PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

Título

DATA AUGMENTATION AND SUBWORD SEGMENTATION FOR SPELL-CHECKING IN AMAZONIAN LANGUAGES

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGÍSTER EN INFORMÁTICA CON MENCIÓN EN CIENCIAS DE LA COMPUTACIÓN

AUTOR

ING. CARLO ANDRÉ ALVA COHELLO

ASESOR

MG. FELIX ARTURO ONCEVAY MARCOS

JULIO, 2021

RESUMEN

En el Perú se han identificado 48 lenguas originarias, según la información extraída de la Base de Datos oficial de Pueblos Indígenas u originarios (BDPI). Estas son de tradición oral [BDPI, 2020]. Por lo que no había una forma oficial de enseñanza. El Instituto Linguistico de Verano (ILV) recopiló y documentó diversas lenguas nativas [Faust, 1973], como un primer intento para tener un documento formal para la enseñanza de una lengua originaria. Fue después que el Gobierno Peruano con su estrategia de inclusión social "Incluir para crecer" creó una guía oficial para la enseñanza de las lenguas originarias en su intento de normalizar el uso de estas lenguas [Jara Males, Gonzales Acer, 2015].

Como se menciona en [Forcada, 2016], el uso de tecnologías del lenguaje permite obtener una normalidad, incremento de literatura, estandarización y mayor visibilidad. En el caso de Perú, ha habido iniciativas, como analizadores morfológicos [Pereira-Noriega, et al., 2017] o correctores ortográficos [Alva, Oncevay, 2017], enfocados en las lenguas originarias de escasos recursos computacionales que pretenden apoyar el esfuerzo de revitalización, la educación indígena y la documentación de las lenguas [Zariquiey et al., 2019].

Enfocándose en lenguas amazónicas se realizó un proyecto utilizando redes neuronales para desarrollar un corrector ortográfico enfocado en las lenguas originarias con buenos resultados a nivel de precisión [Lara, 2020]. En ese trabajo, al disponer de poca cantidad de datos se generaron datos sintéticos con un método aleatorio los cuales al ser evaluados con las métricas CharacTER [Wang, et al., 2016] y BLEU [Papineni, et al., 2002] obtuvieron resultados bastante bajos. Además, las lenguas amazónicas al ser ricas a nivel morfológico y tener un vocabulario extenso es difícil representar palabras fuera del vocabulario, por lo que es recomendable usar sub-palabras como término medio [Wu, Zhao, 2018].

El presente proyecto desarrolla distintos métodos de generación de datos, diferentes al aleatorio, que son más robustos al considerar errores que son más cercanos a la realidad. A su vez, para reducir el costo computacional y mantener la capacidad de generar un vocabulario abierto, adicionalmente se entrena redes neuronales que reciban como entrada sub-palabras tales como sílabas y segmentos divididos por *byte pair encoding* (BPE). Finalmente, de los experimentos concluimos que hubo mejoras con los métodos y la segmentación propuesta y se tienen más recursos computacionales para nuestras lenguas amazónicas.

TABLA DE CONTENIDOS

LISTA DE TABLAS

LISTA DE FIGURAS

1 Motivación

En el Perú se han identificado 48 lenguas originarias, según la información extraída de la Base de Datos oficial de Pueblos Indígenas u originarios (BDPI). Estas son de tradición oral, lo que quiere decir que se enseñan a través de cantos, cuentos, entre otros [BDPI, 2020]. Al ser así, no se necesitaba una guía o documento formal para su enseñanza. El Instituto Linguistico de Verano (ILV) recopiló y documentó diversas lenguas nativas [Faust, 1973], siendo este un primer intento para tener un documento formal para la enseñanza de una lengua originaria. Éste contenía lecciones e información básica para poder hablarla con ejemplos en castellano y la lengua en cuestión. Sin embargo, no fue muy difundido ni utilizado. Fue después que el Gobierno Peruano con su estrategia de inclusión social "Incluir para crecer" creó una guía oficial para la enseñanza de las lenguas originarias en su intento de normalizar el uso de estas lenguas [Jara Males, Gonzales Acer, 2015].

Aún con estas iniciativas del gobierno, se tiene que seguir realizando acciones que permitan normalizar y darles mayor visibilidad a estas lenguas originarias, ya que estas al contar con pocos o ningún tipo de datos anotados o recursos computacionales son consideradas como lenguas minoritarias o de escasos recursos [Hartmann, et al., 2014]. Como se menciona en [Forcada, 2016], el uso de tecnologías del lenguaje permite obtener una normalidad, incremento de literatura, estandarización y mayor visibilidad. En el caso de Perú, ha habido iniciativas, como analizadores morfológicos [Pereira-Noriega, et al., 2017] o correctores ortográficos [Alva, et al., 2017], enfocados en las lenguas originarias de escasos recursos computacionales que pretenden apoyar el esfuerzo de revitalización, la educación indígena y la documentación de las lenguas [Zariquiey et al., 2019].

Con el desarrollo de redes neuronales artificiales, éstas se han utilizado en distintos proyectos para lenguas de escasos recursos ([Ragni, et al., 2014], [Karakanta, et al., 2018] y [Almansor, Al-Ani, 2018]). Aún así hay ciertos problemas a resolver. Uno de ellos es que para el entrenamiento de las redes se necesitan grandes cantidades de datos [Náplava, Straka, 2019]. Enfocándose en lenguas amazónicas se realizó un proyecto utilizando esta tecnología para desarrollar un corrector ortográfico enfocado en las lenguas originarias con buenos resultados solo a nivel de precisión [Lara, 2020]. En ese trabajo, para resolver el problema de la poca cantidad de datos, a través de un método aleatorio se crearon datos sintéticos que fueron utilizados para entrenar redes neuronales que eran modelos de secuencia a secuencia y utilizaban cadenas de caracteres como entrada. Los resultados fueron evaluados de forma automática con las métricas CharacTER [Wang, et al., 2016] y BLEU [Papineni, et al., 2002]. A diferencia de la precisión, estos modelos obtuvieron resultados bastante bajos al evaluarlos con métricas como CharacTER y BLEU.

Considerando las lenguas amazónicas como un lenguaje rico a nivel morfológico y con un extenso vocabulario, un problema bastante conocido es poder representar palabras fuera del vocabulario [Wu, Zhao, 2018] cuando se tienen modelos con vocabularios de tamaño fijo o cerrados. Para sobrellevarlo se suele realizar un preprocesamiento de las palabras para que se trabaje a nivel de caracteres [Ataman, et. al., 2019]. Esta estrategia conlleva a mayores tiempos de procesamiento y entrenamiento para los modelos, además se considera que los caracteres no son una buena unidad mínima para representar una palabra por lo que se utiliza un termino medio como las sub-palabras [Wu, Zhao, 2018].

Para afrontar el problema de la escasez de conjunto de datos en las lenguas minoritarias, el presente proyecto pretende desarrollar distintos métodos de generación de datos, diferentes al aleatorio, que sean más robustos al considerar errores que son más cercanos a la realidad. Para validar la mejora en los métodos de aumento de datos se comparará los resultados obtenidos de la red neuronal para la corrección ortográfica entrenada con los datos de la línea base de [Lara, 2020] y los nuevos datos generados. A su vez, para reducir el costo computacional y mantener la capacidad de generar un vocabulario abierto, se plantea entrenar nuevas redes neuronales que reciban como entrada sub-palabras tales como sílabas y segmentos divididos por *byte pair encoding* (BPE), lo cual diferencia del trabajo antes mencionado que usa únicamente cadena de caracteres. Finalmente, para validar estos nuevos datos con sub-palabras se entrenarán las nuevas redes neuronales y serán evaluadas con las mismas métricas de la línea base de [Lara, 2020].

2 Objetivos

2.1 Objetivo general

Implementar métodos para crear datos sintéticos y redes neuronales que utilicen sub-palabras para la corrección ortográfica en 4 lenguas amazónicas: Shipibo-Konibo, Ashaninka, Yanesha, Yine.

2.2 Objetivos específicos

- Objetivo 1: Implementar métodos para crear datos sintéticos enfocados en la distribución de teclas, errores fonéticos más comunes y la similitud de sílabas
- Objetivo 2: Entrenar redes neuronales secuencia a secuencia que utilicen conjuntos de datos de sub-palabras
- Objetivo 3: Comparar el rendimiento de los modelos de corrector ortográfico entrenados con los datos originales contra los mismos modelos entrenados con datos sintéticos y datos segmentados

2.3 Resultados

- Objetivo 1: Implementar métodos para aumento de datos enfocados en la distribución de teclas, errores fonéticos más comunes y la similitud de sílabas
    - Métodos de aumento de datos
    - Data sintética
- Objetivo 2: Implementar redes neuronales secuencia a secuencia que utilicen conjuntos de datos de sub-palabras
    - Data segmentada
    - Modelos entrenados con sub-palabras
- Objetivo 3: Comparar el rendimiento de los modelos de corrector ortográfico entrenados con los datos originales contra los mismos modelos entrenados con datos sintéticos y datos segmentados
    - Informe de evaluación de resultados

3 Estado del arte

Para la revisión del estado del arte se decidió plantear preguntas de investigación que nos ayuden a enfocar lo que se quiere realizar, plantear cadenas de búsquedas que nos ayuden a resolver las preguntas definidas y finalmente identificamos librerías digitales donde aplicar las cadenas de búsqueda para revisar los resultados de las búsquedas.

Para empezar se plantearon las preguntas de investigación:
¿Qué métodos existen para el aumento de datos en lenguas de escasos recursos para la tarea de corrección ortográfica?
¿Qué métodos o algoritmos existen para la segmentación de palabras y se han aplicado en la tarea de corrección ortográfica?

En base a las preguntas planteadas se determina ciertas palabras clave que ayudan a resolver las preguntas de investigación. La lista de términos es:

- Data augmentation and spell-checking
- Text augmentation and spell-checking
- Independent language and spell-checking
- Low-resource languages and spell-checking
- Minority languages and spell-checking
- Subword segmentation and spell-checking
- Spell-checking

Una vez definida la lista de términos se generaron las siguientes cadenas de búsqueda:

Data augmentation AND “low-resource languages”
Text augmentation AND “independent language”
Sub word segmentation AND “low-resource languages”
Text augmentation AND “Minority languages”
Sub word segmentation AND “independent language”

Ya definidas se realizó la búsqueda en librerías digitales como ACM[1], Scopus[2], IEEE Xplore[3], Web of Science[4], ScienceDirect[5] y ACL Anthology [6] para obtener una lista de referencias, las cuales fueron revisadas para poder acotar la lista a través de la revisión del título y los resumenes que tenían relación con lo que se buscaba. Cabe mencionar que ACL Anthology fue la que mejores referencias tenía. Finalmente de la lista filtrada se revisaron los documentos completos en búsqueda de información relavante para el proyecto.

3.1 Trabajos relacionados

Durante la revisión para responder a la primera pregunta de investigación se encontró un trabajo [Li et al., 2018] que utilizaba redes neuronales recurrentes anidadas para la detección de errores ortográficos para el inglés. Una particularidad del trabajo era

[1] https://www.acm.org/
[2] https://www.scopus.com/
[3] http://ieeexplore.ieee.org/
[4] https://apps.webofknowledge.com/
[5] http://www.sciencedirect.com/
[6] https://www.aclweb.org/anthology/

que las redes fueron entrenadas con data generada considerando similitud fonética. Las redes recurrentes que implementaron eran una CharRNN y una WordRNN como se ve en la figura 1. La primera se encargaba de recolectar información ortográfica al tomar cada palabra como una secuencia de caracteres, y la segunda predecía la palabra correcta mediante el uso del contexto y la información ortográfica de la primera red.

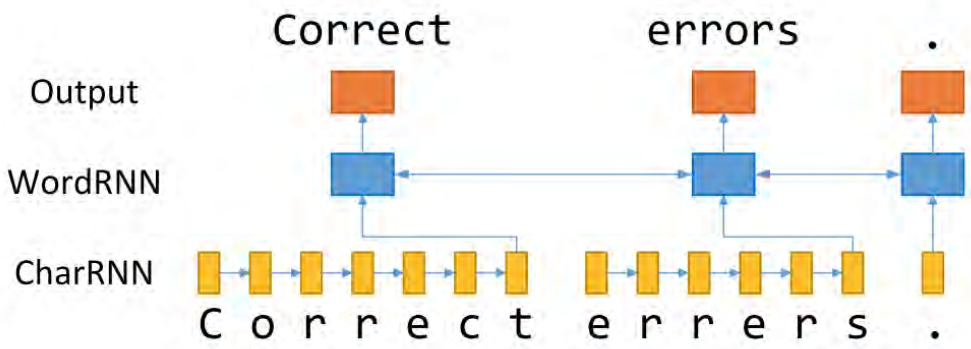

Figura 1. Arquitectura de las RNN anidada [Li., et al., 2018]

Un problema común para el entrenamiento de las redes neuronales es la cuantiosa cantidad de datos que requiere. Un enfoque para lidiar con ello es la generación de datos sintéticos, elegido en el trabajo [Li, et al., 2018]. Ellos propusieron que los errores suelen darse a raíz de que uno escribe lo que escucha, de manera que suelen confundir las palabras que son similares en su pronunciación. Su enfoque se encargaba de obtener los símbolos fonéticos y entrenar un modelo que permitirá recibir una palabra y devuelva los fonemas. Con esto pudieron crear datos cambiando los caracteres de fonemas similares en base a un umbral que eligieron. Con ese nuevo conjunto de datos pudieron entrenar su modelo de corrección ortográfica y lograron superar la línea base de trabajos similares. Otro trabajo [Shah, de Melo, 2020] que presentó una respuesta a la pregunta de investigación, considera el problema de la corrección como un problema de traducción. Ya que pasar de una oración incorrecta a una correcta es como si fuese una "traducción" [Chollampatt, Ng 2017]. Con este enfoque pueden usarse métodos orientados a la traducción para resolver el problema de corrección ortográfica. Para obtener datos de entrenamiento para su modelo de corrección ortográfica generan los datos de error en base a un análisis estadístico de un pequeño corpus de oraciones mal escritas, donde obtienen los errores tipográficos que más se repiten. Muchos de estos errores son errores tipográficos por teclas cercanas y son conocidos como errores de dedos gruesos. Con las probabilidades calculadas de cada tipo de error, luego generan textos corruptos utilizando estos errores de manera aleatoria con un coeficiente que multiplicado por la probabilidad del tipo de error correspondiente, agrega o no agrega el error. De esta manera generan una distribución similar en este nuevo texto con error. Una vez completado, utilizan un algoritmo modificado de corrección ortográfica basado en diccionarios que obtiene la peor sugerencia de corrección, logrando así tener errores de contexto, ya que estas palabras están bien escritas, pero no corresponden al contexto de la oración. Finalmente, proceden a evaluar la efectividad al detectar y corregir los errores de la red neuronal entrenada con sus datos. Los errores ortográficos que consideraron en el trabajo fueron 5: substitución, inserción, repetición, eliminación y transposición. Dentro de las métricas utilizadas en este trabajo, consideraron BLEU. Esta métrica calcula la media geométrica de la precisión de los n-gramas multiplicada por una

penalización de brevedad. La precisión de los n-gramas se calcula dividiendo el número de n-gramas de la traducción del sistema que aparecen en alguna de las traducciones de referencia entre el número de palabras de la traducción del sistema [Mayor, et al., 2009]. Este fue elegido porque es comunmente utilizado donde se ve a la corrección ortográfica como un problema de traducción.

Para responder la segunda pregunta de investigación, se encontró un trabajo [Zhou, et al., 2017] en el que enfocan la corrección ortográfica como un problema de traducción; de esta manera utilizan unas redes recurrentes multicapas como codificador (*encoder*) y decodificador (*decoder*). Un problema que mencionan acerca de estas redes recurrentes es la dificultad del aprendizaje de dependencias a largo plazo (*long term dependencies*). Adicionalmente mencionan que no es clara la estrategia a seguir cuando la entrada y la salida de la red neuronal tienen longitudes distintas. Para afrontar esos problemas, plantearon el uso de un esquema basado en caracteres como entrada para el entrenamiento. Aquí es donde utilizan un método de segmentación conocido como *byte pair encoding* (BPE) [Gage, 1994], el cuál es un algoritmo de compresión que se encarga de reemplazar con un byte los patrones repetidos que encuentra. De esta manera genera un equilibrio entre la lentitud de entrenar solo con caracteres y el problema de la gran cantidad de vocabulario de salida. En este otro trabajo [Sennrich, et al., 2015] hablan de distintas segmentaciones de palabras enfocados en palabras raras para un traductor automático neural, considerando una palabra rara como una entidad, palabras compuestas o prestadas de otro idioma y palabras complejas morfológicamente. Su hipótesis es que la segmentación apropiada es suficiente para que la red pueda aprender y generalizar una traducción de una palabra rara que no haya sido vista antes. Ya que, si los datos de entrenamiento fuesen al nivel palabras completas, la red no podría traducir o generar palabras que no fueron vistas antes. La segmentación elegida fue una variación del BPE, donde se usan caracteres en lugar de bytes a la hora de segmentar las palabras. Esta segmentación ofreció mejores resultados contra las distintas opciones de segmentación que presentaron. Otro tipo de segmentación que puede ser útil para partir las palabras son las sílabas, este método es utilizado en un corrector ortográfico para shipibo-konibo [Alva, Oncevay, 2017]. En ese trabajo utilizaron grafos de sílabas donde los nodos eran las sílabas y las aristas eran la representación de como se unían para formar las palabras. Para ello, utilizaron como guía las reglas para formar las palabras y en base a eso generaron todas las sílabas posibles. Con la sílabas listas, procesaron el conjunto de datos que tenían para obtener las relaciones entre las sílabas y completar las relaciones en su grafo. Finalmente desarrollaron un algoritmo que recorría el grafo para encontrar posibles soluciones por cada palabra que se deseaba corregir en la oración.

3.2 Observaciones y conclusiones

Al haber revisado diferentes trabajos que permitieron responder a las preguntas que se plantearon, se descubre distintos enfoques para la generación de datos sintéticos y la segmentación de datos. Para la segmentación de datos se ha planteado usar BPE, así como se ha visto en los trabajos previos. Sin embargo, este no será el único ya que también se ha considerado el uso de silabificadores así como se vio en la revisión en [Alva, Oncevay, 2017], para segmentar las palabras en sílabas y tener un punto intermedio entre el uso de palabras y caracteres para el entrenamiento de la red neuronal. Para la generación de datos sintéticos se ha considerado los tipos de errores mencionados durante la revisión (substitución, inserción, repetición, eliminación y

transposición) los cuales serán generados con distintos enfoques. Uno de los enfoques que dio inspiración fue encontrado en la revisión y es el de similitud fonética. Como no se cuenta con una función de grafema a fonema para cada una de las lenguas se planteó aprovechar los silabificadores para la generación de errores en base a sílabas similares. Otro enfoque a considerar es el uso del conocimiento de lingüistas que nos apoyaron realizando unos documentos donde encontramos cuales son los errores ortográficos más comúnes. Así se pueden tener errores más cercanos a la realidad tal como se ve en [Etoori, et. al., 2018]. Otro enfoque es por la cercanía de los caracteres en el teclado, por ejemplo, si se ha presionado la tecla "s" se consideran las teclas a su alrededor "q, w, e, a, d, z, x" para generar el error. Esto se considera en uno de los trabajos revisados como error de dedos gruesos, al presionar por equivocación una tecla cercana.

Con este proyecto se dispondrá de distintos recursos computacionales para las lenguas amazónicas de escazos recursos como nuevos conjuntos de datos y también de métodos genéricos para aumento de datos que pueden ser aplicados a otras lenguas amazónicas. Lo que permitirá que se pueda generar mayor cantidad de datos de entrenamiento y que sean más cercanos a la realidad para futuros proyectos. Adicionalmente, se dispondrá de nuevos modelos de corrección ortográfica con datos segmentados y la implementación de estos métodos de segmentación que pueden ser replicados para las otras lenguas.

4 Marco teórico

Se presentan los conceptos necesarios para lograr un entendimiento del problema y su solución.

Redes neuronales:

Es una técnica de aprendizaje de máquina que simula el mecanismo de aprendizaje de las neuronas, donde la neurona es representada como una unidad computacional [Aggarwal, 2018].

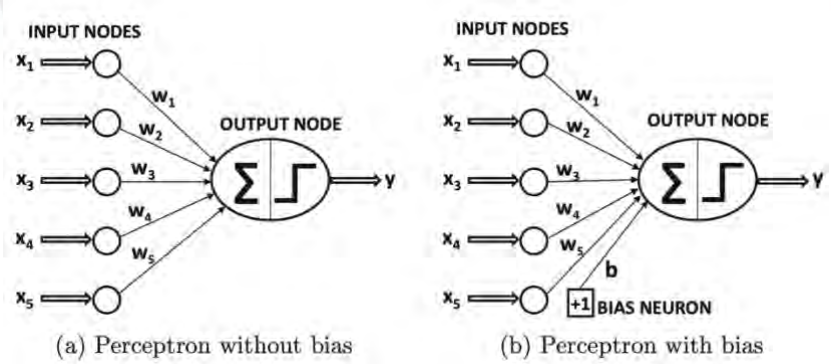

Figura 2: Red neuronal simple [Aggarwal, 2018]

Redes neuronales Recuerrentes:

Es un tipo de red neuronal que permiten retener la información previa al realizar el proceso de entrenamiento [Mikolov, et al., 2011]. Este tipo de redes se ve mejor representado mediante se avanza con el tiempo.

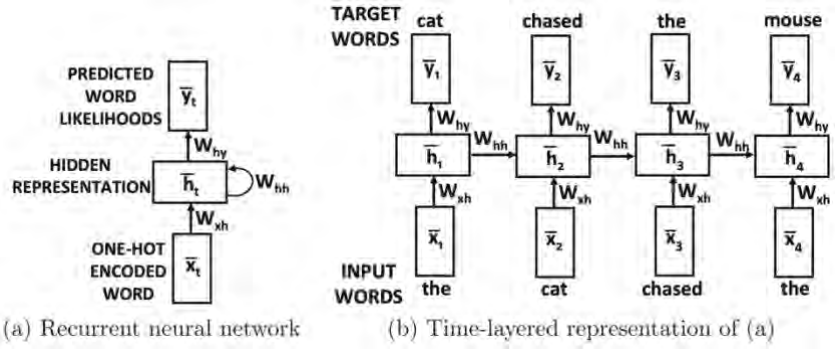

Figura 3: Representación de una red recurrente en el tiempo [Aggarwal, 2018]

Modelo Secuencia a Secuencia (Seq2seq):
Es un modelo que consiste de dos capas (Codificador y Decodificador) en el cual se base en vectores de secuencias. Que recibe una secuencia que es codificada en la primera capa y se decodifica en la siguiente con la información del contexto de la secuencia inicial. [Wu, et al., 2019]

Data augmentation:
Es una técnica que permite explotar el conocimiento de un dominio para incrementar el número de ejemplos en datos de entrenamiento.
[Hernández-García, König, 2018]

Byte Pair Encoding (BPE):
Es un algoritmo de compresión de datos que reemplaza pares de bytes repetidos por un byte único que no se encuentre en los datos iniciales [Gage, 1994].

Lenguas originarias del Perú:
Son aquellas anteriores a la difusión del castellano o español y que se preservan y emplean en el ámbito del territorio nacional [El Peruano, 2017].

Lenguas minoritarias:
Son lenguas habladas por grupos minoritarios que tienen un carácter histórico o social [Allardt, 1984].

Lenguas de escasos recursos:
Son lenguas que poseen recursos computacionales limitados y con escasos recursos de entrenamiento [Hartmann, et al., 2014].

Métricas
BLEU:
Es una métrica usada para evaluar la calidad de una traducción hecha por un sistema de traducción automática. Esta métrica varía entre 0 y 100, donde un mayor puntaje quiere decir que es mejor [Papineni, et al., 2002].

CharacTER:
Es una métrica que se refiere a la mínima cantidad de ediciones de caracteres requeridas para ajustar la oración inicial a la oración hipotética en una traducción donde un menor valor representa que es mejor al requerir menor esfuerzo de post-edición[Wang, et al., 2016]. La ecuación siguiente es la manera de representar CharacTER:

$$CharacTER = \frac{distancia\ edición + costo\ intercambio}{\#\ caracteres\ en\ la\ oración\ de\ hipótesis}$$

Tipos de errores ortográficos[Shah, de Melo, 2020]
Error de substitución:
Se refiere cuando un caracter es reemplazado por un carácter incorrecto.
Ejemplo:
Correcta: aínbo yákatara nokon wáata iki
Incorrecta: aínbo yákatara nojon wáata iki

Error de inserción:
Se refiere cuando se inserta un caracter adicional que no corresponde.
Ejemplo:
Correcta: aínbo yákatara nokon wáata iki
Incorrecta: aínbo yákatara nokoni wáata iki

Error de repetición:
Se refiere cuando un caracter es duplicado de manera incorrecta.
Ejemplo:
Correcta: aínbo yákatara nokon wáata iki
Incorrecta: aínbo yákatara nokoon wáata iki

Error de eliminación:
Se refiere cuando un caracter es omitido o eliminado de una palabra.
Ejemplo:
Correcta: aínbo yákatara nokon wáata iki
Incorrecta: aínbo yákatara nokn wáata iki

Error de transposición:
Se refiere cuando dos caracteres consecutivos tienen el orden inverso.
Ejemplo:
Correcta: aínbo yákatara nokon wáata iki
Incorrecta: aínbo áykatara nokon wáata iki

5 Metodología

5.1 Datos

Como primera fase, se realiza una evaluación y procesamiento de los datos de las lenguas elegidas: Shipibo-Konibo, Ashaninka, Yanesha, Yine. Este conjunto de datos es de la misma fuente [Bustamante, et al., 2020] que la línea base, el cual fue extraído del Sistema Digital para el Aprendizaje PerúEduca[7] que tiene información de cada lengua utilizada para este trabajo. Este consta de guías de alfabetos, relatos tradicionales, manuales de escritura y herramientas de trabajo.

[7] http://www.perueduca.pe/

Lengua	Cantidad de oraciones	Cantidad de palabras	Cantidad de palabras únicas	Total de palabras / palabras únicas
Shipibo-Konibo	22,032	192,058	23,721	5.87
Ashaninka	12,629	101,468	22,944	4.42
Yanesha	13,241	101,018	23,626	4.27
Yine	7,658	57,750	14,142	4.08

Tabla 1: Análisis de conjunto de datos

En la Tabla 1, se observa que se dispone de una mayor cantidad de oraciones en Shipibo-Konibo; sin embargo, la cantidad de palabras únicas es bastante similar a Yanesha y Ashaninka que presenta casi la mitad de oraciones. También vemos que tenemos un ratio de total de palabras entre palabras únicas entre 4 y 6 puntos.

5.2 Métodos de aumento de datos

5.2.1 Aumento de datos basado en teclas cercanas

Una vez conocidos nuestros datos, se define el primer enfoque planteado, que es basado en los errores que pueden darse al usar el teclado, usando la posible coincidencia en errores entre teclas cercanas. Por este motivo se utiliza una estructura que nos permite mapear las teclas cirncundantes a la presionada para ser utilizado en la generación de errores.

La función de errores que se desarrolló recorre una oración por palabra, durante este recorrido se genera aleatoriamente valor de “Verdadero” o “Falso” para evaluar si a esa palabra se le debe generar error. Esto se hace para que en una oración no haya palabras erróneas en su totalidad sino que existan correctas y erróneas. Si a la palabra le corresponde generarle error, se genera un número aleatorio entre 0 y 2. De acuerdo al valor se realiza una inserción, una eliminación o un intercambio de letra a la palabra a una de sus letras elegidas al azar. Esta letra errónea es escogida aleatoriamente de la lista de letras circundantes en el teclado a la letra que se tiene que reemplazar.

Para entender mejor se muestra los pasos de generar el error en una palabra:

1. La palabra correcta es “bari”.
2. Se tiene que generar el error de intercambio a la letra “b” en “bari”.
3. La letra “b” en el teclado tiene a su alrededor a: v, g, h, n.
4. Aleatoriamente se escoge la “h” para ser el reemplazo.
5. Finalmente la palabra queda: hari.

Esta función es independiente de la lengua y por cada oración correcta se crean 5 oraciones erróneas en cada una de las lenguas propuestas. Obteniendo la cantidad de datos que se muestran en la tabla siguiente.

Lengua	Cantidad de oraciones	Cantidad de oraciones incorrectas	Promedio de errores por oración
Shipibo-Konibo	22,032	110,160	4
Ashaninka	12,629	63,145	4
Yanesha	13,241	66,205	3
Yine	7,658	38,209	3

Tabla 2: Cantidad de oraciones generadas por cercanía de teclas

5.2.2 Aumento de datos basados en errores comunes por fonología

Para este enfoque se aprovechó la posibilidad de confundir un fonema con otro al ser similares. Esto quiere decir que al ser pronunciado de manera similar puede generar duda y escribir erróneamente una palabra. Un ejemplo de lo mencionado en Yanesha para el fonema /o/:

- Palabra errónea: p**u**echech
- Palabra correcta: p**o**echech

De esta manera se podrá entrenar un corrector con datos más robustos y reales ya que suelen ser errores frecuentes en los hablantes. Para mapear estos errores y poder desarrollar una función que los pueda generar, se obtuvo la ayuda de lingüistas que nos proporcionaron documentos que fueron trabajados en colaboración con docentes y hablantes de la lengua utilizando de fuente documentos previos que recopilaban información como: fonemas, formas de pronunciación, grafemas e información lingüística relevante. En este documento lograron recopilar estos errores fonéticos comunes antes mencionados en tablas que usamos. En el apéndice A se encuentran estos documentos para que puedan ser revisados.

A partir de estos documentos se implementó un método independiente de la lengua que permite generar estos errores. Este consiste en analizar la oración y por cada error común que encuentra genera un error en la oración. De esta manera se logró tener una buena cantidad de oraciones como se ve en la Tabla 3.

Un ejemplo para entender el método realizado a través de los pasos que se siguen:

1. Se trasladan los errores mapeados en una estructura de datos como un diccionario. Donde la llave es la letra correcta y se tiene como valor una lista con los errores comunes ya que pueden ser más de 1.

```
common = {
        'ë': ['e'],
         'o': ['u'],
        'k': ['qu', 'c'],
    }
```

2. Se evalúa una oración por cada elemento del diccionario como: achankëllñapa' atthuch murrechená.
3. Si se encuentra la letra (ë), se realiza el cambio: “achank**ë**llñapa' atthuch murrechená” por “achank**e**llñapa' atthuch murrechená”.
4. Si se encuentra la letra (k) se realiza el cambio: “achan**k**ëllñapa' atthuch murrechená” por “achan**c**ëllñapa' atthuch murrechená”.
5. Finalmente quedan las oraciones incorrectas.

Lengua	Cantidad de oraciones	Cantidad de oraciones incorrectas	Promedio de errores por oración
Ashaninka	12,629	90,929	6
Yanesha	13,241	93,604	1

Tabla 3: Cantidad de oraciones generadas por errores comunes por fonología

Actualmente solo se cuenta con la información necesaria para 2 lenguas (Ashaninka y Yanesha). Sin embargo, la función desarrollada permite generar los errores fácilmente cuando se obtenga la información respectiva de Yine y Shipibo-Konibo.

5.2.3 Aumento de datos basados en similitud de sílabas

Para este enfoque se utilizó los 3 silabificadores que se tenían a disposición. El de Shipibo-konibo [Alva, Oncevay, 2017] con los de Yine y Yanesha[8] que estuvieron basados en el primero. Los silabificadores devuelven un arreglo de sílabas por cada palabra analizada. Se debe considerar que no todas las palabras pueden ser silabificadas porque pueden tener préstamos del castellano que no siguen las mismas reglas de silabificación. Aún con estas limitaciones se pudieron obtener 2028 sílabas únicas para Yanesha, 817 para Yine y 354 para Shipibo-konibo. Una vez que se tienen las sílabas únicas se procede a generar el error de las sílabas evaluando cada palabra en la oración para ver si es posible obtener sus silabas, en caso se pueda, se reemplaza una de sus sílabas de manera aleatoria por una similar. Para encontrar la sílaba similar para el reemplazo se escogió utilizar la distancia de edición.

Un ejemplo más claro para entender el método:

1. Silabíficar una oración:
   Oración: kachorin yamax chorishrin
   Sílabas:
   ka – chio – rin
   ya – ma
   cho – rish – rin
2. Por cada palabra se escoge una sílaba aleatoriamente.
   ka → wa
   ya → ia
   ma → max
3. Se reemplaza y se forma la nueva oración
   oración incorrecta: wachorin iamax chorishrin

Tras aplicar este método se obtuvieron la cantidad de oraciones que se ve en la Tabla 4.

[8] https://github.com/iapucp/eib-spell-checking/tree/master/neurSP/utils

Lengua	Cantidad de oraciones	Cantidad de oraciones incorrectas	Promedio de errores por oración
Yine	7,658	15,316	7
Yanesha	13,241	26,482	6
Shipibo-konibo	22,032	44,065	8

Tabla 4: Cantidad de oraciones generadas por similitud de sílabas

5.3 Métodos de segmentación

5.3.1 Segmentación por sílabas

Para este enfoque era necesario el uso de los silabificadores, para poder tener las oraciones como sílabas para la entrada del modelo de corrección ortográfica. Sin embargo, salió la cuestión de qué hacer cuando una palabra no puede ser silabificada. Entonces se decidió separar la palabra en caracteres y agregarlos directamente a la cadena de entrada que se iba a generar. Se decidió que sean caracteres en lugar de la palabra completa porque de esta manera se tiene un abanico mayor para que el modelo pueda presentar la corrección. Otro punto para considerar era como distinguir el inicio y fin de una palabra en la cadena de entrada resultante luego de procesar toda la oración, en este caso se decidió dejar el espacio en blanco como un carácter dentro de la cadena de entrada y utilizar un carácter adicional "<sp>" para separarlos. En los siguientes ejemplos se aprecia mejor la explicación y también podemos ver la proporción de palabras que fueron silabificadas en una oración en la siguiente tabla.

Lengua	Conjunto de datos	Promedio de palabras en oración	Promedio de palabras silabificadas
Yanesha	Errores fonéticos	9	8
	Teclas cercanas	7	7

Tabla 5: Proporción de palabras silabificadas en una oración

Caso: oración completamente silabificable
Como se ve en la oración ya procesada se considera los espacios en blanco ya que esto permite reconstruir la oración.

Oración: errochenat yakren yenaremh
Oración procesada: e<sp>rro<sp>che<sp>nat<sp> <sp>ya<sp>kren<sp>
<sp>ye<sp>na<sp>remh

Caso: Oración no silabificable
Como se ve en la oración ya procesada se agrega caracter por caracter a la entrada cuando una palabra no se puede silabificar.

Oración: pekellkayes ñeth puktetsa
Oración procesada: pe<sp>kell<sp>ka<sp>yes<sp> <sp>ñeth<sp>
<sp>p<sp>u<sp>k<sp>t<sp>e<sp>t<sp>s<sp>a

5.3.2 Segmentación por byte pair encoding

Se decidió utilizar *byte pair encoding* (BPE) ya que este ayuda a representar un vocabulario abierto a través de un vocabulario de tamaño fijo. Se utilizó una implementación de BPE basada en [Sennrich, et al., 2015]. Donde en lugar de reemplazar con un byte se reemplaza con un caracter el patrón de caracteres con más repeticiones y así sucesivamente por una cantidad de veces determinada.

Se realizó un script que permita generar el vocabulario en base a la cantidad de operaciones de mezcla. Una vez generado ese vocabulario, este sirve para procesar la oración y representarla en base a este vocabulario. Se realizaron distintas conjuntos de datos en base a las cantidades de operaciones de 2500, 5000, 7500 y 10000.

Un ejemplo de como queda el procesamiento traes aplicar BPE con las distintas cantidades de operaciones se ve en los ejemplos de la Tabla 6. En la descripción del repositorio[9] del paquete utilizado que implementó el BPE utilizado para el proyecto, si se desea recontruir la oración original se debía realizar un reemplazo de caracteres.

Oración original	Oración procesada con 2500 operaciones	Oración procesada con 5000 operaciones	Oración procesada con 7500 operaciones	Oración procesada con 10000 operaciones
esu't penten atma'ntatarethó errathe' all	esu't penten atma'ntat@@ areth@@ ó errathe' all	esu't penten atma'ntat@@ arethó errathe' all	esu't penten atma'ntat@@ arethó errathe' all	esu't penten atma'ntat@@ arethó errathe' all

Tabla 6: Ejemplos de una oración con BPE con distintas cantidades de operaciones

6 Experimentación y resultados

Una vez generado los datos de los métodos de aumento de datos se pretende realizar distintas pruebas con el modelo que mejor puntuación obtuvo de la línea base [Lara, 2020] y poder identificar si la data sintética creada mejora los resultados de este. Para validarlo se usarán 2 métricas ya antes definidas que son: CharacTER y BLEU.

Según los valores obtenidos en la experimentación de cada lengua de la línea base [Lara, 2020] donde probaron distintas redes neuronales como: un modelo secuencia a secuencia simple, un modelo secuencia a secuencia con regularización y otro modelo secuencia a secuencia con mecanismo de atención. El mejor modelo fue de secuencia a secuencia con un mecanismo de atención [Luong, Pham, Manning, 2015] con 46 caracteres de longitud en entrada y salida. Por lo que inicialmente se evaluará un modelo con los mismos parámetros, pero entrenado con los distintos datos sintéticos generados. Un punto para tener en cuenta es que los datos de entrada y salida están delimitados a máximo 46 caracteres por lo que para usar los datos se han filtrado oraciones que tengan más de esa cantidad.

Para iniciar con los experimentos se definió que se utilizaría el modelo secuencia a secuencia con mecanismo de atención, el cual obtuvo los mejores resultados de los

[9] https://github.com/rsennrich/subword-nmt

experimentos en la línea base [Lara, 2020]. Se seguiría utilizando el mismo paquete de Python, llamado TensorFlow, que nos permite crear, entrenar y manipular redes neuronales. La configuración que se utilizó es la misma que la línea base, donde se separó el conjunto de datos en 80% para el entrenamiento, 10% para la validación y 10% para la prueba. Sin embargo, para efectos de comparación también se utilizó el mismo conjunto de pruebas que la línea base. Además se utilizaron 100 épocas para el entrenamiento, Adam como optimizador, *categorical cross-entropy* para la pérdida, entre otros parámetros. Los detalles se encuentran en el Apéndice.

6.1 Aumento de datos

Para el primer experimento se realizó el entrenamiento del modelo con los datos generados por la cercanía de teclas. Luego de realizar el entrenamiento se obtuvieron las métricas de BLEU y CharacTER que fueron también utilizadas en la línea base.
Para obtener el valor de BLEU se utilizó una implementación en Python basada en un proyecto[10][Post, 2018] que facilitó utilizar como entrada 2 archivos de texto que corresponden a la oración incorrecta y la correcta que fueron generados a partir del conjunto de prueba. A su vez, estos archivos son utilizados para obtener el valor de CharacTER usando una implementación basada en el trabajo original [Wang, et al., 2016]. Los resultados obtenidos se pueden ver en la siguiente tabla, donde se puede ver la comparación del modelo de la línea base entrenado con errores generados aleatoriamente contra el mismo modelo entrenado con errores generados por cercanía de teclas, teniendo en cuenta que es mejor obtener un valor alto en BLEU y un valor bajo en CharacTER.

Lengua	BLEU (Línea base)	BLEU	CharacTER (Línea base)	CharacTER
Shipibo-Konibo	18.5	42.3	0.1937	0.1123
Ashaninka	12.4	14.8	0.1803	0.1286
Yanesha	6.9	23.3	0.2157	0.1430
Yine	15.2	17.3	0.2025	0.1708

Tabla 7: Resultado de métricas por cercanía de teclas

Como podemos ver se obtuvieron mejores resultados según las métricas. Esto puede deberse a que se tienen más casos de corrección ya que al considerar las teclas cercanas tenemos una gama más amplia de opciones. Además, cabe notar que se tuvo una mayor cantidad de datos con respecto a la cantidad original por lo que esto también afecta al resultado como se muestra en la tabla siguiente.

Lengua	Datos originales	Teclas	Sílabas	Fonemas
Shipibo-Konibo	11,010	110,160	44,065	-
Ashaninka	3,851	63,145	-	90,929
Yanesha	5,682	66,205	26,482	93,604
Yine	4,025	38,290	15,316	-

Tabla 8: Cantidad de datos originales contra los generados

[10] https://github.com/mjpost/sacrebleu

Para el segundo experimento se utilizó el modelo entrenado con los datos generados en base a los errores fonéticos más comunes basados en la información que brindaron los lingüistas. Los resultados de este experimento se pueden ver en la tabla siguiente:

Lengua	BLEU (Línea base)	BLEU	CharacTER (Línea base)	CharacTER
Ashaninka	12.4	1.8	0.1803	0.2428
Yanesha	6.9	1.9	0.2157	0.1276

Tabla 9: Resultado de métricas por errores más comunes de acuerdo a los fonemas

Este segundo experimento mostró unos peores resultados en comparación a la línea base y los resultados anteriores. Este caso podría deberse a que los errores aleatorios son muy diferentes a errores basados en fonética. Este resultado, si bien es negativo en acierto, es positivo para la motivación del trabajo, ya que indica que es necesario considerar otros tipos de fuentes de errores para que el modelo sea más robusto

Para el tercer experimento se utilizó el modelo entrenado con los datos generados en base a los errores por sílabas similares. Como se ven en la Tabla 10, los resultados nuevamente fueron inferiores con respecto a la línea base y se debe a que los errores aleatorios son muy distintos a los errores de sílabas, tal como se dio en el experimento anterior.

Lengua	BLEU (Línea base)	BLEU	CharacTER (Línea base)	CharacTER
Shipibo-konibo	18.5	11.3	0.1937	0.2078
Yine	15.2	4.1	0.2025	0.1956
Yanesha	6.9	6.3	0.2157	0.1944

Tabla 10: Resultado de métricas por errores de sílabas similares

Una vez realizado los experimentos con los conjuntos de datos generados de cada uno de los métodos de aumento de datos, se decidió generar un conjunto de datos de validación y de prueba en base a los 3 tipos de errores, y al ser Yanesha la lengua que posee conjunto de datos con los 3, se procede a realizar el nuevo experimento solo con esta lengua.

Para este experimento se separó cada conjunto de datos en 3 particiones:

- 80% para entrenamiento
- 10% para validación
- 10% para prueba

Luego se combinó cada conjunto de validación y cada conjunto de prueba para que los modelos se entrenen y usen estos nuevos conjuntos combinados (Fonemas: 50%, Teclado 35% y Sílabas: 15%), a diferencia de los experimentos anteriores donde se utilizó el conjunto de pruebas de la línea base. Al tener el mismo conjunto de pruebas se pueden comparar mejor los resultados entre los modelos. Con esta idea en mente se hicieron 2 pruebas por cada tipo de error, la primera donde se prueba el modelo entrenado contra sus datos de prueba correspondiente y ese mismo modelo contra los datos de prueba combinado. Así podemos apreciar cuál modelo ofrece un mejor resultado a nivel general y a nivel de su tipo de error. Para entender mejor el experimento se ve en la Tabla 11 los conjuntos de datos utilizados y los resultados de las métricas de los modelos entrenados con esos conjuntos para la lengua Yanesha.

Entrenamiento	Validación	Prueba	BLEU	CharacTER
Fonemas	Combinado	Fonemas	86.4	0.0167
		Combinado	**52.2**	**0.0700**
Sílabas		Sílabas	12	0.1315
		Combinado	26.6	0.1350
Teclado		Teclado	58	0.0555
		Combinado	41.7	0.1062

Tabla 11: Resultados de la combinación de errores para Yanseha

Entrenamiento	Validación	Prueba	BLEU	CharacTER
Fonemas	Combinado	Fonemas	74.4	0.0334
		Combinado	**54**	**0.0538**
Teclado		Teclado	58.8	0.0402
		Combinado	45.8	0.0982

Tabla 12: Resultados de la combinación de errores para Ashaninka

Al ver los resultados nos damos cuenta que en casi todos los casos cada modelo obtiene buenos resultados cuando es puesto a prueba con su mismo tipo de error a diferencia de la prueba combinada. Esto quiere decir que es más fácil identificar y realizar la corrección para el tipo de error para el cuál fue entrenado. Para el caso de las sílabas, una hipótesis es que la validación favoreció a los casos que no contaban errores generados por sílabas, provocando que al probar con el combinado se obtenga un mejor resultado. Otro punto puede ser la menor proporción de errores, la cual es menor en sílabas con respecto a los otros, ya que se utilizaron: 32,220 entradas de errores fonéticos, 22,554 entradas de errores de teclado y 9,667 entradas de errores de sílabas.

6.2 Aumento de datos

Luego de obtener estos resultados con los modelos entrenados con los conjuntos de datos generados por los métodos de aumento de datos, se decidió entrenar nuevos modelos utilizando conjunto de datos basados en segmentación de sub-palabras. Los 2 elegidos son basados en sílabas y en *byte pair encoding* (BPE).

Para el experimento de sub-palabras con BPE se utilizó el conjunto de datos de errores fonéticos en Yanesha y Ashaninka ya que este fue el que mejor rendimiento obtuvo en las distintas pruebas. Como se mencionó en un capítulo anterior se generaron vocabularios con distintos número de operaciones de mezcla: 2500, 5000, 7500 y 10000. Además se tuvo que realizar modificaciones al modelo de correción que se utilizó en los experimentos anteriores, resultando en un modelo nuevo que usa otro tipo de entrada de datos ya que en los experimentos anteriores era a nivel de caracteres.

Número de operaciones	BLEU	CharacTER
2500	60.6	0.1221
5000	66.5	0.0887
7500	70.4	0.0727

10000	69.3	0.0746

Tabla 13: Resultados con BPE al conjunto de datos de errores fonéticos de Yanesha

En la Tabla 13 vemos los resultados obtenidos para la lengua Yanesha usando el conjunto combinado. Con estos resultados podemos observar que hay un incremento en los resultados con respecto al incremento de número de operaciones de mezclado. Esto puede deberse a que al tener un vocabulario más amplio, el modelo dispone de un abanico mayor de detalles para ofrecer una corrección a la oración. Ya que al ser una mayor cantidad de vocabulario se puede obtener mayor cantidad de información. Sin embargo, hay que considerar que entre más amplio el vocabulario también representa que la red tiene que realizar un mayor procesamiento y toma más tiempo así como se ve en la Tabla 18. Por otro lado, mientras más operaciones "merge" haya, el vocabulario por completo va a ser integrado en el modelo, con pocas particiones, lo cual no es conveniente en una tarea de corrección.

Número de operaciones	Métrica BLEU	Métrica CharacTER
2500	45.9	0.1739
5000	51.1	0.1398
7500	55.8	0.1190
10000	59.4	0.1039

Tabla 14: Resultados con BPE al conjunto de datos de errores fonéticos de Ashaninka

En la Tabla 14 de los resultados obtenidos del modelo entrenado con BPE en Ashaninka podemos encontrar el mismo patrón que en la tabla de resultados de Yanesha. Donde una mayor cantidad de operaciones de mezclado permite obtener mejores resultados en la corrección. Se entiende que es el mismo fenómeno de tener un vocabulario más amplio para obtener la corrección.

Finalmente, se realizó el último experimento donde el conjunto de datos ahora es a nivel de sílabas. Para este enfoque se tuvo que realizar ajustes para que acepte esta nueva entrada. Es importante recalcar que se necesitaba una gran cantidad de memoria RAM para realizar el procesamiento ya que este consideraba las distintas sílabas que se podían formar que eran 5689 sílabas únicas y a su vez los caracteres de las palabras que no se pudieron silabificar para generar la codificación y decodificación necesaria para el modelo. Como se tuvo limitaciones por el hardware requerido, se realizó el experimento con el conjunto de datos de errores fonéticos para la lengua Yanesha y se obtuvieron los siguientes resultados contra el conjunto de datos combinado.

BLEU	CharacTER
50.4	0.1854

Tabla 15: Resultados con sílabas como entrada con el conjunto de datos de errores fonéticos de Yanesha

Podemos notar que utilizando segmentación de sub-palabras para generar la entrada para el modelo de corrección se obtienen buenos resultados en las métricas evaluadas con respecto a entrada en base a caracteres como en la Tabla 16. Esto se debe a que se tiene mayor información al poder representar mejor las oraciones con las sub-palabras, pero hay que tener en cuenta la limitación de hardware que notamos al realizar el experimento con sílabas como entrada. Que a pesar de que se obtuvieron

resultados mejores, la segmentación con BPE obtuvo mejores resultados y un entrenamiento del modelo más rápido como se puede ver en la Tabla 18.

Lengua	Conjunto de datos	Modelo	Métrica BLEU	Métrica CharacTER	Tiempo de entrenamiento
Yanesha	Fonemas	BPE 7500 operaciones	70.4	0.0727	**33.3 minutos**
		BPE 10000 operaciones	69.3	0.0746	44.26 minutos
		Sílabas	50.4	0.1854	51.8 minutos
		Caracteres	**86.4**	**0.0167**	52.6 minutos
Yanesha	Sílabas	Sílabas	34.3	0.2240	113 minutos
Yanesha	Teclado	Sílabas	48.8	0.1441	95.3 minutos

Tabla 16: Comparación de métricas para segmentación en Yanesha

Lengua	Conjunto de datos	Modelo	Métrica BLEU	Métrica CharacTER	Tiempo de entrenamiento
Ashaninka	Fonemas	BPE 7500 operaciones	55.8	0.1190	**22.7 minutos**
		BPE 10000 operaciones	59.4	0.1039	23.3 minutos
		Caracteres	**74.4**	**0.0334**	24.3 minutos

Tabla 17: Comparación de métricas para segmentación en Ashaninka

Lengua	Conjunto de datos	Modelo	Tiempo de entrenamiento	Uso de RAM
Ashaninka	Fonemas	BPE 2500 operaciones	20 minutos	1.57 GB
		BPE 5000 operaciones	25.2 minutos	1.66 GB
		BPE 7500 operaciones	22.7 minutos	1.22 GB
		BPE 10000 operaciones	23.3 minutos	1.34 GB
		Caracteres	24.3 minutos	1.82 GB
Yanesha	Fonemas	BPE 2500 operaciones	32.1 minutos	1.74 GB
		BPE 5000 operaciones	32.15 minutos	1.75 GB
		BPE 7500 operaciones	33.3 minutos	1.92 GB
		BPE 10000 operaciones	44.26 minutos	1.77 GB
		Silabas	51.8 minutos	17.48 GB
	Teclado	Silabas	95.3 minutos	17.61 GB
	Sílabas	Sílabas	113 minutos	17.01 GB

Tabla 18: Tiempos y recursos para entrenamiento de modelos con segmentación de sub-palabras

7 Conclusiones y trabajos futuros

7.1 Conclusiones

- Se construyeron 3 métodos distintos de aumento de datos que son: por cercanía de teclas, por similitud de sílabas y por errores comunes cometidos por los hablantes en base a fonemas similares. De los enfoques mencionados, el de cercanía de teclas se puede realizar sin ayuda externa como silabificadores previamente desarrollados o un análisis de expertos y lingüistas para identificar los errores comunes de los cuales el que mejores resultados obtuvo de las métricas es el método de errores comunes cometidos por los hablantes.
- Se desarrollaron scripts que permiten crear conjunto de datos utilizando sub-palabras (BPE y sílabas) a partir de un conjunto de datos basado en caracteres donde el modelo entrenado con BPE con mayor número de operaciones obtuvo mejores resultados en las métricas y también en recursos de procesamiento.
- Se realizaron distintas pruebas, solo con conjunto de datos utilizando los datos generados sintéticamente y los mismos con una segmentación aplicada como BPE y el uso de sílabas. Obteniendo mejoras en los resultados en comparación de la línea base que utilizaba errores generados por aleatoriedad.
- Se obtuvieron mejores resultados que la línea base propuesta, además que ahora se utilizan distintas entradas para el entrenamiento (sub-palabras) y obtienen mejores resultados con las métricas BLEU y CharacTER en los experimentos hechos con distintos tipos de errores generados.

7.2 Trabajos futuros

Este trabajo deja desarrollos que pueden ser utilizados para distintas lenguas de escasos recursos reemplazando componentes como los silabificadores o una estructura de datos como un diccionario con los errores comunes. De manera que se puedan crear nuevos y más recursos para las distintas lenguas.

Se pueden considerar nuevos métodos para el aumento de datos utilizando funciones de grafema a fonema para utilizar la similitud de fonemas para reemplazar grafemas con fonemas similares. También se pueden utilizar sinónimos de las palabras y así generar aún una mayor cantidad de oraciones, pero se necesitaría tener mapeados todos los sinónimos en la lengua. Finalmente, se pueden combinar los distintos métodos y así generar datos más complejos.

Se obtuvieron buenos resultados con la cantidad de datos generados. Al tener más datos para el entrenamiento, sería interesante probar otros segmentadores de sub-palabras como SentencePiece (*unigram model*) [Kudo, Richardson, 2018] que podrían dar buenos resultados y poder comparar si hay una mejor con respecto a los resultados actuales.

Otro punto para considerar es poder elevar el proyecto a servicios web o paquetes de Python para que puedan ser fácilmente utilizados por la comunidad. Lo cual nos ayudaría a realizar una validación real de la lengua ya que hablantes nativos podrían evaluar si los resultados son correctos y coherentes. También de esta manera, poder

obtener más datos que puedan ayudar a construir un conjunto de datos aún mayor del que ya se dispone.

8 Bibliografía

Aggarwal, C. C. (2018). Neural networks and deep learning. Springer, 10, 978-3.

Allardt, E. (1984). What constitutes a language minority? Journal of Multilingual & Multicultural Development, 5(3-4), 195-205.

Almansor, E. H., & Al-Ani, A. (2018, July). A hybrid neural machine translation technique for translating low resource languages. In *International Conference on Machine Learning and Data Mining in Pattern Recognition* (pp. 347-356). Springer, Cham.

Alva, C. A. y Oncevay, F. A. (2017). Spell-checking based on syllabification and character-level graphs for a Peruvian agglutinative language. En EMNLP 2017 Workshop on Subword and Character Level Models in NLP, SCLeM 2017. ACL Anthology. Recuperado de: https://aclanthology.org/W17-4116/

Ataman, D., Aziz, W., & Birch, A. (2019). A latent morphology model for open-vocabulary neural machine translation. *arXiv preprint arXiv:1910.13890*.

Base de Datos de Pueblos Indígenas u Originarios BDPI. Consulta: 13 de noviembre de 2020. Diponible en: https://bdpi.cultura.gob.pe/lenguas.

Bustamante, G., Oncevay, A., & Zariquiey, R. (2020, May). No data to crawl? monolingual corpus creation from PDF files of truly low-resource languages in Peru. In Proceedings of the 12th Language Resources and Evaluation Conference (pp. 2914-2923).

Chollampatt, S., and Ng, H. T. (2017). Connecting the dots: Towards human-level grammatical error correction. In Proc. of the 12th Workshop on Innovative Use of NLP for Building Educational Applications.

Etoori, P., Chinnakotla, M., & Mamidi, R. (2018, July). Automatic spelling correction for resource-scarce languages using deep learning. In Proceedings of ACL 2018, Student Research Workshop (pp. 146-152).

Faust, N. (1973). Lecciones para el aprendizaje del idioma shipibo-conibo. Documento de trabajo, 1.

Forcada Mikel (2006), Open-source machine translation: an opportunity for minor languages. Proceedings of Strategies for developing machine translation for minority languages (5th SALTMIL workshop on Minority Languages).

Gage, P. (1994). A new algorithm for data compression. C Users Journal, 12(2), 23-38.

Hartmann, W., Le, V. B., Messaoudi, A., Lamel, L., & Gauvain, J. L. (2014). Comparing decoding strategies for subword-based keyword spotting in low-resourced languages. In INTERSPEECH (pp. 2764-2768).

Hernández-García, A., & König, P. (2018). Data augmentation instead of explicit regularization. arXiv preprint arXiv:1806.03852.

Jara Males, P., & Gonzalez Acero, C. (2015). Estrategias institucionales y modalidades de atención en servicios para la inclusión social de poblaciones vulnerables. Inter-American Development Bank.

Karakanta, A., Dehdari, J., & van Genabith, J. (2018). Neural machine translation for low-resource languages without parallel corpora. *Machine Translation*, *32*(1), 167-189.

Kudo, T., & Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226.

Lara, César. (2020). Corrección ortográfica de lenguas amazónicas usando redes neuronales secuencia a secuencia (Tesis de maestria). Pontificia Universidad Católica del Perú, Lima.

Li, H., Wang, Y., Liu, X., Sheng, Z., & Wei, S. (2018). Spelling error correction using a nested rnn model and pseudo training data. arXiv preprint arXiv:1811.00238.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

Mayor, A., Alegria, I., de Ilarraza, A. D., Labaka, G., Lersundi, M., & Sarasola, K. (2009). Evaluación de un sistema de traducción automática basado en reglas o por qué BLEU sólo sirve para lo que sirve. Procesamiento del Lenguaje Natural, (43), 197-205.

Mikolov, T., Kombrink, S., Burget, L., Černocký, J., & Khudanpur, S. (2011, May). Extensions of recurrent neural network language model. In 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 5528-5531). IEEE.

Náplava, J., & Straka, M. (2019). Grammatical Error Correction in Low-Resource Scenarios. arXiv preprint arXiv:1910.00353.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics (pp. 311-318).

Pereira-Noriega J., Mercado-Gonzales R., Melgar A., Sobrevilla-Cabezudo M., Oncevay-Marcos A. (2017) Ship-LemmaTagger: Building an NLP Toolkit for a Peruvian Native Language. In: Ekštein K., Matoušek V. (eds) Text, Speech, and Dialogue. TSD 2017. Lecture Notes in Computer Science, vol 10415. Springer

Post, M. (2018). A call for clarity in reporting BLEU scores. arXiv preprint arXiv:1804.08771.

Ragni, A., Knill, K. M., Rath, S. P., & Gales, M. J. (2014, September). Data augmentation for low resource languages. In *INTERSPEECH 2014: 15th Annual Conference of the International Speech Communication Association* (pp. 810-814). International Speech Communication Association (ISCA).

Roberto Zariquiey, Harald Hammarström, Mónica 380 Arakaki, Arturo Oncevay, John Miller, Aracelli Gar- 381 cía, and Adriano Ingunza. 2019. Obsolescencia 382 lingüística, descripción gramatical y documentación 383 de lenguas en el perú: hacia un estado de la cuestión. 384 Lexis, 43(2):271–337.

Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909.

Shah, K., & de Melo, G. (2020). Correcting the Autocorrect: Context-Aware Typographical Error Correction via Training Data Augmentation. arXiv preprint arXiv:2005.01158.

Tensorflow (2020). https://www.tensorflow.org/. Consultado: 29 de noviembre del 2020

Wang, W., Peter, J. T., Rosendahl, H., & Ney, H. (2016, August). Character: Translation edit rate on character level. In Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers (pp. 505-510).

Wu, P., Lu, Z., Zhou, Q., Lei, Z., Li, X., Qiu, M., & Hung, P. C. (2019). Bigdata logs analysis based on seq2seq networks for cognitive Internet of Things. Future Generation Computer Systems, 90, 477-488.

Wu, Y., & Zhao, H. (2018). Finding better subword segmentation for neural machine translation. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data* (pp. 53-64). Springer, Cham.

Zhou, Y., Porwal, U., & Konow, R. (2017). Spelling correction as a foreign language. arXiv preprint arXiv:1705.07371.

9 Apéndice A

Parámetros	Valor
Optimizador	Adam
Pérdida	Categorical Cross-entropy
Métrica	Precisión
Épocas	100
Tamaño de bacth	64

Tabla 19: Parámetros de configuración del modelo