiver	archiver	archiver	archiver
4	(adj-f) archival	archival	archival	archival	archival	archival	archival	archival	archival
5	(n) {comp} archival file	{comp} archival file	{comp} archival file	archival file	archival file	archival file	archival file	archival file	archival file
6	(n) {comp} archival memory	{comp} archival memory	{comp} archival memory	archival memory	archival memory	archival memory	archival memory	archival memory	archival memory
7	(n) {comp} archiving	{comp} archiving	{comp} archiving	archiving	archiving	archiving	archiving	archiving	archiving
8	(n) archive	archive	archive	archive	archive	archive	archive	archive	archive
9	(n) {comp} archive server	{comp} archive server	{comp} archive server	archive server	archive server	archive server	archive server	archive server	archive server
10	(n) (See アーカイブ) archives	(See アーカイブ) archives	archives	archives	archives	archives	archives	archives	archives
11	(n) {comp} archive data set	{comp} archive data set	{comp} archive data set	archive data set	archive data set	archive data set	archive data set	archive data set	archive data set
12	(n) {comp} archive bit	{comp} archive bit	{comp} archive bit	archive bit	archive bit	archive bit	archive bit	archive bit	archive bit
13	(n) {comp} archive file	{comp} archive file	{comp} archive file	archive file	archive file	archive file	archive file	archive file	archive file
14	(n) {comp} archive library	{comp} archive library	{comp} archive library	archive library	archive library	archive library	archive library	archive library	archive library
15	(n) Arkansas	Arkansas	Arkansas	Arkansas	Arkansas	Arkansas	Arkansas	Arkansas	Arkansas
16	(n) argyle	argyle	argyle	argyle	argyle	argyle	argyle	argyle	argyle
17	(n) argyle check	argyle check	argyle check	argyle check	argyle check	argyle check	argyle check	argyle check	argyle check
18	(n) ergot	ergot	ergot	ergot	ergot	ergot	ergot	ergot	ergot
19	(n) archaeology	archaeology	archaeology	archaeology	archaeology	archaeology	archaeology	archaeology	archaeology
20	(n) archetype	archetype	archetype	archetype	archetype	archetype	archetype	archetype	archetype
21	(n) architecture	architecture	architecture	architecture	architecture	architecture	architecture	architecture	architecture

Figura 11. Limpieza de expresiones en inglés

Tras la limpieza, se juntan las dos partes limpias para generar la estructura deseada de una secuencia katakana seguida de una expresión en inglés. Esta limpieza no es perfecta y por ende se ha perdido parte de la información en el proceso dejando como resultado 33 491 entradas válidas para su uso.

```
Erwin   アーウィン
archiver    アーカイバ
archival    アーカイバル
archival file   アーカイバルファイル
archival memory アーカイバルメモリ
archiving   アーカイビング
archive アーカイブ
archive server  アーカイブサーバ
archives    アーカイブス
archive data set    アーカイブデータセット
archive bit アーカイブビット
archive file    アーカイブファイル
archive library アーカイブライブラリ
Arkansas    アーカンソー
argyle  アーガイル
argyle check    アーガイルチェック
ergot   アーガット
archaeology アーキオロジー
```

Figura 12. Contenido EDICT2 post limpieza

3.1.3. División de la data

A partir del diccionario EDICT2 se forman dos conjuntos de data, uno de entrenamiento y uno de pruebas. Como sus nombres lo indican, un conjunto servirá para entrenar el HMM, mientras que el otro se lo utilizará para probar el HMM. Es importante considerar que la data de pruebas sea lo suficientemente grande para otorgar resultados significativos, a la par que sea una muestra representativa de la data completa y que posea características similares con la data de entrenamiento [37].

Las proporciones más comunes para dividir la data son entre 70% a 80% para la data de entrenamiento, mientras que de 30% a 20% para la data de pruebas. Cabe recalcar que al final los resultados que arrojen la data de pruebas suelen tener menor precisión que los resultados obtenidos con la data de entrenamiento. En caso de que ocurra lo contrario puede ser una señal de que una parte de la información presente en la data de pruebas también se encuentre dentro de la data de entrenamiento lo cual imposibilita tener una medida precisa de que tan bien trabaja un modelo con nueva data [37].

Siguiendo los porcentajes antes establecidos, se dividen de manera aleatoria los 33 491 registros iniciales. La data de entrenamiento posee entonces 23 444 entradas, mientras que la data de prueba tiene las 10 047 entradas restantes. En adelante, se trabajará con

la data de entrenamiento para la generación del HMM y la data de prueba únicamente será usada para el análisis de resultados.

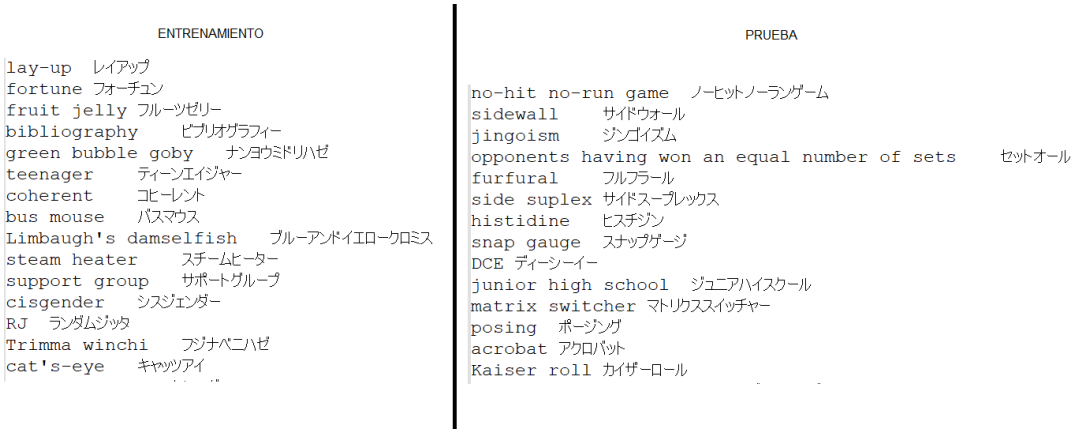


Figura 13. Muestra de data de entrenamiento y prueba

3.1.4. Obtención de fonemas y secuencias katakana

A raíz de la data de entrenamiento se obtienen los fonemas presentes en la respectiva palabra en inglés por medio de G2P. Este proceso se lo realiza de manera automática utilizando las herramientas presentadas en la Sección 2.1.1 de *Festvox* y *Festival,* en especial por medio de los modelos CART y las reglas letter-to-sound. Como esto ya está entrenado con anterioridad, basta con ejecutar las herramientas y obtener las conversiones a partir de la lectura de la estructura Utterance de *Festival*.

Una estructura Utterance [38] (enunciados en español) codifica las propiedades relevantes del enunciado que se va a ser utilizado en la síntesis de voz de *Festival.* La estructura se crea a partir de la pronunciación de dicho enunciado por parte de un hablante nativo. Con ello se posee dicho enunciado con propiedades fonéticas correctas y con una pronunciación natural del idioma.

La estructura en sí consiste en una serie de elementos (palabras, sílabas, frases, etc.) que se encuentran conectados en función de relaciones. Cómo se representan estas relaciones depende del método de síntesis que se emplee. Sin embargo, ciertas relaciones como la conexión de palabras y sílabas, la interrelación de fonemas y la conectividad de estas relaciones de manera jerárquica.

La Figura 14 muestra un ejemplo de esta estructura para un enunciado en inglés con pronunciación americana. En el primer nivel del árbol se detalla la letra “B”, esto indica que se está trabajando con la estructura del “enunciado B”, el cual forma parte de una

frase u oración más compleja. Para el segundo nivel ya se indican las palabras que componen a dicho enunciado. Por otra parte, en el tercer nivel se encuentran las pronunciaciones de las sílabas fonéticas de la palabra en cuestión. Por último, el cuarto nivel representa a las secuencias fonéticas conformadas a partir de las respectivas sílabas fonéticas.

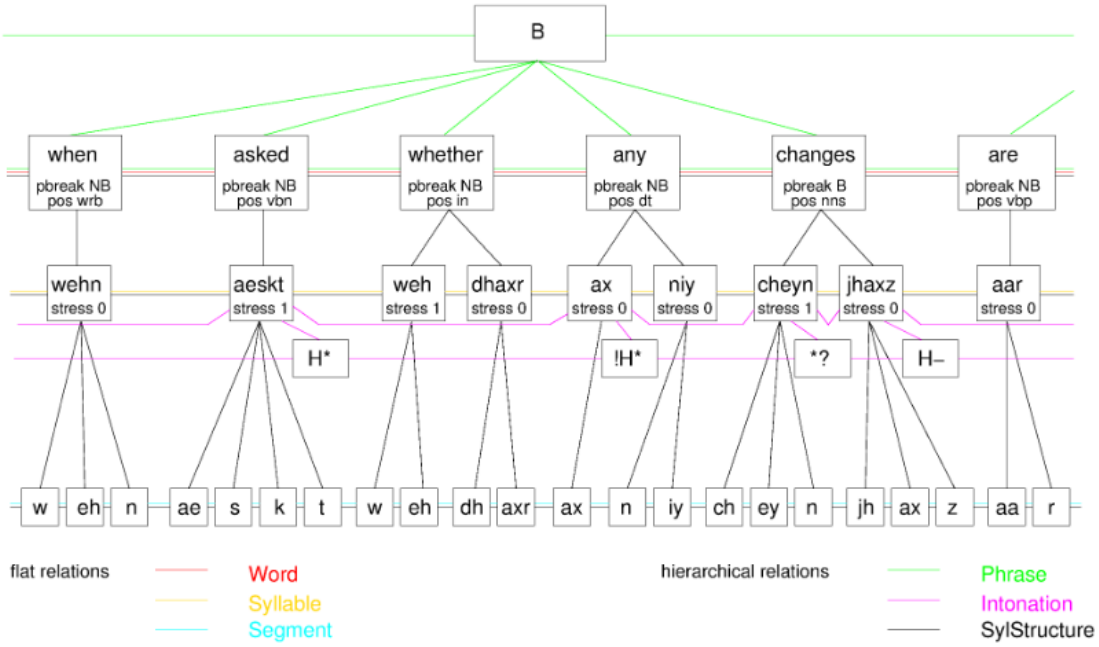


Figura 14. Ejemplo estructura Utterance [38]

En la Figura 15 se detalla una muestra de los resultados de la conversión. Cada registro tiene la misma estructura la cual tiene como separador a los caracteres “=>”. Primero se tiene un identificador seguido de las palabras de entrada. A esta le siguen las mismas palabras con la diferencia de que se añade un separador con el carácter “|” para indicar que dicho registro tiene múltiples palabras como en el registro 5. Por último, se tiene finalmente a los fonemas en cuestión. Los ~ separan a las sílabas fonéticas y estos serán remplazados por espacios en blanco para el proceso de alineación.

```
"1=>lay-up=>lay-up=>`l_ey - `ah_p"
"2=>fortune=>fortune=>`f_ao_r~ch_ax_n"
"3=>fruit jelly=>fruit|jelly=>`f_r_uw_t | `jh_eh~l_iy"
"4=>bibliography=>bibliography=>`b_ih~b_l_iy~`aa~g_r_ax~f_iy"
"5=>green bubble goby=>green|bubble|goby=>`g_r_iy_n | `b_ah~b_ax_l | `g_ow~b_iy"
"6=>teenager=>teenager=>`t_iy~`n_ey~jh_er"
"7=>coherent=>coherent=>k_ow~`hh_ih~r_ax_n_t"
"8=>bus mouse=>bus|mouse=>`b_ah_s | `m_aw_s"
"9=>Limbaugh's damselfish=>Limbaugh's|damselfish=>`l_ih_m~`b_ao_z |
`d_ae_m~s_ax_l~`f_ih_sh"
"10=>steam heater=>steam|heater=>`s_t_iy_m | `hh_iy~t_er"
"11=>support group=>support|group=>s_ax~`p_ao_r_t | `g_r_uw_p"
"12=>cisgender=>cisgender=>s_ih_s~`g_eh_n~d_er"
"13=>RJ=>RJ=>`aa-r~`jh-ey"
"14=>Trimma winchi=>Trimma|winchi=>`t_r_ih~m_ax | `w_ih_n~k_iy"
"15=>cat's-eye=>cat's-eye=>`k_ae_t_s - `ay"
```

Figura 15. Resultados G2P

Por otro lado, a las secuencias katakana también se les realiza un tratamiento previo a la alineación. Por medio de una herramienta de expresiones regulares (regex101 [39]) se aplican ciertas reglas para que la secuencia katakana esté en un formato que permita realizar la posterior alineación. Las secuencias katakana originales son extraídas de la data de entrenamiento.

Tabla 3. Reglas para limpieza de katakana

REGLA	EXPRESIÓN REGULAR	EXPLICACIÓN
1	s/(.)/$1 /gm	Añade espacio en blanco luego de cada katakana.
2	s/ $//gm	Elimina el último espacio en blanco.
3	s/ ([ァィゥェォャュョヮーッン])/$1/gm	Une los katakanas pequeños con el katakana anterior.

Tabla 4. Ejemplo de katakana post limpieza

スロープスタイル (slopestyle)	
REGLA	**CAMBIO**
1	ス ロ ー プ ス タ イ ル
2	ス ロ ー プ ス タ イ ル
3	ス ロー プ ス タ イ ル

マニュファクチャリング (manufactoring)	
REGLA	**CAMBIO**
1	マ ニ ュ フ ァ ク チ ャ リ ン グ
2	マ ニ ュ フ ァ ク チ ャ リ ン グ
3	マ ニュ ファ ク チャ リン グ

3.1.5. Alineación

En el corpus original se posee una alineación de palabras en inglés con secuencias katakana. Sin embargo, es necesario alinear los fonemas de las mismas palabras en inglés con su correspondiente secuencia katakana. Para realizar este proceso se utiliza la herramienta *Multi-threaded GIZA* (MGIZA) la cual es una variante de GIZA++ con mayor rendimiento.

La instalación de MGIZA está detallada en [40], basta con tener un ambiente Ubuntu y las librerías cmake y libboost-all-dev. El proceso consiste en clonar el repositorio de Git e instalar la herramienta con la librería make.

Para ejecutar MGIZA se necesitan varios archivos dentro de una misma carpeta, la cual en esta ocasión será “train_v2”. Primero se coloca el contenido de la carpeta “mgizapp/bin” junto con el archivo “merge_alignment.py” que se encuentra dentro de “mgizapp/inst/scripts”. A continuación, se colocan los archivos que serán alineados entre sí con 23 442 entradas cada uno (en el camino se han perdido 2 registros del total de 23 444). En este caso se trabaja con un archivo en inglés que posea los fonemas (train.en) y otro en japonés que posea las secuencias katakana (train.jp). Para el primer archivo, en caso de que una entrada contenga múltiples palabras, se las separa por el carácter “|” como se indicó anteriormente. Por último, se añade el archivo “configfile” de configuración de mgiza en donde se especifica que se trabajará con los archivos a alinear mencionados anteriormente.

El archivo “configfile” se encuentra en el siguiente enlace https://pastebin.com/b1ksHtUy y pertenece al autor de [40]. Dicho archivo contiene las configuraciones recomendadas para un correcto uso de MGIZA. De este se presta especial atención a la opción *ncpus* en donde se define la cantidad de hilos que usará MGIZA para la alineación. El número de hilos depende de los núcleos del computador y se recomienda duplicar aquel valor.

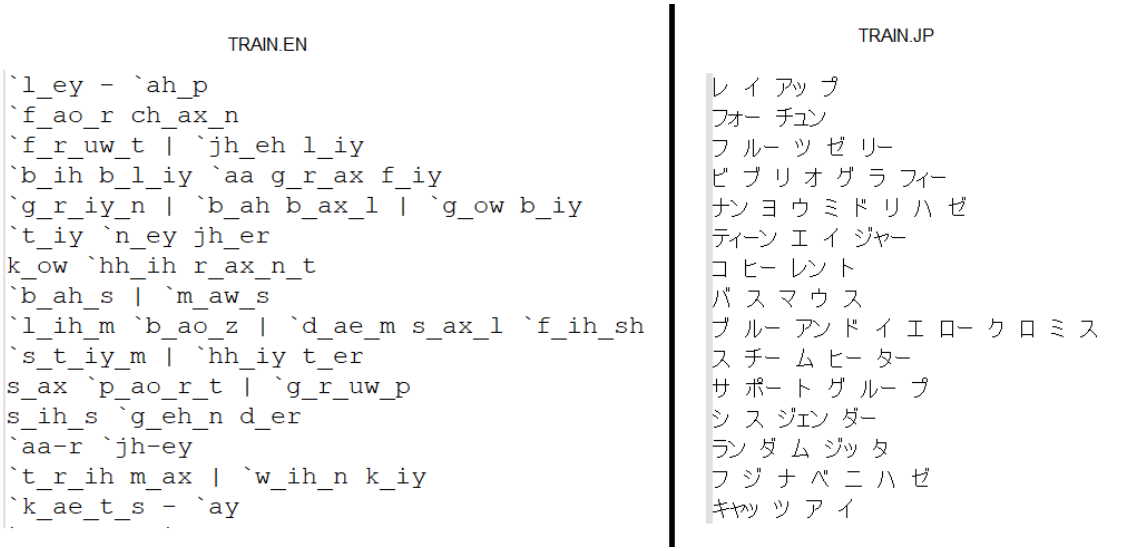


Figura 16. Muestra archivos train.en y train.jp

Con los archivos listos, dentro de la carpeta “train_v2” se ejecutan una serie de comandos detallados en [40] para realizar la alineación. Se tendrá como resultado múltiples archivos de nombre “src_trg.dict.A3.final” en donde se encontrará la estructura de alineación indicada en la Figura 17. En el archivo de configuración se definió la

cantidad de hilos que MGIZA usará y en el presente trabajo dicho número es 8. En base a este número existirán la misma cantidad de archivos A3, por lo que se deben unir los archivos para tener las alineaciones en su totalidad.

```
# Sentence pair (20) source length 3 target length 7 alignment score : 2.79707e-07
ス ロー プ ス タ イ ル
NULL ({ }) `s_l_ow ({ 1 2 }) p_ax ({ 3 }) `s_t_ay_l ({ 4 5 6 7 })
# Sentence pair (21) source length 3 target length 4 alignment score : 0.00438186
ジャン ク アー ト
NULL ({ }) `jh_ah_ng_k ({ 1 2 }) | ({ }) `aa_r_t ({ 3 4 })
# Sentence pair (22) source length 4 target length 5 alignment score : 0.000253005
シ ル キー シャー ク
NULL ({ }) `s_ih_l ({ 1 2 }) k_iy ({ 3 }) | ({ }) `sh_aa_r_k ({ 4 5 })
# Sentence pair (23) source length 3 target length 3 alignment score : 0.0120104
ト マ チン
NULL ({ }) `t_ow ({ 1 }) `m_ey ({ 2 }) `t_iy_n ({ 3 })
# Sentence pair (24) source length 4 target length 5 alignment score : 0.000420684
タ ブ レッ ト ペン
NULL ({ }) `t_ae ({ 1 }) b_l_ax_t ({ 2 3 4 }) | ({ }) `p_eh_n ({ 5 })
```

Figura 17. Muestra archivo A3 de alineación final

En la Figura 17 se observa una muestra de la alineación realizada. Para entender dicha alineación se trabaja en grupos de tres filas. A manera de ejemplo se explicarán las filas 1, 2 y 3. La primera fila contiene un identificador (20) seguido de la longitud (3) en cuanto a sílabas que se encuentran en el archivo “train.en” y de la longitud (7) de la secuencia katakana que será alineada perteneciente al archivo “train.jp”. De igual manera, cada alineación tiene su valor tras los cálculos que MGIZA realiza para llegar a dicha respuesta, que en este caso es de 2.79707e-07.

La segunda fila contiene a la secuencia katakana (ス ロー プ ス タ イ ル) en función del identificador. Mientras que la tercera fila se compone de múltiples subconjuntos, el subconjunto “NULL” indicará sí un carácter katakana no fue alineado correctamente. El resto de subconjuntos serán las sílabas fonéticas seguido de la posición del carácter katakana alineado a dicha sílaba. En el ejemplo se observa que la sílaba fonética “`s_l_ow” fue alineada con los katakanas en posición 1 “ス” (su) y 2 “ロー” (ro). Por su parte, la sílaba fonética “p_ax” ha sido alineada con el katakana “プ” (pu) pues este corresponde a la posición 3. Para finalizar, los katakanas de las posiciones 4 “ス” (su), 5 “タ” (ta), 6 “イ” (i) y 7 “ル” (ru) fueron alineados con la sílaba fonética “`s_t_ay_l”.

3.1.6. Reglas fonéticas permitidas

Tras obtener la alineación de las sílabas fonéticas con sus secuencias katakana, se deben generar reglas fonéticas que servirán para entrenar el HMM. La generación de

estas reglas se la realiza de manera manual y se basa puramente en la alineación ya obtenida.

Primero se deben relacionar los fonemas con uno o varios caracteres katakana. En el ejemplo anterior se observa que los katakanas "ス" y "ロー" están alineados con la sílaba fonética "`s_l_ow". Por ende, se puede relacionar al katakana "ス" con el fonema "`s" y a los katakanas "ロー" con los fonemas "l_ow". Es preferible generalizar lo más posible estas relaciones para poseer reglas más generales puesto que se trabaja con una data significativamente grande y se debe agilizar el procedimiento.

Para generar una regla permitida, lo más importante es que los caracteres katakana tengan una alineación válida con los fonemas de las sílabas fonéticas. Por ejemplo, el fonema /m/ de manera básica estaría alineado a los katakanas マ (ma), ミ (mi), ム (mu), メ (me), モ (mo). Sin embargo, es necesaria información estadística que confirme esto.

Así pues, se analizan las alineaciones realizadas por MGIZA y con dicho análisis es posible generar una regla fonética general para este conjunto de katakanas, la cual, representada en una expresión regular es *`?m\w*.* En la Tabla 5 se detallan las sílabas fonéticas en donde se han encontrado a los diferentes caracteres katakana.

Tabla 5. Alineaciones de マミムメモ

KATAKANA	SÍLABA FONÉTICA
マ	`m_ah / m_ax / `m_a / `m_ey / m_aw / `m_ae / `m_eh / m_e /
ミ	m_ax / `m_ih / `m_iy / m /
ム	m /
メ	`m_eh / m_ax / `m_iy / m_e / m / `m_ey /
モ	m_ax / `m_aa / m_ow / `m_ow / m_e / `m_ao / `m_ah /

Al tratarse de un proceso manual, solo se ha trabajado con una muestra de las sílabas fonéticas, puesto que sería un proceso muy largo analizar 6 102 sílabas fonéticas únicas de manera independiente cada una.

Desafortunadamente, no siempre es posible generalizar tanto y es necesario crear reglas adicionales que puedan cubrir de mejor manera las alineaciones de los katakanas. Tal es el caso del conjunto ツ (tsu), テ (te), ト (to), タ (ta), チ (chi), que, si bien en la Tabla 6 se ve que todos poseen al fonema /t/, existen excepciones como los fonemas

/d/, /z/ o /ch/ para los katakana タ y チ, por lo que se ha optado por generar una regla para cada uno.

Tabla 6. Alineaciones de ツテトタチ

KATAKANA	SÍLABA FONÉTICA
ツ	t / `t / t_s / t_ax / `t_uw / `t_uh /
テ	`t_eh / `t_e / `t_ay / t_ax / t_iy / `t_ey / t_ax / t_eh /
ト	t / `t_o / t_o / `t_ow / t_ow /
タ	t / t_a / t_er / `t_aev / `t_a / t_ax / `t_ae / t_e / dh_er / `t_ae / `t_ah /
チ	ch / s / t_ax / t_iy / `z_ih / t_ih / `t_w /

El proceso es el mismo para cada carácter katakana y en la Tabla 7 se da a conocer las reglas fonéticas permitidas utilizadas en el presente trabajo. Cabe recalcar que 4 684 (19%) palabras de la data de entrenamiento total (23 442 registros) no cumplen con las reglas fonéticas y han sido excluidas de procedimientos futuros.

Tabla 7. Reglas fonéticas permitidas

REGLA	SONIDO INGLÉS	ESCRITURA KATAKANA	EXPRESIÓN REGULAR
1	a - ya	アヤ	`?[aery]\w*
2	i	イ	`?[iy]|ax\w*
3	u	ウ	`?[auvw]\w*
4	e	エ	`?[aeiy]\w*
5	o	オ	`?[aow]\w*
6	ba - bi - bu - be - bo	バビブベボ	`?[bv]\w*
7	da - dji - ji - de - do	ダヂヅデド	`?d\w*
8	ga - gi - ge - gu - go	ガギグゲゴ	`?g\w*
9	fu	フ	`?[fh]\w*
10	ha - he	ハヘ	`?([fh]|ch)\w*
11	hi	ヒ	`?[fhak]\w*
12	ho	ホ	`?[fhw]\w*
13	ka - ki - ku - ke - ko	カキクケコ	`?k\w*
14	ma - mi - mu - me - mo	マミムメモ	`?m\w*
15	na - ni - un - ne - no	ナニヌネノ	`?n\w*
16	n	ン	`?(ng?|m)\w*
17	pa - pi - pu - pe - po	パピプペポ	`?p\w*
18	ra - ri - re - ru - ro	ラリルレロ	`?[rl]\w*
19	sa - su	サス	`?([sz]|th)\w*
20	shi	シ	`?([szy]|th)\w*
21	se - so	セソ	`?(s|th)\w*
22	tsu - te - to	ツテト	`?t\w*
23	ta	タ	`?[td]\w*
24	chi	チ	`?([tz]|ch)\w*
25	wa - wo	ワヲ	`?([wv]|er)\w*

26	yu - yo	ユ ヨ	`?(y|jh)\w*
27	za	ザ	`?([zsd])\w*
28	zu - zo	ズゾ	`?z\w*
29	ze	ゼ	`?([zs]|jh)\w*
30	ji	ジ	`?([dsz]|jh)\w*
31	ze - ge - ga	ゼゲガ	^`?jh\w*
32	nai	ナイ	`?(n_ay)|(n_ax)

3.2. Generación de diccionarios

Este proceso toma como entrada al archivo de alineaciones para que a través de las reglas fonéticas se pueda validar que la alineación se ha realizado correctamente. Primero se alinean aquellos caracteres katakana que no lograron ser alineados por medio de MGIZA en la Sección 3.1.5. Uno de estos casos se observa en la Figura 18 donde la línea 204 dentro del subconjunto “NULL” el katakana en posición 3 (イ) no pertenece a ninguna sílaba fonética.

```
202 # Sentence pair (404) source length 5 target length 5 alignment score : 1.77802e-06
203 ダ ラ イ ラ マ
204 NULL ({ 3 }) `d_ae ({ 1 }) `l_ey ({ 2 }) | ({ }) `l_aa ({ 4 }) m_ax ({ 5 })
205 # Sentence pair (1558) source length 2 target length 5 alignment score : 3.57092e-06
206 レ イ ア ウ ト
207 NULL ({ 2 5 }) `l ey ({ 1 }) `aw t ({ 3 4 })
```

Figura 18. Alineación MGIZA incorrecta

Gracias a las reglas fonéticas se procede a realizar una nueva alineación dando como resultado que el katakana “イ” pertenece a la sílaba fonética “`l_ey” como se aprecia en la Figura 19. Adicionalmente, puede ocurrir que MGIZA haya realizado una alineación incorrecta la cual puede ser o no corregida gracias a las reglas.

```
# Sentence pair (404) source length 5 target length 5 alignment score : 1.77802e-06
ダ ラ イ ラ マ
NULL ({ }) `d_ae ({ 1 }) `l_ey ({ 2 3 }) | ({ }) `l_aa ({ 4 }) m_ax ({ 5 })
```

Figura 19. Alineación correcta tras uso de reglas fonéticas

Una vez que se hayan realizado las nuevas alineaciones, es necesario validarlas para tener una data de entrenamiento limpia. La validación más importante es que las sílabas fonéticas hayan sido alineadas con una secuencia katakana prevista en el conjunto de reglas.

En la Figura 20 se presenta que la secuencia katakana “ナン ヨ ウ ミ ド リ ハ ゼ” está alineada con las sílabas fonéticas “`g_r_iy_n / `b_ah / b_ax_l / `g_ow / b_iy”. Sin embargo, está alineación es incorrecta puesto que el katakana “ナ” (na) no puede estar relacionado con el fonema /g/ en la primera sílaba fonética.

Este error ocurre debido a cómo se realizó la limpieza inicial del corpus EDICT2. La secuencia katakana “ナン ヨ ウ ミ ド リ ハ ゼ” escrita en alfabeto latino es “Nanyou Midori Haze”. Mientras que las sílabas fonéticas “`g_r_iy_n / `b_ah / b_ax_l / `g_ow / b_iy” corresponden a “green bubble goby”. A simple vista se puede observar que ambas expresiones son diferentes y, por ende, la alineación es errónea en su totalidad.

```
# Sentence pair (5) source length 7 target length 8 alignment score : 5.50108e-21
ナン ヨ ウ ミ ド リ ハ ゼ
NULL ({ }) `g_r_iy_n ({ 1 2 3 }) | ({ }) `b_ah ({ 4 }) b_ax_l ({ }) | ({ }) `g_ow ({ 6 7
}) b_iy ({ 8 })
```

Figura 20. Alineación incorrecta por limpieza incorrecta

Luego de la validación se generan los diccionarios de emisión y de transición que servirán de entrada para el HMM. Como se ha mencionado anteriormente, el diccionario de emisión relaciona una sílaba fonética del inglés con una secuencia katakana en donde además se especifica el inicio (<s>) y final (</s>) de una cadena tal como se observa en la Figura 21. Por ejemplo, la primera fila relaciona la sílaba fonética “`l_ey” con la secuencia katakana “レイ” la cual es la transliteración de la palabra en inglés “lay”.

```
<s> `l_ey レイ </s>
<s> `ah_p アップ </s>
<s> `f_ao_r フォー </s>
<s> ch_ax_n チュン </s>
<s> `f_r_uw_t フルーツ </s>
<s> `jh_eh ゼ </s>
<s> l_iy リー </s>
<s> g_r_ax グラ </s>
<s> f_iy フィー </s>
<s> `b_ih ビ </s>
```

Figura 21. Diccionario de emisión

Por su parte, el diccionario de transición posee en su interior secuencias de caracteres katakana que conforman una palabra o expresión del inglés junto con el inicio y final de una cadena. En la Figura 22 se muestra en la primera fila la transliteración de la palabra en inglés “lay up”.

<s> レイ アップ </s>
<s> フォー チュン </s>
<s> フルーツ </s>
<s> ゼ リー </s>
<s> ビ ブリ オ グラ フィー </s>
<s> ティーン エイ ジャー </s>
<s> コ ヒー レント </s>
<s> バス </s>
<s> マウス </s>
<s> スチーム </s>

Figura 22. Diccionario de transición

Adicionalmente, se separa en un archivo independiente a aquellas palabras que poseen una alineación incorrecta para realizar una segunda revisión manual. De igual manera, se separan aquellas palabras cuyas sílabas fonéticas no guardan relación alguna con la transliteración presente en la data de entrenamiento. Estas serán usadas para otra validación final posterior en la Sección 4.4 y es necesario que existan archivos diferentes para la data de entrenamiento y la data de pruebas.

3.3. ELABORACIÓN DE LOS HMM

3.3.1. Cálculo de probabilidades

Cabe recordar que existen dos dominios en un HMM, el observado (sílabas fonéticas del español/inglés) y el oculto (Katakanas). Una vez que los datos de los dos dominios se encuentren correctamente alineados, se obtiene la matriz cuadrada "A" de probabilidades de transición y una matriz no cuadrada "B" de probabilidades de emisión, donde A corresponde a una cadena de Markov [27]. Esta cadena es un proceso estocástico a tiempo discreto $\{X_n : n = 0, 1, \dots\}$, con espacio de estados discretos, y la cual satisface la propiedad de Markov que define un proceso estocástico, es decir, para cualquier entero $n \geq 0$, y para cualquier estado $x_0, \dots, x_{n+1}$, se cumple que $P(x_{n+1}|x_0, \dots, x_n) = P(x_{n+1}|x_n)$. En un proceso estocástico se asume que no hay necesidad de conocer el pasado en su totalidad para predecir el evento siguiente. Esto se debe a que este proceso toma en consideración acciones predecibles y elementos aleatorios [41].

Para el cálculo de probabilidades de ambas matrices se emplea un modelo de conteo de n-gramas. Estos son una subcadena de caracteres pertenecientes a una cadena más grande [42]. En este contexto se emplean unigramas (1 caracter) y bigramas (2 caracteres) de manera secuencial. En la Figura 22 tomando de ejemplo la primera línea se observan separados por un espacio en blanco a los unigramas “<s>”, “レイ”, “アップ” y “</s>”. Por otro lado, los bigramas surgen de la unión de dos unigramas secuenciales y son “<s> レイ”, “レイ アップ” y “アップ </s>”.

Con la data de entrenamiento se obtienen 5418 unigramas únicos dentro del diccionario de transición lo cual indica que existen 5416 secuencias katana diferentes, puesto que los 2 unigramas restantes son el inicio y final de cadena.

La matriz “A” se obtiene por medio del diccionario de transición. Primero se realiza el conteo de unigramas y se otorga un valor en base a la frecuencia con la que aparece. Por ejemplo, de la data completa, el unigrama “レイ” tiene un valor de 32. A continuación, se contabilizan los bigramas del mismo modo que se contó a los unigramas, así pues, el bigrama “レイ アップ” tiene un valor de 1. Con ambos valores se obtiene la probabilidad de que exista una transición desde el unigrama “レイ” hacia el unigrama “アップ” es decir, 1 dividido para 32 dando como resultado 0.03125.

La matriz “B” sigue el mismo procedimiento, pero utiliza al diccionario de emisión y en dicho contexto, los unigramas son la sílaba fonética del inglés y los bigramas corresponden a una sílaba fonética alineada con una secuencia katakana. De ese modo, con la primera línea de la Figura 21 de referencia, el unigrama “`l_ey” tiene valor de 129, mientras que el bigrama “`l_ey レイ” tiene 14 como valor. Siendo así su probabilidad igual la división de 14 para 129 con resultado de 0.10852.

Cabe señalar que todos los cálculos se los realizan con base logarítmica de 10. Por ende, la mayor probabilidad se encontrará en el rango de números negativos y va a tender a 0, mientras que el menor valor estará en el mismo rango, pero su tendencia será hacia -99. Los resultados anteriores pasan a ser -1.50515 (0.03125) y -0.96445 (0.10852).

Esto principalmente para evitar problemas donde la cantidad de los dígitos en una fracción no es suficiente para realizar los cálculos, o porque estos son extremadamente pequeños como para ser representados por la CPU o la memoria del computador

(*underflow*) [43]. Al trabajar con logaritmos, las multiplicaciones se convierten en sumas y las divisiones en restas, por lo que desaparece la posibilidad de una multiplicación o división para cero.

A raíz de la matriz "A" surge el vector de probabilidades inicial P_0. Sus componentes representan a cada uno de los estados ocultos y contiene la distribución de probabilidades de que la cadena de Markov se inicie en un estado oculto más probable sin considerar al unigrama de inicio de cadena.

4. EXPERIMENTACIÓN

4.1. Preparación de Viterbi

Para aplicar el algoritmo de Viterbi se emplea una librería disponible para Perl, la cual es "Algorithm::Viterbi" [44]. Esta permite encontrar el camino de predicción más probable dada una secuencia de observaciones de entrada y calcular el valor de la probabilidad de dicha predicción. Sin embargo, es necesario modificar la librería puesto que el algoritmo que esta emplea es diferente al indicado en la Sección 2.3.1.

La librería original omite la primera iteración donde se calcula las probabilidades en t=0 y donde no existen transiciones puesto que es el primer estado a predecir. En su lugar, desde el primer momento intenta predecir dos estados para una observación, el actual y el siguiente. Sus cálculos los realiza a partir de la segunda parte del algoritmo, desde t=1 en adelante, generando que el camino a predecir posea n+1 elementos de los que entran como observaciones. En los ejemplos de [44] y en la Figura 23 se entiende mejor esta situación. La tercera parte del algoritmo sí se encuentra en esta librería, pero realiza cálculos adicionales que en el contexto actual no son necesarios.

```
my $observations = [ 'walk', 'shop', 'clean' ];
 my $start = { 'Rainy'=> 0.6, 'Sunny'=> 0.4 };
 my $transition = {
    'Rainy' => {'Rainy'=> 0.7, 'Sunny'=> 0.3},
    'Sunny' => {'Rainy'=> 0.4, 'Sunny'=> 0.6},
    };

my $emission = {
   'shop' => {
     'Sunny' => '0.3',
     'Rainy' => '0.4',
   },

   'walk' => {
     'Sunny' => '0.6',
     'Rainy' => '0.1'
   },
   'clean' => {
     'Sunny' => '0.1',
     'Rainy' => '0.5'
     }
};
```

Produces:

```
$VAR1 = '0.033612';
$VAR2 = [
          'Sunny',
          'Rainy',
          'Rainy',
          'Rainy'
        ];
$VAR3 = '0.009408';
```

Figura 23. Ejemplo librería Viterbi inicial

La adaptación entonces consiste en añadir la primera iteración del algoritmo, modificar la segunda parte para considerar al estado actual y al estado anterior, y eliminar cálculos innecesarios de la tercera parte para conseguir el valor de la probabilidad del camino a predecir. Por último, dado que las probabilidades de transición y emisión, así como el vector inicial se encuentran en función de logaritmo de base 10, también se deben adaptar los cálculos y valores que la librería posee hacia este logaritmo.

La nueva librería modificada, bajo el nombre de “ViterbiLogV2”, necesita de variables de entrada las cuales son la matriz de probabilidades de transición, la matriz de probabilidades de emisión, el vector inicial y la secuencia de observaciones. Previamente se indicó que la matriz de transición necesitaba ser cuadrada pero que no se la había generado como tal. Esto es gracias a que la librería utilizada permite definir un valor a aquellas probabilidades que tienden a -99 y a aquellas que no existan debido a que la observación de la data de prueba no existe dentro de la propia matriz. Para validar dicha existencia, la librería debe poseer una copia de las probabilidades de emisión de la data de entrenamiento con la cual comparar.

Ambas matrices de probabilidades y el vector inicial se encuentran en archivos externos, por lo que la librería debe ser capaz de leerlos y utilizar los valores respectivos a las observaciones de entrada.

El principal cambio de la librería Viterbi se da dentro de la función “forward_viterbi” la cual es la encargada de generar la predicción y obtener la probabilidad de dicha predicción. En la parte izquierda de la Figura 24 se encuentra la librería Viterbi original y a la derecha está la modificada.

Se puede observar que el análisis ocurre considerando la observación ($observation), el estado actual ($state) y el estado anterior ($prev_state), ya no se trabaja con el estado siguiente ($next_state). Adicionalmente, se ha añadido un bloque *if* donde se realiza el primer cálculo en tiempo 0 y se ha reordenado dentro del bloque *else* a los cálculos para el tiempo 1 en adelante. Cabe recalcar que la variable $valmax ha cambiado de valer 0 a valer -2000 pues se está trabajando con logaritmos.

VITERBI

```
foreach my $output (@$observation) {
  my $U = { };
  foreach my $next_state (@{$self->{states}}) {
    my $total = 0;
    my $argmax = [ ];
    my $valmax = 0;
    foreach my $state (@{$self->{states}}) {
  my ($prob, $v_path, $v_prob) = @{$T->{$state}};

  my $e = $self->get_emission($output, $state);
  my $t = $self->get_transition($state, $next_state);

  my $p = $e * $t;
  $prob *= $p;
  $v_prob *= $p;
  $total += $prob;

  if ($v_prob > $valmax) {
    $argmax = [ @$v_path, $next_state ];
    $valmax = $v_prob;
  }
    }
    $U->{$next_state} = [ $total, $argmax, $valmax ];
  }
  $T = $U;
}
```

VITERBI MODIFICADO

```
foreach my $output (@$observation) {
    my $U = { };
    my $argmax;
    my $valmax = -2000;
    my $total = 0;

    foreach my $state (@{$self->{states}}) {
      if ($emision_katakanas{$output}{$state}) {
        $total = 0;
        my $prev_state = '';
        my $e = $self->get_emission($output, $state);
        my $t = 0;
        my $p;

        if ($contador == 1) {
          my ($prob_ini) = @{$T->{$state}};
          $p = $prob_ini + $e;
          $total += $p;
          if ($total > $valmax) {
            $argmax = $state;
            $valmax = $total;
          }
        }else{
          $valmax = -2000;
          foreach $prev_state (@{$self->{states}}) {
            if ($prev_state eq $prev_katakana) {
              my ($prob_ini) = @{$T->{$prev_state}};
              $t = $self->get_transition($prev_state, $state);
              $p = $prob_ini + $t;

              if ($p > $valmax) {
                $valmax = $p;
                $total = $p + $e;
              }
            }
          }
        }
        $U->{$state} = [ $total ];
        }
    }
}
```

Figura 24. Comparación de librería Viterbi

4.2. Ejecución del programa

Primero se debe tener preparado el archivo A.3 perteneciente a la lineación realizada por MGIZA. Con él se ejecuta el archivo Perl “03-silabas” que utilizará las reglas fonéticas permitidas para alinear aquellas palabras que MGIZA no pudo. Tras esta nueva alineación se procede con la validación de las sílabas fonéticas por medio del archivo Perl “03-validacion” y empleando una vez más las reglas mencionadas anteriormente. En este punto también se han generado los diccionarios de emisión y de transición, así como se han separado las palabras y las sílabas fonéticas no válidas.

El siguiente paso está dividido en dos archivos Perl, en el primero de nombre “04-calculo-transición” se obtiene la matriz A de probabilidades de transición y el vector de probabilidades iniciales P_0. Por otro lado, utilizando el archivo “04-calculo-emision” se genera la matriz B de probabilidades de emisión.

El siguiente paso consiste en obtener los resultados por medio del algoritmo de Viterbi. El programa utiliza el archivo Perl “05-viterbi-jp-2” en el que se tienen como entradas a los archivos de probabilidades de emisión y transición junto con el archivo del vector inicial. Adicionalmente recibe las observaciones (Figura 25) de las sílabas fonéticas de la data de pruebas a las que se predecirá su transliteración al katakana más probable.

En este archivo se utiliza la librería de Viterbi modificada de nombre “ViterbiLogV2.pl” y el cálculo se realiza de igual manera que en la Sección 2.3.1. Los resultados en cuestión se observan en la Figura 26.

```
<s> `n_ow `hh_ih_t `n_ow `r_ah_n `g_ey_m </s>
<s> `s_ay `d_w_ao_l </s>
<s> `jh_ih_ng `g_ow ih z_ax_m </s>
<s> ax `p_ow n_ax_n_t_s `hh_ae v_ih_ng `w_ah_n `ae_n `iy k_w_ax_l `n_ah_m b_er `ah_v
`s_eh_t_s </s>
<s> f_er f_r_ax_l </s>
<s> `s_ay_d `s_uw `p_l_eh_k_s </s>
<s> `hh_ih s_t_ax `d_iy_n </s>
<s> `s_n_ae_p `g_ey_jh </s>
<s> `d-iy `s-iy `iy </s>
<s> `jh_uw n_y_er `hh_ay `s_k_uw_l </s>
<s> `m_ey t_r_ih_k_s `s_w_ih ch_er </s>
<s> `p_ow z_ih_ng </s>
<s> `ae k_r_ax `b_ae_t </s>
<s> `k_ay z_er `r_ow_l </s>
<s> `b_l_ae_k `s_p_aa t_ih_d `k_ae_t `sh_aa_r_k </s>
```

Figura 25. Observaciones de data de pruebas

```
<s> ノー ヒット ノー ラン ゲーム </s>
<s> サイ </s>
<s> ジン ゴ イ ズム </s>
<s> ア ポ ハ ピング ワン アン イー コール ナン バー オブ </s>
<s> ファ </s>
<s> サイド スー プレックス </s>
<s> ハイ スタ ジン </s>
<s> スナップ ゲージ </s>
<s> ディー シー エー </s>
<s> ジュ ニア ハイ スクール </s>
<s> マ トリクス スイ チャー </s>
<s> ポ ジング </s>
<s> ア クロ バット </s>
<s> カイ ゼル ロール </s>
<s> ブラック スポッ ト キャット シャーク </s>
```

Figura 26. Resultados del programa

Para ejemplificar el proceso de predicción de secuencias katakana más probable se utilizará el registro 12 de la Figura 25 y la Figura 26. Las sílabas fonéticas son “<s> `p_ow z_ih_ng </s>” y corresponden a la palabra en inglés “posing”. El resultado que el programa arrojó fue la secuencia katakana “<s> ポ ジング </s>” alineando el inicio y fin de cadena de manera idéntica, la sílaba fonética “`p_ow” está alineada con “ポ (po)” y a la sílaba fonética “z_ih_ng” le corresponde “ジ (ji) ン (n) グ (gu)”.

A continuación, sólo se mostrarán los cálculos para la cadena predicha más probable y estos están con valores logarítmicos, por ende, la probabilidad es mayor cuando el valor tiende a 0. En la Figura 27 se detallan las probabilidades utilizadas. Cabe recalcar que las flechas que parten del inicio hacia los estados katakana corresponden a las probabilidades del vector inicial. Las flechas que unen los estados katakana (dominio

oculto) entre sí son las probabilidades de transición y las flechas entre cortadas que unen un estado katakana con un estado de sílaba fonética del inglés (dominio observable) son las probabilidades de emisión. Los cálculos siguen la lógica de la Sección 2.3.1.

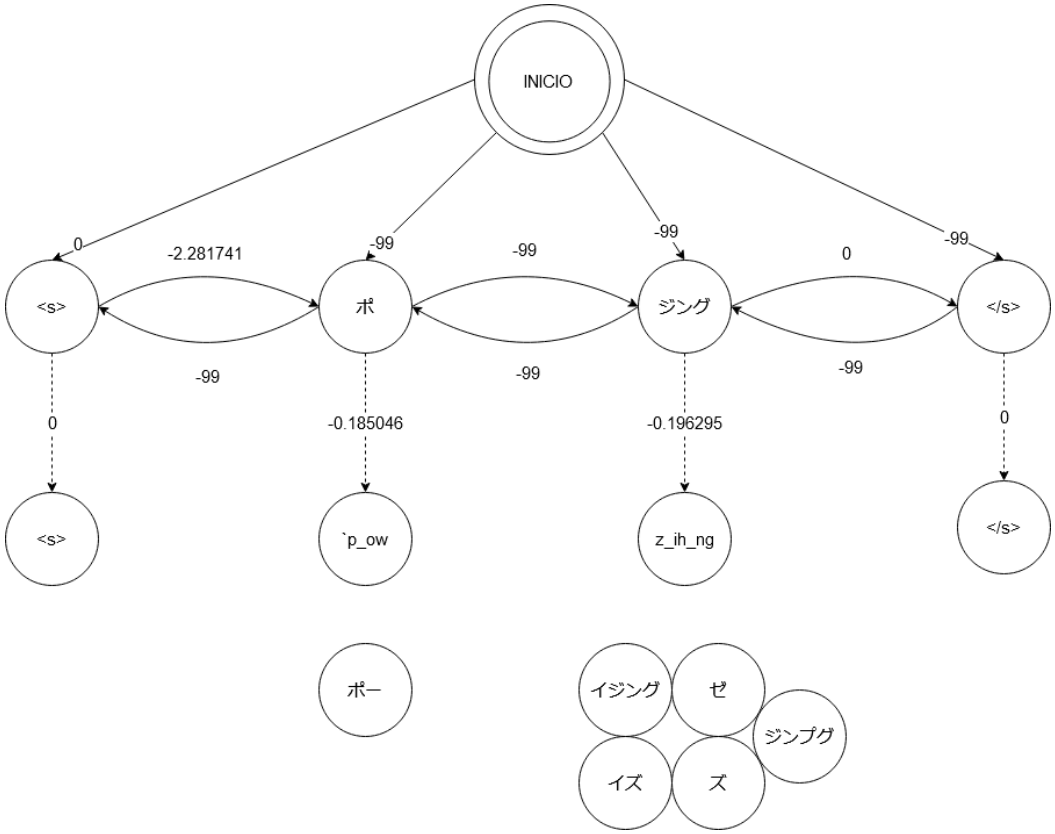


Figura 27. Probabilidades predicción de "posing"

La probabilidad de que la secuencia katakana sea “<s>” dado que la observación es “<s>” necesita de la probabilidad de emisión (0) y la probabilidad de inicio (0) dando como probabilidad final 0 en el primer estado, tras este paso, la probabilidad de inicio es indiferente y de hecho se observa que salvo “<s>” todas tienen como valor -99, esto se debe a que en el programa siempre se inicia con el símbolo de inicio de cadena y por ello su probabilidad es 0.

La siguiente observación es “`p_ow” y con ella existen dos secuencias katakana probables, la primera es “ポ” y la segunda “ポー”. Aquella con mayor probabilidad resultó ser “ポ” al sumar la probabilidad del estado anterior (0) con la probabilidad de transición de “<s>” a “ポ” (-2.281741) y a esa respuesta sumarle la probabilidad de emisión respectiva (-0.185046) dando como resultado una probabilidad de -2.466787 en este

estado. Como acotación, la probabilidad con la otra secuencia katakana es de -3.705276 que evidentemente es menor a la anterior.

Con respecto a la observación “z_ih_ng”, el programa calculó que la mayor probabilidad la tiene la secuencia katakana “ジング”. La probabilidad se obtiene de la suma del estado anterior (-2.466787) con la probabilidad de transición de “ポ” hacia “ジング (-99), este valor indica que esta transición no existe en la matriz A. Por último, a dicho valor se le añade la probabilidad de emisión (-0.196295) teniendo así un resultado final para este estado de -101.663082. El resto de los valores para las secuencias katakana son:

- イジング (-102.985301).
- ゼ (-102.985301).
- ジンプグ (-102.985301).
- イズ (-102.50818).
- ズ (-102.20715).

Por último, para la observación de fin de cadena “</s>”, de igual manera se suma la probabilidad del estado anterior (-101.663082) con la probabilidad de transición que va desde “ジング” hacia “</s>” (0), que tenga este valor significa que “ジング” solo tiene una transición y esa es hacia “</s>”. Finalmente se suma la probabilidad de emisión (0) dando como resultado -101.663082 que a su vez es la probabilidad final de la predicción realizada. No existen más cálculos puesto que el carácter de fin de cadena no tiene otras secuencias posibles.

4.3. Errores y consideraciones adicionales

El procedimiento no está exento de errores y la gran parte de ellos se encuentran relacionados con la alineación de las sílabas fonéticas del inglés con una secuencia katakana.

4.3.1. Limpieza de EDICT2

El proceso de limpieza tiene fallos tanto en el apartado de las secuencias katakana como de las expresiones en inglés. Sin embargo, es en este último donde más se evidencian los fallos. Se han identificado varios casos donde ocurren los fallos, aunque es probable que existan otros más pues se está trabajando con una data grande.

- El más evidente es cuando la limpieza del apartado en inglés no tiene relación alguna con la transliteración al katakana. En la Figura 28 se observa en la parte superior que el registro 97 de la limpieza relaciona a la secuencia katakana ラブコール con una larga frase. Mientras que en la parte inferior dentro del registro 47 329 de EDICT2 se observa la transliteración correcta que debería ser “love call”.

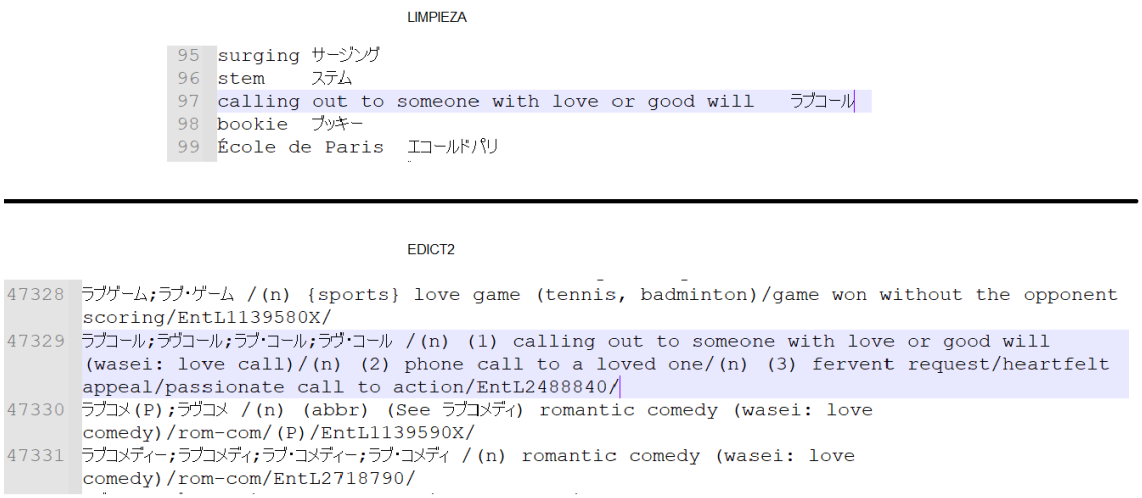


Figura 28. Comparativa de transliteración de ラブコール

- Otro caso sucede cuando la secuencia katakana hace referencia a una abreviación de la expresión en inglés. Tal es el caso del registro 35 809 de la Figura 29 pues allí se identifica que la secuencia katakana パトカー corresponde a la abreviación de パトロールカー que viene del inglés *patrol car.* Afortunadamente, el registro 35 819 tiene la transliteración correcta, pero el hecho de que este primer registro exista ya genera cierto ruido en la limpieza.

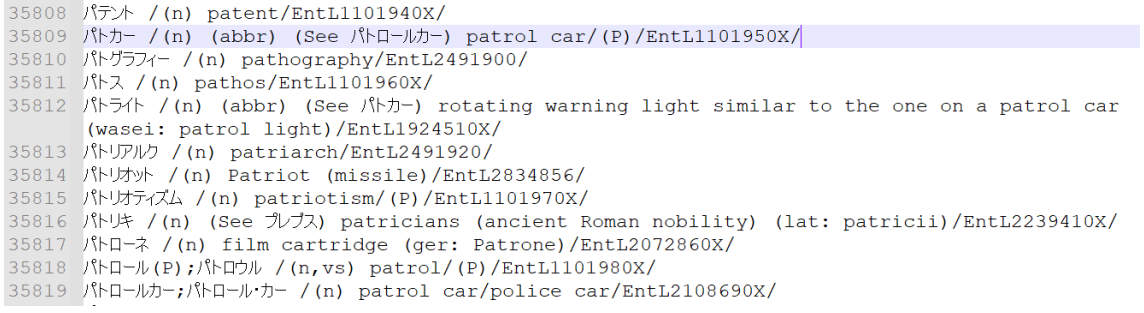


Figura 29. Registros de “patrol car” en EDICT2

- Por último, hay un caso que ocurre cuando un registro de EDICT2 no posee una transliteración adecuada. Para ello se retoma el error de la secuencia katakana “ナンヨウミドリハゼ” que fue alineada con “green bubble goby”, pero cuya transliteración al inglés es “Nanyou Midori Haze”. Cuando se observa el diccionario original en la fila 32 400 de la Figura 30, se identifica que la transliteración esperada nunca formó parte del diccionario y por ende la limpieza fue realizada correctamente. Es posible que este registro se trate de una traducción cuando se toma en cuenta que “Midori” significa verde en japonés que a su vez es “green” en inglés.

```
32399 ナンヨウハギ属 [ナンヨウハギぞく] /(n) Paracanthurus (genus in the family Acanthuridae whose sole
      member is the palette surgeonfish)/EntL2557420/
32400 ナンヨウミドリハゼ /(n) green bubble goby (Eviota prasina)/pepperfin pygmy goby/EntL2549070/
32401 ナヴァラン;ナバラン /(n) {food} navarin (fre:)/EntL2831171/
32402 ナ形容詞;な形容詞 [ナけいようし(ナ形容詞);なけいようし(な形容詞)] /(n) (esp. in JSL contexts) (See 形容動詞)
      na-adjective/adjectival noun/EntL2421310X/
```

Figura 30. Registro de ナンヨウミドリハゼ en EDICT2

4.3.2. Alineación incorrecta de MGIZA

Este error viene relacionado con el error anterior. Principalmente porque sí la data previa que sirve de base para realizar las alineaciones está incorrecta, por más que MGIZA lo intente no se podrá tener una alineación válida.

Esto a su vez desenvuelve en que las sílabas fonéticas estén mal alineadas y por lo tanto no todas puedan ser tomadas en cuenta para la generación de los diccionarios. Originalmente, la data de entrenamiento poseía 6 102 sílabas fonéticas, de las cuales se utilizan 5 418 (88%) dejando el valor restante (12%) inutilizable.

4.3.3. Múltiples transliteraciones validas

Este apartado no es un error como tal, pero es necesario ponerlo en consideración. En el diccionario EDICT2 se puede observar que para una misma expresión en inglés pueden existir diversas transliteraciones válidas al katakana. Esto mismo puede ocurrir cuando el programa predice la secuencia katakana más probable ya que está influenciado por las probabilidades de emisión. Más adelante, con la Figura 31 se observará que para una sílaba fonética existen múltiples secuencias katakana y el programa se decanta por la de mayor probabilidad, aunque, dependiendo del contexto, esta puede no ser la más adecuada.

4.4. Validación final

Esta validación utiliza las sílabas fonéticas no válidas obtenidas en la Sección 3.2 y está pensada principalmente para el análisis de resultados. Como ya se mencionó anteriormente, debe existir un archivo diferente para la data de entrenamiento y otro para la data de pruebas. Esto se debe a que cada data posee los errores de la Sección 4.3 y por ende son diferentes las palabras y sílabas fonéticas no válidas a analizar en cada caso.

La validación consiste en extraer de los resultados finales a aquellas palabras que posean sílabas fonéticas no válidas según los errores de la sección anterior y, por ende, no sería correcto que estas formen parte de los resultados oficiales. se realiza con el archivo Perl “06-validacion”.

5. RESULTADOS Y DISCUSIÓN

5.1. Data de entrenamiento y prueba

De manera manual se ha observado que la mayor parte de las repuestas del programa son extremadamente cercanas a la transliteración original presente en EDICT2, simplemente existen diferencias en pocos caracteres katakana de la secuencia en cuestión.

Por ejemplo, la palabra “ebonite” (polímero volcánico), cuyas sílabas fonéticas son “`eh b_ax `n_ay_t”, tiene como transliteración original a la secuencia katakana “エボナイト”, mientras que la respuesta del programa es “エバナイト”. Esta respuesta es muy parecida a la original y a simple vista se identifica que el segundo carácter katakana es el que difiere de una respuesta a la otra.

Esta diferencia ocurre principalmente por las probabilidades de la matriz de emisión. En la Figura 31 se detallan las probabilidades de la sílaba fonética “b_ax” donde se observa que tanto el katakana “ボ” como “バ” forman parte de la matriz, pero que aquella con mayor probabilidad es precisamente el katakana “バ”. Por ello el programa ha optado por dicho carácter como el más probable a formar parte de la predicción.

```
b_ax,ベ,-1.501744
b_ax,ビ,-0.626682
b_ax,ボー,-2.103804
b_ax,ボ,-0.899684
b_ax,ブ,-1.103804
b_ax,バ,-0.290890
b_ax,バー,-2.103804
```

Figura 31. Probabilidades de emisión para sílaba fonética b_ax

Considerando que en el ejemplo solo uno de cinco caracteres que forman la secuencia katakana está incorrecto, se ha decidido realizar un análisis en función de umbrales considerando la cantidad de katakanas diferentes como su métrica. Este análisis se lo realiza utilizando la librería “Text::Levenshtein” [45] permitiendo encontrar la distancia de Levenshtein (Sección 2.5) entre dos cadenas de caracteres.

Se han delimitado cuatro umbrales en donde se da por correcta a la predicción del programa cuando exista cierta cantidad de katakanas diferentes. El primer umbral sólo acepta secuencias katakana idénticas entre el original y el resultado del programa. El segundo umbral acepta uno o menos katakanas diferentes. El tercer umbral permite dos o menos diferencias y el cuarto umbral da por correcta a la secuencia cuando existen tres diferencias o menos. La secuencia del ejemplo se encontraría en el umbral dos.

Así pues, cuando se utilizan los 23 442 registros de la data de entrenamiento, el primer umbral tiene un porcentaje de secuencias katakana del 47,27%. Para el umbral número dos el porcentaje incrementa a 60,50%. Por su parte, el tercer umbral llega al 70,69% mientras que el último umbral posee un 78.57% de secuencias katakana correctas.

Adicionalmente, una vez identificados los errores en la Sección 4.3 y tras la validación final de la Sección 4.4, se ha realizado otro análisis utilizando los mismos umbrales. En este segundo análisis, se han identificado dentro de la data de entrenamiento a 18 371 registros utilizables. Los nuevos porcentajes son 58,59% en el primer umbral, 74,30% en el siguiente, 85,35% para el tercer umbral y 92,28% al final. Por su lado, en la Figura 33 se encuentra la evolución de la cantidad de predicciones correctas en función del umbral respectivo.

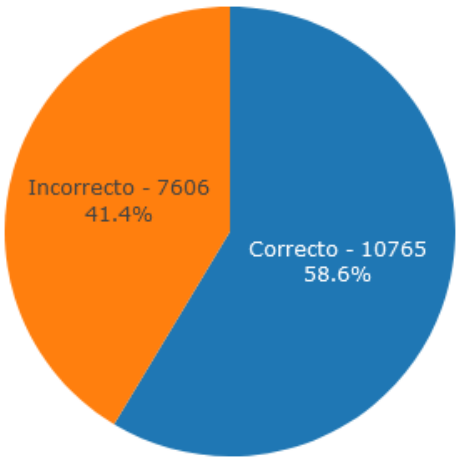


Figura 32. Porcentaje data de entrenamiento umbral 1

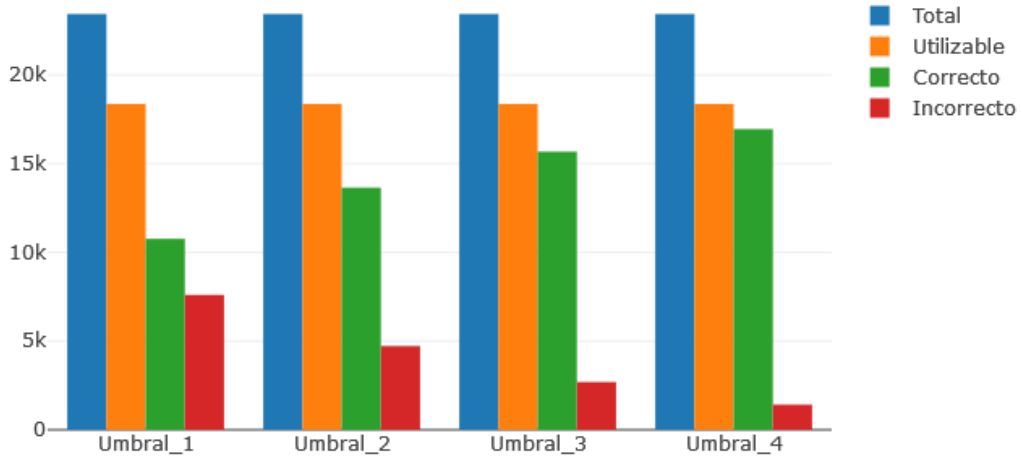


Figura 33. Evolución de predicciones correctas en data de entrenamiento

Para la data de prueba el análisis es el mismo y se tienen los siguientes resultados. De las 10 047 observaciones de entrada, aquellos que sí son posibles utilizar tras la validación final son 7 980 registros. El primer umbral tiene un 40,2% correcto, el segundo umbral es del 58,43%, el porcentaje del tercer umbral es 73,03% y el último umbral posee 84,76% de predicciones aceptadas. De igual manera, en la Figura 35 se encuentra la evolución de cantidad de predicciones aceptadas en función de los umbrales de la data de prueba.

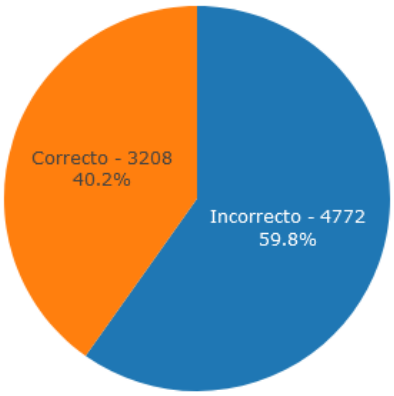


Figura 34. Porcentaje data de prueba umbral 1

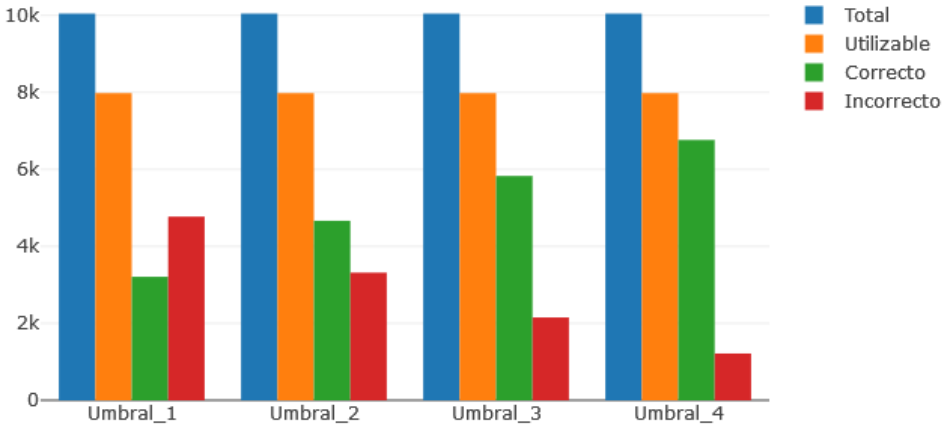


Figura 35. Evolución de predicciones correctas en data de prueba

5.2. English to katakana converter tool

Esta es una herramienta web [46] que ofrece múltiples características que relacionan el inglés con el japonés. Entre ellas, se encuentra disponible en la dirección https://www.sljfaq.org/cgi/e2k.cgi una herramienta que realiza la conversión de palabras del inglés hacia una secuencia katakana. Esta herramienta posee sus propias reglas de conversión [47] las cuales están basadas en los sonidos del inglés al momento de pronunciar una palabra. Dichos sonidos están representados en su forma proveniente al Alfabeto Fonético Internacional (IPA por sus siglas en inglés).

Por limitaciones de la herramienta, principalmente debido a que la cantidad de conversiones que permite realizar de manera simultánea bordea las 20 palabras, se ha optado por utilizar una muestra de 100 palabras para el análisis. Estas palabras provienen de la data de prueba y han sido seleccionadas de manera aleatoria. Tras la

limpieza de palabras no válidas se obtienen los siguientes resultados presentes en la Taba 8 para los diferentes umbrales.

Tabla 8. Comparativa de resultados

Total	100			
Utilizable	76			
	UMBRAL			
	1	**2**	**3**	**4**
PROGRAMA	23	33	42	57
	30,26%	43,42%	55,26%	75%
HERRAMIENTA	31	46	57	69
	40,78%	60,52%	75%	90,78%

5.3. Discusión

Considerando los errores y consideraciones de la Sección 4.3, a continuación, se plantean posibles soluciones.

- Realizar una mejor limpieza del diccionario EDICT2. Para ello se debería realizar un análisis para identificar correctamente cómo se encuentran las expresiones en inglés dentro del diccionario. Previamente se indicaron tres casos donde la incorrecta limpieza afectó al proceso de alineación, sin embargo, es posible que dichos casos no sean los únicos y, por ende, este análisis adicional puede servir de ayuda.
- Las alineaciones de MGIZA dependen de que las entradas en inglés y en katakana estén correctas y guarden relación una con la otra, sin embargo, existen casos donde la alineación no se ha realizado correctamente. Para reducir estos posibles errores, entran en juego las reglas fonéticas permitidas, las cuales, durante la generación de los diccionarios de emisión y transición, ayudan a validar los resultados de MGIZA, así como permiten alinear lo que no se pudo. La creación de reglas, a su vez, depende meramente de la primera alineación por lo que se entra en un proceso de mejora continua entre las reglas y MGIZA. La cantidad de sílabas fonéticas no válidas puede disminuir durante este proceso.
- Cabe recordar que el programa genera la predicción de secuencias katakana más probable dada una serie de observaciones de entrada. Sin embargo, eso no quiere decir que realmente sea la correcta o la más adecuada como ocurrió con la palabra en inglés “ebonite”. Para mejorar las respuestas del programa, una buena alternativa es considerar aquellas otras posibles predicciones para

que, por medio de un nuevo análisis (automático o manual) generar nuevos y mejores procesos de selección de los caracteres katakana que complementen a los resultados obtenidos de manera probabilística.

La suma de los errores mencionados anteriormente puede ser una de las causas por las que el programa tiene un menor porcentaje de aciertos en cada uno de los umbrales con respecto a la herramienta web *English to Katakana Converter Tool*. Definitivamente la herramienta genera mejores resultados, y esto puede estar relacionado con las reglas de transliteración que maneja. Esta trabaja en función de los sonidos y su representación con la forma IPA, a diferencia del programa donde se emplea el alfabeto latino para representar los fonemas. Cabe señalar que utilizan 43 reglas en comparación de las 32 que el programa emplea, y que sus reglas no están tan generalizadas para funcionar con múltiples caracteres katakana (salvo casos como la “r” o la “l”). La herramienta muestra que es posible empezar un análisis a partir de letras y sonidos básicos para llegar a secuencias katakana válidas.

Con esta información presente, es altamente probable que, para realizar una transliteración válida y adecuada, sea preferible realizar un análisis con una granularidad menor que utilizar sílabas fonéticas. Sin embargo, considerando que el katakana es un silabario de la escritura japonesa, quizás realizar una transliteración que considera una sola sílaba fonética es contraproducente, podría ser necesario emplear sílabas anteriores o siguientes para trabajar con trigramas y obtener estimaciones con mayores probabilidades.

Por una parte, es positivo trabajar con sílabas fonéticas que contengan una secuencia katakana extensa, pues se tiene más contexto, la cantidad de variaciones para dicha sílaba fonética es reducida y las secuencias son más directas. Pero, por otro lado, en los casos donde las sílabas fonéticas son pequeñas o de aquellas que no se encuentren dentro de los diccionarios, la secuencia katakana obtenida puede ser errónea o no la más adecuada.

Otro punto para considerar son las reglas fonéticas permitidas, que como se ha mencionado anteriormente, tienen área de mejora. Además, es importante tomar en cuenta cuando sería buena idea generalizarlas y cuando no hacerlo. Posiblemente existan casos donde sea mejor generar reglas independientes para cada uno de los fonemas que conforman la sílaba fonética.

La tónica general en el cálculo de la probabilidad de la predicción más probable dada una serie de observaciones, es que sea una probabilidad baja. Recordando el ejemplo de la Sección 4.2 en donde las probabilidades fueron 0 tras la primera observación,

pasando por -2.466787 en la segunda observación y finalizando con -101.663082 como probabilidad final. Nos encontramos ante la teoría básica de la probabilidad en la que se menciona que, al inicio de un análisis probabilístico, la probabilidad de un evento va a ser alta pero que, con el paso de los experimentos, dicha probabilidad llegará (generalmente disminuyendo su valor) a un punto considerado como la probabilidad verdadera tal como se explica en [48] utilizando un ejemplo básico de lanzar un dado al aire. Los cambios abruptos de probabilidad se dan principalmente debido a que en la data de entrenamiento no se encuentran registros que satisfagan con las nuevas observaciones a transliterar.

6. OPTIMIZACIÓN

6.1. Datos y almacenamiento

La primera gran mejora está presente al momento de generar las matrices de probabilidades. Recordando que en la data de entrenamiento se encontraron m=5418 unigramas únicos del dominio oculto y que la matriz de transición debe ser una matriz cuadrada, la generación de las probabilidades tendría 29 354 724 entradas con un espacio de almacenamiento de 990 MB aproximadamente. Del mismo modo, para las probabilidades de emisión se emplean los mismos m=5418 unigramas junto con los n=5250 bigramas dando un total de 28 444 500 entradas en la matriz de emisión con un almacenamiento aproximado de 880 MB.

La optimización en cuestión consiste en no considerar todas las combinaciones posibles con los unigramas y bigramas que se posee. En cambio, se opta por generar dos matrices de m*n elementos donde únicamente se toman en cuenta aquellas combinaciones que sí sean posibles gracias a la data de entrenamiento. Por ejemplo, en la matriz de transición y en función de la data, a la secuencia katakana “ストル” únicamente le pueden seguir las siguientes tres secuencias katakana “メン”, “メント” y “</s>” de este modo, se están descartando 5414 unigramas cuyo valor no tiene relevancia ya que tiende a -99. Así pues, la cantidad de datos disminuyó a 22 187 registros y el almacenamiento ahora es de 564 KB.

Con la matriz de emisión ocurre algo similar puesto que en función de la data se tiene que, la sílaba fonética “k_w_ax” tiene una correspondencia únicamente con las secuencias katakana “クア”, “キ” y “クイ”. Sin embargo, en esta matriz sí es posible que

una secuencia katakana provenga de diferentes sílabas fonéticas, en el caso de la secuencia katakana “クア” está presente en las sílabas “k_w_ax”, “`k_w_ey” y “`k_w_eh” cada una con sus respectivas probabilidades. Ahora, se tienen 10 012 datos y 276 KB de almacenamiento.

Tabla 9. Optimización de entradas y almacenamiento

	NO OPTIMIZADO		OPTIMIZADO	
	TRANSICIÓN	**EMISIÓN**	**TRANSICIÓN**	**EMISIÓN**
NUM ENTRADAS	29 354 724	28 444 500	22 187	10 012
ALMACENAMIENTO	900 MB	880 MB	564 KB	276 KB

Dentro del código, esta mejora se ve reflejada en dos bloques *foreach* que están presentados en la Figura 36. El primer bloque originalmente marcaba aquellos bigramas con combinaciones inexistentes en la data de entrenamiento y que por ende su valor tendía a -99. Un ejemplo de ello puede ser el bigrama que une las secuencias katakana “スト ル” y “ウィン” pues como ya se explicó, esta no existe.

Dentro del segundo bloque *foreach* se realizaba el cálculo de las probabilidades y en los casos donde el bigrama tenía -99 como valor, se marcaba a su probabilidad con -99 de igual manera. Estos dos cálculos ya no son necesarios y por ende se los puede excluir del código, incluso, el primer bloque podría ser eliminado.

```
foreach $llave_1 (keys %unigrama){
    foreach $llave_2 (keys %unigrama){
        if (!exists $bigrama{$llave_1}{$llave_2}) {
            #$bigrama{$llave_1}{$llave_2} = -99;
        }
    }
}

foreach $llave_1 (keys %bigrama){
    foreach $llave_2 (keys %{$bigrama{$llave_1}}){
        if ($bigrama{$llave_1}{$llave_2} == -99) {
            #$bigrama_prob{$llave_1}{$llave_2} = -99;
        }else{
            my $numerador = &log_base_n(10, $bigrama{$llave_1}{$llave_2});
            my $denominador = &log_base_n(10, $unigrama{$llave_1});
            $bigrama_prob{$llave_1}{$llave_2} = $numerador - $denominador;
        }
    }
}
```

Figura 36. Bloques foreach optimizados

6.2. Tiempo de ejecución

La segunda gran optimización fue posible gracias a la librería Viterbi [44]. En ella es posible definir el valor de probabilidades de transición y de emisión que no existan dentro de las matrices utilizadas y, por ende, al ser inexistentes, se les define el valor de -99.

Esto principalmente se debe a que la data de prueba tiene sílabas fonéticas y secuencias katakana que no aparecieron en la data de entrenamiento. En caso de que no se definan estas probabilidades desconocidas, el programa analiza a la observación de entrada con todas las probabilidades de emisión y de transición para obtener el camino de predicción más posible y valor más cercano a 0. Sin embargo, al trabajar con una gran cantidad de datos conocidos y múltiples observaciones desconocidas, es posible que dicha observación sea alineada de diferente manera con cada nueva ejecución del programa.

Para identificar la mejora en tiempo de ejecución se han realizado múltiples pruebas con diferente tamaño de muestras. En la Tabla 10 se indica un resumen de dichas pruebas. Primero, utilizando once observaciones de la data de entrenamiento, el tiempo de ejecución sin optimizar es de 15 minutos con 6 segundos, mientras que con la optimización el tiempo es de 1 segundo. De igual manera, utilizando once observaciones de la data de prueba se observa que el tiempo de ejecución sin la optimización es de 19 minutos y 35 segundos, mientras que con la optimización es de 1 segundo.

Por último, para las 23 442 observaciones de la data de entrenamiento, el programa optimizado se demora 12 minutos y 51 segundos. Sin embargo, por limitaciones de hardware, no se ha podido completar una ejecución del programa con las mismas observaciones y sin optimizar, por lo que con de una regla de tres en función de las once observaciones previas, se ha estimado que dicha ejecución tomaría 31 966 minutos, lo que se traduce aproximadamente a 532 horas. Ocurre lo mismo al ejecutar las 10 0447 observaciones de la data de prueba donde sus tiempos son 6 minutos con 9 segundos tras la optimización, y por regla de tres con las once observaciones ya ejecutadas, 289 horas aproximadamente sin tener la optimización.

Tabla 10. Optimización en tiempo de ejecución

	ENTRENAMIENTO		PRUEBA	
	NO OPTIMIZADO	**OPTIMIZADO**	**NO OPTIMIZADO**	**OPTIMIZADO**
OBSERVACIONES	11	11	11	11
TIEMPO	15m 6s	1s	19m 35s	1s
OBSERVACIONES	23 442	23 442	10 047	10 047
TIEMPO	532h (aprox)	12m 51s	289h (aprox)	6m 9s

7. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

El programa desarrollado por medio de los HMM brinda una alternativa que puede servir como base o punto de partida para resolver problemas de transliteración de palabras en inglés hacia secuencias katakana. Del mismo modo confirma que la transliteración por medio del uso de la conversión G2P brinda resultados aceptables.

Al utilizar sílabas fonéticas del inglés y al trabajar con secuencias katakana grandes, se tienen resultados más directos y sin muchas variaciones ni otras posibles predicciones. Por otra parte, su aplicación se encuentra muy generalizada y esto ha ocasionado que, al necesitar secuencias katakana más pequeñas o con estructuras más específicas, se obtengan resultados erróneos o no adecuados. Este es el principal motivo por el que se tiene un 59,8% de transliteraciones incorrectas.

El correcto uso del diccionario EDICT2 tiene gran impacto al momento de realizar las alineaciones que sirven de estructura básica para la generación de las probabilidades de los HMM. La pérdida del 12% de sílabas fonéticas por motivo de una incorrecta alineación, pesa en exceso en la propia generación de probabilidades y, por ende, en la predicción de las secuencias katakanas más probables.

La optimización de los recursos generados y de las herramientas utilizadas es de suma importancia durante el desarrollo de cualquier programa. Esta importancia se ve aumentada cuando se trabaja con grandes cantidades de datos, por lo que siempre hay que tenerla en cuenta.

7.2. Recomendaciones

Para futuros trabajos se recomienda utilizar una segmentación en función de los caracteres individuales de una sílaba fonética. Esto conlleva beneficios y falencias por igual junto con una potencial sobrecarga al momento de realizar las predicciones, puesto que los árboles de decisión serán más extensos. Sin embargo, los resultados pueden ser más precisos lo que significaría transliteraciones válidas.

El trabajo con sílabas fonéticas debe complementarse utilizando mayor contexto. Esto también incluye el análisis del estado actual junto con el estado anterior en un HMM. Trabajar de este modo significaría utilizar trigramas u otros n-gramas para la generación de probabilidades, así como tomar en cuenta los diferentes caminos que se pueden

tomar dentro de la toma de decisiones, ya que recordemos que una sílaba fonética puede tener múltiples secuencias katakana probables.

La escritura en katakana sirve para representar palabras provenientes de otros idiomas, si bien el inglés es el que predomina, esto no quiere decir que sea el único. Durante la búsqueda de diccionarios inglés/katakana se han encontrado registros que muestran que varias palabras provenientes del español, francés, alemán y portugués tienen una transliteración al katakana. Por lo tanto, es posible generar un modelo de transliteración para dichos idiomas.

8. BIBLIOGRAFÍA

[1] R. Saito, «Learning a Language of two Alphabets: Practical Aproaches in Hiragana and Katakana Acquisition for Beginner Learners of Japanese Language,» 2018. [En línea]. Available: https://revistas.uta.edu.ec/erevista/index.php/dide/article/view/646/491.

[2] R. Rubio, «Estudio de la alternancia de uso de las tres grafías del sistema de escritura japonés -Kanji, Hiragana y Katakana,» Universidad Autónoma de Madrid, 2018. [En línea]. Available: https://dialnet.unirioja.es/servlet/tesis?codigo=212182.

[3] N. Al Jahan, «The Origin and Development of Hiragana and Katakana,» ACADEMIA, 2017. [En línea]. Available: https://www.academia.edu/40998205/The_Origin_and_Development_of_Hiragana_and_Katakana.

[4] J. Oh, C. KeySun y H. Isahara, «A Comparison of Different Machine Transliteration Models,» Journal of Artificial Intelligence Research, 2006. [En línea]. Available: https://www.jair.org/index.php/jair/article/view/10468/25090.

[5] Wikipedia, «キト,» Wikipedia Japan, 2 Noviembre 2020. [En línea]. Available: https://ja.wikipedia.org/wiki/%E3%82%AD%E3%83%88.

[6] E. Cotto, «Procesamiento fonológico y la fluidez en la lectura oral,» *Reforma Educativa en el Aula,* 2012.

[7] A. Deri y K. Knight, «Grapheme-to-phoneme models for (almost) any language,» *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,* vol. 1, pp. 399-408, 2016.

[8] B. Ármannsson, «Grapheme-to-phoneme transcription of English words in Icelandic text,» Universidad de Uppsala, 22 Junio 2021. [En línea]. Available: http://uu.diva-portal.org/smash/get/diva2:1571500/FULLTEXT01.pdf.

[9] M. Divay y A. J. Vitale, «Algorithms for Grapheme-Phoneme Translation for English and French,» *Computational Linguistics,* vol. 23, nº 4, 1997.

[10] Festvox, «Festvox,» Carnegie Mellon University's Speech Group, 2022. [En línea]. Available: http://festvox.org/.

[11] The Centre for Speech Technology Research, «The Festival Speech Synthesis System,» Universidad de Edimburgo, [En línea]. Available: https://www.cstr.ed.ac.uk/projects/festival/.

[12] Festvox, «Building letter-to-sound rules automatically,» Festvox, [En línea]. Available: http://festvox.org/bsv/x1471.html.

[13] L. Breiman, J. Friedman, R. Olshen y C. Stone, Classification And Regression Trees, Nueva York: Routledge, 1984.

[14] A. Black, P. Taylor y R. Caley, «The Festival Speech Synthesis System Documentation - CART Trees,» Festvox, 25 Diciembre 2014. [En línea]. Available: http://festvox.org/docs/manual-2.4.0/festival_25.html#CART-trees.

[15] A. Black, P. Taylor y R. Caley, «The Festival Speech Synthesis System Documentation - Features,» Festvox, 25 Diciembre 2014. [En línea]. Available: http://festvox.org/docs/manual-2.4.0/festival_14.html#Features.

[16] Speech Zone, «Module 5 – speech synthesis – phonemes and the front end,» speech.zone, 2022. [En línea]. Available: https://speech.zone/courses/speech-processing/module-5-speech-synthesis/.

[17] Johns Hopkins University, «Internet Archive Wayback Machine,» 15 Febrero 2010. [En línea]. Available: http://web.archive.org/web/20100215160706/http://www.clsp.jhu.edu/ws99/projects/mt/toolkit/.

[18] F. Josef Och y H. Ney, «A Systematic Comparison of Various Statistical Alignment Models",» *Computational Linguistics,* vol. 29, nº 1, pp. 19-51, Marzo 2003.

[19] F. Josef Och, «Franz Josef Och,» [En línea]. Available: http://www.fjoch.com/GIZA++.html.

[20] Estadística Multivariada, «Algoritmo de Esperanza-Maximización (EM),» [En línea]. Available: https://est-mult.netlify.app/algoritmo-de-esperanza-maximizaci%C3%B3n-em.html.

[21] J. Brownlee, «A Gentle Introduction to Expectation-Maximization (EM Algorithm),» 28 Agosto 2020. [En línea]. Available: https://machinelearningmastery.com/expectation-maximization-em-algorithm/.

[22] L. Tian, F. Wong y S. Chao, «Word Alignment Using GIZA++ on Windows,» Universidad de Macau - Departamento de Computación y Ciencias de la Información, 2011. [En línea]. Available: https://aclanthology.org/2011.mtsummit-systems.1.pdf.

[23] MT Research Survey Wiki, «IBM Models,» 07 Agosto 2017. [En línea]. Available: http://www2.statmt.org/survey/Topic/IBMModels.

[24] Statistical Machine Translation, «Word Based Models,» statmt.org, [En línea]. Available: http://www2.statmt.org/book/slides/04-word-based-models.pdf.

[25] Baum, Leonard; Petrie, Ted, «Statistical Inference for Probabilistic Functions of Finite State Markov Chains,» *The Annals of Mathematical Statistics,* vol. 37, nº 6, pp. 1554-1563, Diciembre 1966.

[26] L. Baum y J. Eagon, «An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology,» *Bulletin of the American Mathematical Society,* vol. 73, nº 3, p. 360, 1967.

[27] L. Rincón, «Introducción a los procesos estocásticos,» Facultad de Ciencias UNAM, Enero 2012. [En línea]. Available: https://d1wqtxts1xzle7.cloudfront.net/58401144/procesos2012Rincon.pdf?1550110173=&response-content-disposition=inline%3B+filename%3DProcesos2012Rincon.pdf&Expires=1616611453&Signature=OkgYlzrRMdASdx-sRthcL8MZTQsRhwiFDm7lNkwRDJNBKPBcqTaxybCf0IggvVUxV~5y~OM.

[28] L. Maldonado, «Los Modelos Ocultos de Markov,» Telos, 3 Septiembre 2012. [En línea]. Available: https://www.redalyc.org/pdf/993/99324907003.pdf.

[29] PERL, «PERL,» 2022. [En línea]. Available: https://www.perl.org/.

[30] PERL, «PERL Documentation,» 2022. [En línea]. Available: https://perldoc.perl.org/.

[31] V. Levenshtein, «Binary Codes Capable of Correcting Deletions, Insertions and Reversals,» *Soviet Physics Doklady,* vol. 10, nº 8, pp. 707-710, 1966.

[32] D. Rodriguez, «Analytics Lane,» 17 Julio 2020. [En línea]. Available: https://www.analyticslane.com/2020/06/17/la-distancia-de-levenshtein/.

[33] B. Vardarajan y D. Rao, «e-extension Hidden Markov Models and Weighted Transducers for Machine Transliteration,» Johns Hopkins University, 2009. [En línea]. Available: https://www.aclweb.org/anthology/W09-3527.pdf.

[34] ASP Gems, «Metodología de desarrollo de software (III) – Modelo en Espiral,» ASP Gems, 5 Abril 2019. [En línea]. Available: https://aspgems.com/metodologia-de-desarrollo-de-software-iii-modelo-en-espiral/.

[35] J. Breen, «Electronic Dictionary Research and Development Group,» Diciembre 2022. [En línea]. Available: http://www.edrdg.org/wiki/index.php/JMdict-EDICT_Dictionary_Project.

[36] Unicode, «Unicode 14.0 Character Code Charts - Katakana,» Unicode, 2022. [En línea]. Available: https://www.unicode.org/charts/PDF/U30A0.pdf.

[37] Google Developers, «Machine Learning Crash Course,» Google, [En línea]. Available: https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data.

[38] Institute for Natural Language Processing (IMS), «Building a Unit Selection Synthesis Voice,» 04 Marzo 2018. [En línea]. Available: https://www.ims.uni-stuttgart.de/documents/team/schweitz/docs/unitselection.tutorial.pdf.

[39] Regular Expressions 101, «Regular Expressions 101,» 2022. [En línea]. Available: https://regex101.com/.

[40] H. X. Vinh, «Install MGIZA on Ubuntu,» 29 Abril 2016. [En línea]. Available: https://hovinh.github.io/blog/2016-04-29-install-mgiza-ubuntu/.

[41] J. A. Camarena Ibarrola, «Procesos Estocásticos,» Universidad de Michoacana de San Nicolás de Hidalgo, 15 Marzo 2017. [En línea]. Available: http://dep.fie.umich.mx/~camarena/IntroProcesos%20Estocasticos.pdf.

[42] W. Cavnar y J. Trenkle, «N-Gram-Based Text Categorization,» *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval,* vol. 161175, 1994.

[43] Computer Hope, «Underflow,» 16 Noviembre 2019. [En línea]. Available: https://www.computerhope.com/jargon/u/underflo.htm.

[44] K. Dejonghe, «meta cpan,» 21 Noviembre 2006. [En línea]. Available: https://metacpan.org/pod/Algorithm::Viterbi.

[45] N. Bowers, «meta cpan,» 16 abril 2022. [En línea]. Available: https://metacpan.org/pod/Text::Levenshtein.

[46] B. Bullock, «English to katakana converter,» 2022. [En línea]. Available: https://www.sljfaq.org/cgi/e2k.cgi.

[47] B. Bullock, «Rules for conversion,» 2022. [En línea]. Available: https://www.sljfaq.org/afaq/english-in-japanese.html.

[48] Dive Into Deep Learning, «Probability,» Dive Into Deep Learning, 2020. [En línea]. Available: https://d2l.ai/chapter_preliminaries/probability.html.

9. ANEXOS

Anexo I: Carpeta con los archivos necesarios para la transliteración del inglés a katakana.

https://www.dropbox.com/sh/fyyou1xmf15cfd6/AAAF8NUOJskF3vjDS93V9Dwla?dl=0