PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

Corrección ortográfica de lenguas Amazónicas usando redes neuronales secuencia a secuencia.

Tesis para optar el Grado Académico de:
Magíster en Informática con mención en Ciencias de la Computación

AUTOR
César Jesús Lara Avila

ASESOR

Mg. Félix Arturo Oncevay Marcos

LIMA-PERÚ
2020

Resumen

De acuerdo a la Base de Datos Oficial de Pueblos Indígenas u Originarios (BDPI) [1], el Perú cuenta con 55 pueblos indígenas, identificados hasta la fecha; que hablan al menos 47 lenguas originarias y que según el Documento Nacional de Lenguas Originarias del Perú [2] están divididos en 19 familias lingüísticas, siendo las familias Pano y Arawak las que presentan una mayor cantidad de lenguas, ambas con 10 lenguas.

En este trabajo, se plantea un modelo de corrección ortográfica utilizando modelos de redes neuronales profundas, a nivel de caracteres, en lenguas de las dos familias antes mencionadas: Shipibo-Konibo de la familia Pano y Yanesha, Yine y Ashaninka para la familia Arawak. Para ello se han realizamos experimentos en conjuntos de datos obtenidos de páginas como PerúEduca, incorporando errores ortográficos cometidos a nivel de caracteres, en modelos secuencia a secuencia (seq2seq) que han demostrado recientemente ser un marco exitoso para varias tareas de procesamiento de lenguaje natural [6], incluyendo el proceso de corrección ortográfica.

Dedicado a Irene, Chalo y Clau.

Agradecimientos

A Dios, por sus bendiciones y por haberme traído hasta estas instancias y sino a sido Él a lo que lo ha hecho posible.

A Mg. Arturo Oncevay, por su orientación, apoyo y paciencia a lo largo de este trabajo.

A Gina Bustamante , Kervy Rivas, John Miller y Erasmo Gómez, el equipo de Procesamiento de Lenguaje Natural de la PUCP, por el apoyo y las sugerencias.

Índice de figuras

Índice de cuadros

Índice general

Introducción

2.1. Definición del problema

En el Perú, todas las lenguas originarias se identifican como lenguas minoritarias o de escasos recursos computacionales, ya que cuentan con pocos o ningún tipo de datos anotados que puedan ser procesados directamente por algoritmos computacionales. Unas de las principales causas de este problema se debe a que las lenguas nativas peruanas son de tradición principalmente oral que se transmiten de generación a generación, por lo que la estandarización y normalización de su escritura es una tarea compleja, que dificulta su enseñanza, en forma escrita para sus comunidades.

La tarea de corregir la ortografía es un desafío para lenguas amazónicas con escasos recursos, ya que cada una tiene su propio vocabulario y propiedades a nivel morfológico, fonológico, etc, que pueden variar incluso más si es que una lengua proviene de ámbitos geográficos distintos. Esta tarea se hace más díficil cuando las lenguas son habladas por minorías amazónicas, ya que no tienen la atención necesaria de parte de la sociedad en el desarrollo de tecnologías útiles para apoyar su aprendizaje y evitar su extinción.

En este sentido, de acuerdo a lo escrito por Mikel L. Forcada [7], la aplicación de tecnologías del lenguaje como la traducción automática a lenguas minoritarias constituye un avance en visibilizar estas lenguas en contextos informáticos, aumentar la alfabetización o propiciar la estandarización de una lengua sobre la base de una variedad en particular, que es un factor importante en el estado y desarrollo de las lenguas minoritarias.

En este escenario, la institución internacional ILV (Instituto Lingüístico de Verano) que ha trabajado con grupos minoritarios del Perú desde 1946, enfocándose en capacitar, documentar, preparar material educativo sobre lenguas minoritarias para su difusión, con ayuda del Ministerio de Educación, realizó una guia de aprendizaje de la lengua amázonica Shipibo-Konibo [9] en un primer acercamiento de difundir y normalizar el lenguaje.

Un avance mayor lo dió el Ministerio de Educación el 13 de junio del 2015 cuando se oficializaron 24 alfabetos de lenguas originarias a través de la Resolución Ministerial $N°303 - 2015 - MINEDU$ para consolidar un sistema de escritura uńico por cada lengua, permitiendo que estos alfabetos oficiales faciliten la preparación de un material educativo normalizado en un contexto de educación intercultural bilingüe, el cual es política de Estado según el decreto $N°005 - 2017 - MC$.

Asimismo, la ley $N°29735$, en su artículo $2°$, declara de interés nacional el uso, preservación, desarrollo, recuperación, fomento y difusión de las lenguas originarias del Perú, mientras que en su artículo $9°$ se oficializa el uso de las lenguas originarias en las zonas en donde estas sean predominantes con programas de alfabetización mediante modalidad intercultural bilingüe, hace necesario, fomentar y normalizar el uso de estas lenguas en forma escrita.

En el campo computacional de la Lingüística Computacional y del Procesamiento de Lenguaje Natural (PLN), el proceso de normalización de una lengua está estrechamente relacionado a la corrección ortográfica automática, o cómo un programa puede identificar una escritura errónea y sugerir al usuario alternativas correctas o estandarizadas. Pese a la dificultad de la tarea, se han realizado esfuerzos para desarrollar tecnologías para lenguas como el Shipibo- Konibo [10, 11, 12, 13], pero aún es insuficiente ante la enorme variedad de lenguas existentes en el país y la ausencia de especialistas de educación.

Ante ello, la presente tesis busca implementar algoritmos de corrección ortográfica para cuatro lenguas peruanas: Shipibo-Konibo, Yanesha, Yine y Ashaninka. Los métodos funcionan a nivel de caracteres utilizando modelos de redes neuronales, a partir de datos recolectados y procesados de distintas fuentes, con el uso de un algoritmo de generador de errores y un modelo secuencia a secuencia o seq2seq.

2.2. Objetivos

2.2.1. Objetivo general

Implementar algoritmos de corrección ortográfica usando aprendizaje profundo, en particular los modelos secuencia a secuencia o seq2seq, para cuatro lenguas peruanas: Shipibo-Konibo de la familia lingüistica Pano, y Yanesha, Yine y Ashaninka de la familia lingüistica Arawak.

2.2.2. Objetivos específicos

Con el fin de lograr el objetivo general se plantean los siguientes objetivos específicos:

- **Objetivo 1** Recolectar y procesar datos léxicos y corpus textuales de las lenguas originarias peruanas Amazónicas mencionadas en el objetivo general.
- **Objetivo 2** Implementar algoritmos de generación de errores ortográficos sintéticos.
- **Objetivo 3** Implementar algoritmos basado en modelos secuencia a secuencia o seq2seq a nivel de caracteres que efectúen la corrección de una palabra mal escrita en una lengua originaria mencionada.

2.3. Resultados esperados

- **Para el Objetivo 1**
  - Obtener un conjunto de datos a nivel palabras y oraciones, de las lenguas mencionadas en este trabajo, que después de una fase de limpieza y procesamiento, estén disponibles para las siguientes etapas.
- **Para el Objetivo 2**
  - Implementación de un algoritmo de generación de errores ortográficos.

- **Para el Objetivo 3**
  - Implementación de un algoritmo de división de datos: datos de entrenamiento, evaluación y prueba.
  - Corpus paralelos de palabras y oraciones con errores ortográficos y sin errores ortográficos, generados por el algoritmo de generación de errores.
  - Modelo y/o algoritmo de corrección de palabras y oraciones, utilizando modelos de redes neuronales profundas secuencia a secuencia o seq2seq y mecanismos de atención a nivel de caracter
  - Informe de experimentación con métricas de evaluación para el modelo de corrección ortográfica propuesto.

2.4. Herramientas y métodos

2.4.1. Herramientas

Las principales herramientas utilizadas el desarrollo de este trabajo fueron:

- Para la recolección de datos y preprocesamiento de datos se utilizó Beautiful Soup [1] y PDFMiner [2].
- Para el almacenamiento de los datos usamos PyMongo [3] una herramienta para interactuar con MongoDB [4] desde Python.
- La implementación de algunos scripts para manipular, filtrar líneas de texto, fueron escritos en awk. [5].
- Para el desarrollo de la experimentación de los modelos seq2seq se utilizó Keras [6] y su API funcional.
- Se utiliza el visualizador Netron [7], para visualizar la arquitectura de red del corrector ortográfico.
- Todos los modelos de secuencia a secuencia implementados, se realizaron con Colaboratory [8] de Google.

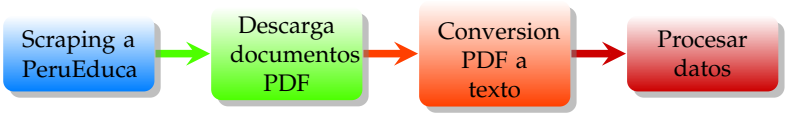
2.4.2. Metodología

En base a los objetivos planteados, se detalla la metodología a desarrollar:

- **Para el Objetivo 1**

[1] https://www.crummy.com/software/BeautifulSoup/
[2] http://www.unixuser.org/~euske/python/pdfminer/index.html
[3] https://github.com/mongodb/mongo-python-driver
[4] https://www.mongodb.com/
[5] https://www.gnu.org/software/gawk/
[6] https://keras.io/
[7] https://github.com/lutzroeder/netron
[8] https://colab.research.google.com/notebooks/welcome.ipynb

  - Construir un web crawler para recolectar datos de las lenguas originarias peruanas Amazónicas en formato PDF. Se realizó luego una conversión de documentos PDF a texto utilizando herramientas en Python como PDFMiner.
  - Los datos convertidos a texto, se escriben en formas de palabras y oraciones para cada una de las lenguas mencionadas, para luego realizar un proceso de limpieza y una análisis exploratorio de los datos conseguidos.
    Se puede resumir todo el proceso, hasta el procesamiento y limpieza de datos en el siguiente esquema:



- **Para el Objetivo 2**
  - Implementar un algoritmo de generación de errores a nivel palabras y oraciones. El algoritmo analiza diferentes formas en que pueden ocurrir los errores ortográficos, cada uno de los cuales tiene la misma probabilidad: reemplaza un caracter con un caracter aleatorio, elimina un caracter, agrega un caracter al azar, intercambia 2 caracteres, elimina espacios en blanco, entre otros.
- **Para el Objetivo 3**
  - Implementación de modelos seq2seq para corrección de palabras y oraciones, utilizando arquitecturas de redes neruaonales secuencia a secuencia y mecanismos de atención.
  - A pesar de que aun no existe una mejor métrica de evaluación y de rendimiento ya que cada métrica depende de los objetivos de la investigación y la aplicación [72], [73], consideramos algunas métricas usuales como la exactitud, que indica el porcentaje de oraciones que se clasificaron como escritas correctamente, del número total de entradas.
  - Propuesto en el estudio [74] utilizamos BLEU también como métrica de evaluación y la métrica `CharacTER` [75].

  La red neuronal recibe un corpus paralelo de palabras y oraciones junto con los errores generados y la red debe ser capaz de devolver las palabras y oraciones corregidas.

2.5. Alcance y limitaciones

2.5.1. Hipótesis

Es posible desarrollar algoritmos de corrección ortográfica para las lenguas nativas peruanas de la Amazonía incluidad en el programa Educación Básica Bilingüe en NOPOKI: Shipibo-Konibo, Asháninka, Yine, Yanesha, a partir del aprovechamiento de conocimiento lingüístico pedagógico y computacional.

2.5.2. Justificación

El presente trabajo se justifica por presentar un enfoque de cómo el procesamiento de lenguaje natural y el aprendizaje profundo pueden asistir en el proceso de normalización del sistema de

escritura de las lenguas nativas peruanas, en el caso de implementar algoritmos y modelos de corrección ortográfica.

2.5.3. Limitaciones

Para la realización de este proyecto se han identificado una limitación referente al tamaño del corpus ya que este es pequeõ y no cuenta con muchos datos. Para ello se ha planteado alimentar con nuevos datos de diferentes fuentes de información como se muestra en el trabajo reciente de Gina Bustamante y Arturo Oncevay [78]. Otra limitación que se considera es la falta de conocimiento lingüístico de las lenguas participantes por lo que se ha optado por contar con la ayuda adicional de hablantes de las lenguas.

Marco conceptual

En el presente capítulo se mencionarán algunos conceptos que se tendrán en cuenta durante el desarrollo de la tesis.

3.1. Lenguas de bajos recursos

El término lengua de bajos recursos es una denominación amplia, que tiene diversas connotaciones según el contexto en que es empleado y como menciona en el artículo de Forcada [7], también se les conoce como lenguas minoritarias, lenguas menores o lenguas de escasos recursos. Para este trabajo, el término se refiere a aquellas lenguas que no tienen una presencia muy amplia en la internet, ni cuentan con los recursos necesarios para procesarlas de forma automatizada por computadoras.

Asimismo como se menciona en el artículo de Streiter, Scannell y Stuflesser [8] a veces estas lenguas no poseen un alfabeto computarizado que les permita generar software que utilice el alfabeto, ni que se generen herramientas electrónicas como diccionarios, sistemas de recuperación de información o correctores ortográficos, que ayudan a la preservación de la cultura que la emplea.

3.2. Clasificación sobre los estados de vitalidad de las lenguas originarias

Las 47 lenguas originarias vigentes pueden ser agrupadas en lenguas vitales, lenguas en peligro y lenguas seriamente en peligro de acuerdo al Documento Nacional de Lenguas Originarias del Perú . De este modo, se obtuvo la siguiente clasificación sobre los estados de vitalidad de las lenguas originarias [2]:

- Lenguas vitales. Son habladas por todas las generaciones de una comunidad lingüística, cuya transmisión intergeneracional es ininterrumpida. Además, las lenguas en este estado son habladas en la mayoría de ámbitos comunicativos. Por ejemplo, el Shipibo-Konibo se considera una lengua vital.
- Lenguas en peligro. Son aquellas habladas mayoritariamente por los adultos de una comunidad lingüística. Asimismo, las lenguas en este estado suelen estar restringidas a ciertos ámbitos comunicativos y la transmisión intergeneracional puede ser parcial en algunas comunidades. Por ejemplo el Asheninka del Pajonal y el Yanesha se consideran lenguas en peligro.
- Lenguas seriamente en peligro. Son aquellas habladas mayoritariamente por adultos mayores de forma parcial, poco frecuente y en ámbitos comunicativos muy restringidos. Además, ya no son transmitidas a las nuevas generaciones. Por ejemplo el Asheninka del

Alto Perené y el Yine en su dialecto `mantxinerï` se considera lenguas en seriamente en peligro.

3.3. Corrección Ortográfica

La corrección ortográfica y gramatical es la tarea de corrección automática de errores en texto escrito. El trabajo de Soni y Thakur [14], propone una esquema de clasificación de errores basados en la frecuencia, niveles y naturaleza de un error, validez del texto y errores superpuestos.

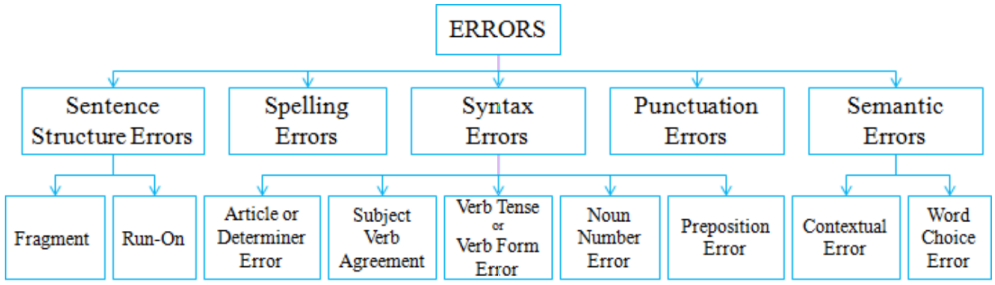


Figura 3.1: Clasificación de errores gramaticales propuesto por Soni y Thakur, [14].

.

De acuerdo al libro de Jurafsky y Martin [15], la corrección ortográfica a menudo se considera desde dos perspectivas. La correción ortográfica de no palabras [1] , la corrección de la ortografía de palabras reales y errores cognitivos.

Del trabajo de Karen Kukich [16], se debe hacer una distinción entre las tareas de detección de errores y corrección de errores y analizar el problema de la corrección de palabra en tres tipos de problemas: (1) detección de errores de no palabras, (2) corrección de errores de palabras aisladas y (3) corrección de errores dependientes del contexto.

3.3.1. Técnicas de detección de errores

En muchos sistemas de corrección ortográfica y gramatical, antes de que alguna corrección es llevada a cabo, se realiza un proceso de detección de errores en una oración de entrada para extraer probables palabras erróneas. De acuerdo al artículo de Kukich [16] dos técnicas son usadas para la detección de errores ortográficos de palabras incorrectas que no existen en un diccionario dado.

N-gramas

Para la tarea de detección de errores, el análisis N-gramas estima la verosimilitud que una entrada dada se escribirá correctamente. Para llevar a cabo esta tarea se realiza una estadística $n-$grama precalculada desde una fuente del lenguaje para tareas comparativas. Es un modelo que asigna una probabilidad a una secuencia de símbolos: caracteres o palabras.

[1] En inglés: non-words

De [17], para la tarea de detección de errores, el análisis de n-gramas estima la probabilidad de que una entrada dada se deletree correctamente. Para lograr esto, se proporciona una estadística de n-gramas precalculada de los recurso de lenguaje para tareas comparativas. La elección de n depende del tamaño del corpus de entrenamiento.

Dictionary lookup

Desde el trabajo de Ahmadi [17], es una técnica básica empleada para comparar cadenas de entradas con la entradas de un recurso de un lenguaje, como un vocabulario o corpus. El lenguaje en cuestión debe contener todas las formas de las palabras y se debe actualizar periódicamente. Si una palabra dada no existe en esas fuentes del lenguaje, es marcada como una posible palabra incorrecta.

Una técnica común para obtener acceso rápido a un diccionario es el uso de una tabla hash. Para buscar una cadena de entrada, se calcula su dirección hash y se recupera la palabra almacenada en esa dirección en la tabla hash preconstruida. Para la tarea de detección de errores, si la palabra almacenada en la dirección hash es diferente de la cadena de entrada o es nula, se indica un error ortográfico [16].

Tareas en esta técnica para mejorar, es reducir el tamaño de las fuentes y mejorar el rendimiento de búsqueda a través de análisis morfológico y algoritmo de coincidencia de patrones.

3.3.2. Técnicas de corrección de errores

El problema de corrección de palabras aisladas conlleva a tres subprocesos: (1) detección de un error, (2) generación de candidatos de corrección y (3) clasificación de cadidatos de corrección [17].

Muchas técnicas tratan estos procesos de manera separada y los ejecuta en secuencia, otras técnicas que incluyen probabilísticas y de redes neuronales combinan los tres subprocesos en un solo paso, calculando una medida de similaridad o probabilidad entre una cadena de entrada y cada palabra o algún subconjunto de palabras en el diccionario. Si la palabra del diccionario mejor clasificada no es idéntica a la cadena de entrada, se indica un error y se calcula una lista clasificada de posibles correcciones [16].

Se puede agregar todos estos enfoques en distintas clases:

3.3.2.1. Técnicas de distancia de edición mínima

El término distancia mínima de edición se define como el mínimo número de operaciones de edición que en muchos sistemas son definidos como inserción, eliminación, subsitución y transposición, que son requeridos para transformar una entrada incorrecta en otra que es la más probable. Los algoritmos más conocidos de esta técnica son los de, Hamming [18], Winkler [19], Wagner-Fisher [20], Damerau-Levenshtein [21] y Levenshtein [22].

El valor de distancia de Levenshtein describe el número mínimo de eliminaciones, inserciones o sustituciones que se requieren para transformar una cadena en otra. A diferencia de la distancia de Hamming, la distancia de Levenshtein funciona en cadenas con una longitud distinta. Se

puede aplicar la metodología de la programación dinámica para encontrar la distancia de Levenshtein entre dos cadenas.

Las técnicas de distancia mínima de edición se aplican a casi todas las tareas de corrección ortográfica, incluyendo edición de texto.

3.3.2.2. Técnicas basadas en reglas

La técnicas basadas en reglas son algoritmos o programas heurísticos que intentan representar el conocimiento del patrón de errores de ortográficos comunes en forma de reglas para transformar errores ortográficos en palabras válidas. Esas reglas generalmente se basan en características morfológicas del lenguaje.

El enfoque clásico de la verificación gramatical es diseñar manualmente reglas gramaticales como se muestra en [23]. Estas reglas de alta calidad están diseñadas por expertos lingüísticos que pueden proporcionar una explicación detallada de los errores marcados, lo que hace que el sistema sea extremadamente útil para el aprendizaje de lenguajes asistido por computadora. Pero el mantenimiento manual de cientos de reglas gramaticales es bastante tedioso.

Al analizar los errores ortográficos más comunes, algunos investigadores han intentado crear una base de errores para la tarea de corrección: [24], [25].

3.3.2.3. Técnicas probabílisticas

Enfoques basados en modelos probabilísticos que aprenden una representación adecuada de los datos también se han utilizado para tareas de corrección de errores, para ello se crea un modelo probabilístico sobre secuencias de caracteres y palabras, conocido como modelo de lenguaje.

Un modelo de lenguaje proporciona una forma de determinar la probabilidad de una secuencia de palabras. Por ejemplo, un modelo de lenguaje de `n-gramas`, funciona como una técnica probabilística para la corrección de errores y determina la probabilidad de una secuencia de palabras $\mathbb{P}(w_1, \ldots, w_m)$ al observar la probabilidad de cada palabra dadas las palabras anteriores:

$$\mathbb{P}(w_1, \ldots, w_m) \approx \prod_{i=1}^{m} \mathbb{P}(w_i | w_{i-(n-1)}, \ldots, w_{i-1}). \tag{3.1}$$

Los modelos de lenguaje son particularmente interesantes en PNL ya que pueden proporcionar información contextual adicional a situaciones en las que una predicción podría ser semánticamente similar, pero sintácticamente diferente.

Estos modelos se usan a menudo cuando se genera el lenguaje y para la adaptación del dominio donde puede haber grandes cantidades de datos de texto sin etiquetar y datos etiquetados limitados.

Kernighan [28] y Church y Gale [29] idearon un algoritmo para corregir errores ortográficos utilizando `el modelo de canal ruidoso`, casi al mismo tiempo. El algoritmo utiliza técnicas de distancia de edición mínima para generar un conjunto de candidatos de un solo error y identificar

un error específico dentro de cada candidato, seguido del uso de Computación Bayesiana para clasificar a los candidatos obtenidos .

Otra técnica común es el modelo Log-Lineal [30], que toma un enfoque muy diferente a los n-gramas basados en conteos. El modelo calcula la probabilidad creando un vector de características que describe el contexto usando diferentes características y calcula un vector de puntuación que corresponde a la probabilidad de cada letra o palabra.

El modelo Oculto de Markov (HMM), ha demostrado una gran habilidad para modelar el lenguaje humano [31], pero desde que el HMM asume independencia condicional de la palabra anterior, excepto en la última, no es un modelo para modelar dependencias de largas distancias, algo que las redes neuronales recurrentes (RNN) si pueden hacer [32]. En efecto las redes neuronales se basan en la distribución de probabilidad del lenguaje y han mostrado un éxito en diferentes tareas relacionadas con el procesamiento del lenguaje natural.

Este trabajo se centra en las redes neuronales que utilizan el modelo seq2seq acompañados de un mecanismo de atención para la corrección de errores.

3.4. Aprendizaje profundo como herramienta

El aprendizaje automático se puede definir como el proceso de resolver un problema práctico al 1) recopilar un conjunto de datos y 2) construir un modelo de algoritmos estadístico basado en ese conjunto de datos[33].

El rendimiento de los algoritmos de aprendizaje automático tradicionales, depende en gran medida de la representación de los datos que reciben. Por lo tanto, el conocimiento previo del dominio, la ingeniería de características y la selección de características son puntos a tener en cuenta en el rendimiento de las salidas o resultados. En efecto en el artículo de LeCun, Bengio y Hinton [34] se mencionan que las técnicas de aprendizaje de máquina son limitadas en su habilidad de procesar los datos en su forma natural, por lo que se requiere una considerable experiencia en el dominio para diseñar un sistema que transformen la data sin procesar a una adecuada representación interna, desde el cual el sistema de aprendizaje pueda detectar o clasificar patrones en los datos.

Desafortunadamente, para muchas tareas con varios formatos de entrada, como imágenes, vídeo, audio y texto, es muy difícil saber qué tipo de características se deben extraer y mucho menos su capacidad de generalización para otras tareas que están más allá de una aplicación actual. El diseño manual de características para una tarea compleja requiere una gran cantidad de dominio, tiempo y esfuerzo.

El progreso para resolver este inconveniente fue muy lento, especialmente para problemas complicados a gran escala, lo que motivó a diseñar enfoques de representación.

El aprendizaje profundo proporciona una manera eficiente de tratar las representaciones a partir de datos utilizando una abstracción llamado grafo computacional y técnicas de optimización numérica. Muchas empresas de tecnología como Google o Facebook han publicado implementaciones de framework de grafos computacionales y bibliotecas construidas para implementar algoritmos del aprendizaje profundo y del PLN. Los métodos de aprendizaje profundo son métodos de aprendizaje de representación [35].

Las arquitecturas de redes neuronales se aplican a una variedad de problemas debido a su poder de representación. El teorema de aproximación universal ha demostrado que una red neuronal de alimentación directa con una sola capa puede aproximarse a cualquier función continua con solo restricciones limitadas en el número de neuronas en la capa [36]. En este contexto si varias capas de neuronas se apilan consecutivamente y se entrenan conjuntamente con el algoritmo de backpropagation, la red aprende múltiples funciones no lineales para adaptarse al conjunto de datos de entrenamiento. El aprendizaje profundo se refiere a muchas capas de redes neuronales conectadas en secuencia [37].

En una red neuronal de retroalimentación simple, como el perceptrón multicapa (MLP), suponemos que todas las entradas y salidas de la red son independientes entre sí, lo cual no es el mejor método para muchas tareas, particularmente para el Modelado del Lenguaje. Al compartir parámetros en diferentes partes de un modelo, podemos convertir un MLP en una red neuronal recurrente (RNN).

La forma más simple de una red neuronal recurrente es un MLP con un conjunto de activaciones de unidades ocultas que se retroalimentan en la red junto con las entradas, para que las activaciones puedan seguir un ciclo. Por lo tanto, a diferencia de las redes de retroalimentación, que son amnésicas con respecto a su pasado reciente, el MLP permite que la red realice procesamiento temporal y aprenda secuencias, que son características importantes para el Modelado del Lenguaje.

La siguiente figura 3.2 ilustra una red de retroalimentación y una red recurrente.

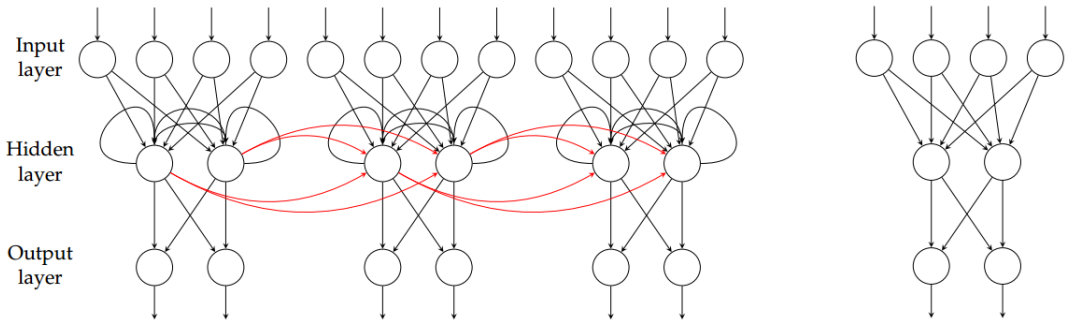


Figura 3.2: Red neuronal recurrente (izquierda) vs. una red neuronal usual (derecha), [17].

Las RNN han sido la arquitectura subyacente para las redes neuronales basadas en MLP. Estas arquitecturas son redes neuronales de aprendizaje profundo y evolucionaron a partir del punto de referencia establecido por innovaciones anteriores como Word2Vec.

Un nuevo enfoque que ha aparecido ahora es ELMo (Embeddings from Language Models) que utiliza RNN para proporcionar embeddings que abordan la mayoría de las deficiencias de los enfoques anteriores [38].

Las arquitecturas RNN siguen siendo bastante avanzadas y vale la pena seguir investigando.

Hasta el 2018 seguían siendo la arquitectura líder para PNL, en cambio, la principal tendencia arquitectónica para el aprendizaje profundo del PNL en 2019 y años posteriores podría ser el Transformer [39].

Pese a ello dado que un RNN realiza un procesamiento secuencial mediante el modelado de

unidades en secuencia, tiene la capacidad de capturar la naturaleza secuencial inherente presente en un lenguaje, donde las unidades son caracteres, palabras o incluso oraciones. Como las palabras en un lenguaje desarrollan su significado semántico en base a las palabras anteriores en la oración, las RNN están hechas a medida para modelar tales dependencias de contexto en el lenguaje y tareas similares de modelado de secuencias, lo que resulta ser una fuerte motivación utilizar RNN sobre otras redes neuronales como las CNN en estas áreas.

Otro factor que contribuye a la idoneidad de RNN para las tareas de modelado de secuencias reside en su capacidad para modelar la longitud variable del texto, incluidas oraciones muy largas, párrafos e incluso documentos [40]. A diferencia de las CNN, las RNN tienen pasos computacionales flexibles que proporcionan una mejor capacidad de modelado y crean la posibilidad de capturar un contexto ilimitado. Esta capacidad para manejar el ingreso de una longitud arbitraria se convirtió en uno de los puntos más importantes del por qué se utilizan RNN [41].

3.5. Redes Neuronales Recurrentes (RNN)

Algunos aspectos importantes de las RNN son:

- La naturaleza del RNN permite a la red comprimir toda la memoria en un espacio de baja dimensión lo que conlleva a desarrollar una característica importante de las redes recurrentes: formar memoria a corto plazo [42].
- RNN es un tipo de red neuronal, que puede procesar datos secuenciales con longitud variable. Los elementos en las secuencias están relacionados entre sí y que su orden es importante [43].
- Los RNN se entrenan a través del BPTT [44].
- Los RNN sufren el problema de la desaparición del gradiente. Una forma simple de limitar la explosión de gradiente es forzar los gradientes a un rango específico.
- Se puede establecer una longitud máxima de secuencia para el procedimiento de entrenamiento y limitar la cantidad de cómputo.
- Son propensas al sobreajuste. El dropout al ser una técnica de regularización que debe modificarse en este escenario.. En este caso podemos reutilizar la misma máscara en cada paso para evitar la pérdida de información entre los pasos de tiempo [45].

3.5.1. Redes de memoria a corto y largo plazo (LSTM)

Propuesta por Hochreiter y Schmidhuber [46], han sido refinadas y popularizadas por muchas personas como Alex Graves [47, 48]. Trabajan muy bien en una gran variedad de problemas que no son resolubles con redes recurrentes tradicionales y ahora son ampliamente utilizadas. Este tipo de redes se desvían de las arquitecturas tradicionales, ya que presentan el concepto de `celda memoria` que les permite aprender a preservar o actualizar su estado en cada iteración de recurrencia.

Una red LSTM está compuesta por celdas de memoria y puertas, para calcular los estados internos. Las 3 puertas controlan el comportamiento de las celdas de memoria: la puerta de

entrada i, la puerta del olvido f y la puerta de salida o. Estas puertas determinan con precisión que parte de un vector dado se debe tener en cuenta en un paso de tiempo de tiempo específico t

Algunos autores recomiendan inicializar los estados LSTM iniciales con un vector 0 [49] y inicializar el sesgo de la puerta de olvido en 1 para recordar más al inicio [50].

3.5.2. Unidad Recurrente Cerrada (GRU)

En 2014, Kyunghyun Cho presentó una alternativa a las redes LSTM, que se llamó unidad recurrente cerrada (GRU) que tiene un similar o mejor desempeño que las LSTM, pero con menos paramétros y operaciones [51].

Una GRU combina las puertas en el LSTM para crear una regla de actualización más simple con una capa menos aprendida, lo que reduce la complejidad y aumenta la eficiencia. La elección entre usar LSTM o GRU se decide en gran medida empíricamente. El GRU usa menos parámetros, por lo que generalmente se elige cuando el rendimiento es igual entre las arquitecturas LSTM y GRU.

De manera similar a establecer el sesgo de la puerta de olvido de una LSTM para mejorar la memoria en las primeras etapas, los sesgos de puerta de reinicio de GRU se pueden establecer en -1 para lograr un efecto similar [50].

3.6. Modelo secuencia a secuencia

En el artículo de Sutskever, Vynials y Q.V.Le [52] se introduce un método llamado secuencia a secuencia o seq2seq , que utiliza las RNN de manera adecuada para resolver distintos problemas del PNL, como la traducción automática, reconocimiento de voz, resúmenes de texto, corrección ortográfica, etc. Ver figura 3.3.

El modelo seq2seq intenta aprender una red neuronal que predice una secuencia de salida a partir de una secuencia de entrada. Las secuencias son un poco diferentes de los vectores tradicionales, porque una secuencia implica un orden de eventos y el tiempo es una forma intuitiva de ordenar eventos.

Dichos modelos constan de dos componentes: un codificador que calcula una representación para una oración de entrada y un decodificador que genera una palabra objetivo a la vez.

Es decir en el proceso de codificación, la secuencia de entrada $x = (x_1, x_2, \dots x_T)$ se alimenta en el codificador, que es básicamente un modelo RNN. Luego, a diferencia de un RNN simple donde se toma en cuenta la salida de cada unidad de estado, solo se mantiene la unidad oculta de la última capa h_t. Este vector se llama vector de contexto c y pretende contener una representación de la oración de entrada:

$$h_t = LSTM(h_{t-1}, x_t)$$
$$c = \tanh(h_T)$$

donde h_t es un estado oculto en el tiempo t, y c es el vector de contexto de las capas ocultas del codificador.

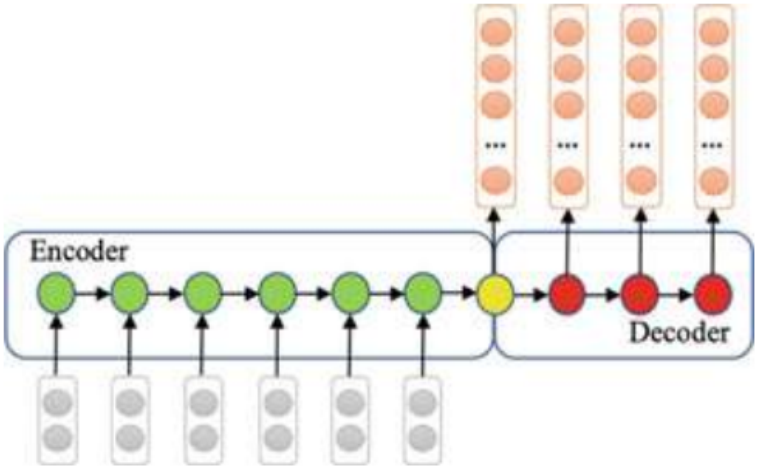


Figura 3.3: Modelo seq2seq con una RNN basado en codificador-decodificador, [37].

Por otro lado, al pasar el vector de contexto c y todas las palabras predichas previamente $y_1, y_2, \ldots, y_{t-1}$ al decodificador, el proceso de decodificación predice la siguiente palabra y_t. El decodificador define una probabilidad sobre la salida y descomponiendo la probabilidad conjunta de la siguiente manera:

$$p(y) = \prod_{t=1}^{T} p(y_t | \{y_1, y_2, \ldots, y_{t-1}\}, c)$$
$$p(y_t | \{y_1, y_2, \ldots, y_{t-1}\}, c) = \tanh(y_{t-1}, c)$$

donde $y = (y_1, y_2, \ldots, y_T)$ y h_t es la unidad oculta.

La siguiente figura 3.4 muestra el proceso de codificación y decodificación para una secuencia de entrada x_1, x_2, x_3. La función Softmax se utiliza para normalizar la distribución de probabilidad para la salida y para calcular la pérdida de error [17].

Estas redes han demostrado la capacidad de abordar el desafío de la longitud variable de entrada y salida.

De acuerdo al trabajo de Schmaltz, Kim, Rush y Shieber [53], los modelos secuencia a secuencia para la tarea de corrección de oraciones, pueden superar significativamente un sistema SMT de última generación sin aumentar el modelo con un modelo secundario para manejar palabras de baja frecuencia [54] o un modelo adicional para mejorar la precisión o con la intersección de un modelo de lenguaje grande [55].

En [56], los autores mencionan que la mayoría de los modelos de secuencia a secuencia para la corrección gramatical de errores (GEC) tienen dos limitaciones: (1) los modelos seq2seq están entrenados con solo pares de oraciones con corrección de errores limitada y limitados por el tamaño de los datos de entrenamiento, los modelos con millones de parámetros pueden no estar bien generalizados, (2) un modelo seq2seq puede fallar al corregir completamente una oración con múltiples errores a través de la inferencia normal seq2seq, porque algunos errores en una

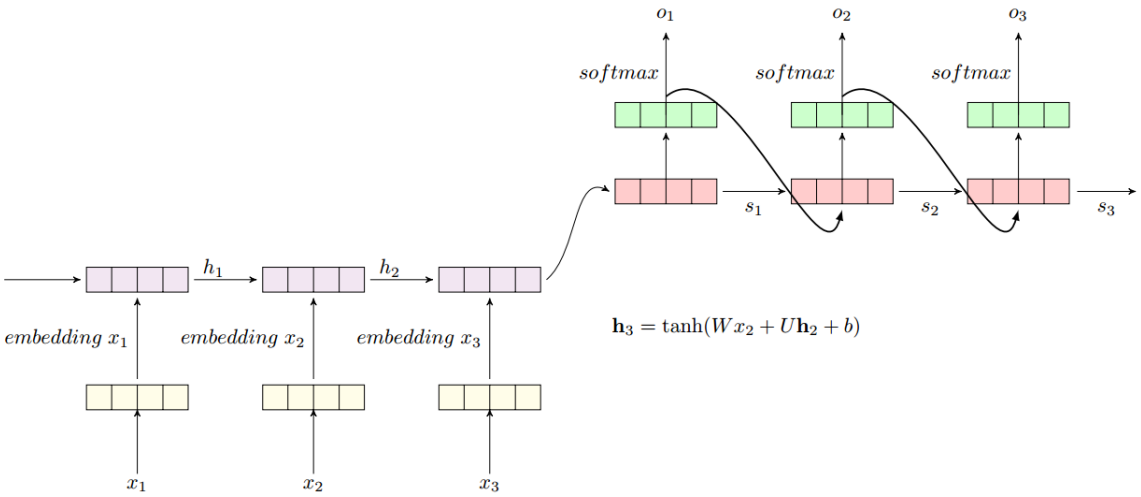


Figura 3.4: Modelo seq2seq para tres pasos de tiempo, [17].

oración pueden hacer que el contexto sea extraño, lo que confunde a los modelos para corregir otros errores.

Para resolver estos inconvenientes los autores plantean un nuevo mecanismo de inferencia y un aprendizaje llamado `fluency boost`, utilizando el concepto de fluidez. En el artículo, se define la fluidez como la probabilidad de que la oración sea escrita por un hablante nativo. Este modelo no solo se entrena un modelo secuencia a secuencia con pares de oraciones corregidas con errores originales, sino que también genera oraciones menos fluidas para establecer nuevos pares de oraciones con corrección de errores al emparejarlas con sus oraciones correctas durante el entrenamiento, siempre que la fluidez de las oraciones sea inferior a la de sus oraciones correctas.

Una dificultad más al enfoque secuencia a secuencia es que el estado oculto tiende a reflejar la información más reciente, perdiendo memoria del contenido anterior. Esto fuerza una limitación para secuencias largas, donde toda la información sobre esa secuencia debe resumirse en una sola codificación, lo que conlleva a utilizar mecanismos de atención.

En adelante trabajo nos refererimos a los modelos secuencia a secuencia como seq2seq.

3.6.1. Sampling

Un modelo seq2seq aprende a predecir las siguientes palabras o caracteres a los que se les da alguna entrada. Sin embargo, lo que se aprende de la salida del modelo es una distribución de probabilidad en lugar de una palabra. Al generar texto, elegimos solo una de las palabras dadas las probabilidades y las retroalimentamos a la red.

Esto se llama sampling y existen algunos tipos que describimos a continuación:

- Búsqueda greedy: Es la forma más simple de sampling. En cada punto, se elegirá la siguiente palabra o caracter más probable, solo uno y es susceptible a cometer problemas de repetición.
- Beam Search: Este método en lugar de seleccionar la mejor palabra siguiente, selecciona las

siguientes k palabras. k se llama el ancho del beam y puede generar múltiples resultados, y las seleccionados son a menudo mejores que los puntos creados mediante la búsqueda greedy.

- Búsqueda aleatoria: funciona seleccionando la siguiente palabra con las probabilidades de cada palabra tomada de la salida del modelo. Este método proporciona resultados que se ven aleatorios. Para corregir esta aleatoriedad, se aumenta la probabilidad de las palabras más probables y se disminuye las probabilidades de las menos probables usando la temperatura [2]. Cuanto más alta es la temperatura, más resultados diversos da y viceversa.

3.7. Mecanismo de Atención

La atención ha sido una de las técnicas más populares para abordar este último problema de dependencias largas y el uso eficiente de la memoria para el cálculo. Según [57], el mecanismo de atención interviene como una capa intermedia entre el codificador y el decodificador, con el objetivo de capturar la información de la secuencia de palabras que son relevantes para el contenido de la oración.

A diferencia del modelo seq2seq que usa el mismo vector de contexto para cada estado oculto del decodificador, el mecanismo de atención calcula un nuevo vector c_t para la palabra de salida y_t en el paso de decodificación t. Esto se puede definir como:

$$c_t = \sum_{j=1}^{T} a_{tj} h_j$$

donde h_j es el estado oculto de la palabra x_j y a_{tj} es el peso de h_j para predecir y_t. Este vector es llamado `vector de atención`, que se calcula con la función softmax:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})}$$
$$e_{ij} = a(s_{i-1}, h_j)$$

donde a es un modelo de alineación que califica qué tan bien coinciden las entradas en la posición j y la salida en la posición i. La puntuación se basa en el estado oculto RNN s_{i-1} y la anotación j-ésima anotación h_j de la oración de entrada.

En el trabajo original [57], el modelo de alineación está parametrizado como una red neuronal de alimentación directa, que se entrena conjuntamente con todos los demás componentes del sistema propuesto. Este vector se normaliza en el vector de atención real mediante el uso de una función softmax sobre las puntuaciones:

$$\alpha_t = \text{softmax}(a)_t$$

[2] Temperature

Este vector de atención se usa para ponderar la representación codificada del estado oculto h_j.

Este mecanismo descrito anteriormente se muestra en la figura 3.5.

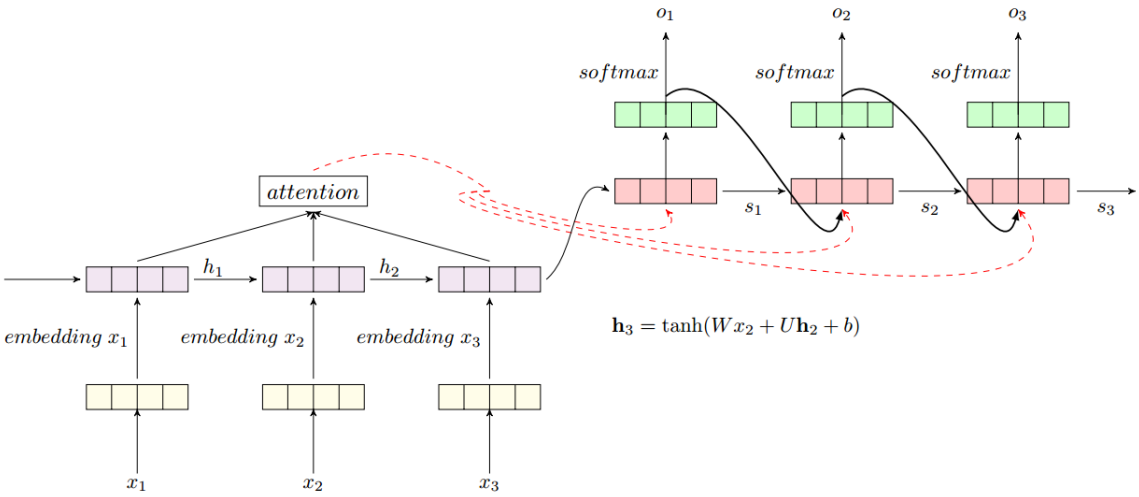


Figura 3.5: Modelo seq2seq con atención para tres pasos de tiempo, [17].

Un grupo de mecanismos de atención repite el cálculo de un vector de atención entre la consulta y el contexto a través de múltiples capas. Esto se conoce como `multi-hop`.

En [58] argumenta que el mecanismo de atención implementado por Bahdanau puede verse como una forma de memoria. Extiende el mecanismo de atención a una configuración de `multi-hop`. Eso significa que la red lee la misma secuencia de entrada varias veces antes de producir una salida y actualiza el contenido de la memoria en cada paso. Otra modificación es que el modelo funciona con múltiples oraciones de origen en lugar de una sola.

Existen otros enfoques de mecanismos de atención, dados en el artículo de Thang, Pam y Manning [59], donde se menciona la diferencia entre atención global y local.

La idea de una atención global es utilizar todos los estados ocultos del codificador al calcular cada vector de contexto, siendo computacionalmente costoso, ya que se debe atender a todas las palabras en el lado de origen para cada palabra objetivo. Para superar esto, se utiliza la atención local que primero elige una posición en la oración fuente. Esta posición determinará una ventana de palabras a las que atiende el modelo.

Los autores también experimentaron con diferentes funciones de alineación y simplificaron la ruta de cálculo en comparación con el trabajo original.

En este trabajo usaremos el mecanismo de atención global, por el número de datos disponibles.

3.8. Métricas de evaluación

Esta sección presenta métodos de evaluación que se utilizan en los modelos.

Aunque se pueden usar varias métricas para evaluar un sistema de traducción automática, en

el caso de la corrección ortográfica automática, la evaluación puede limitarse a una clasificación binaria de predicciones donde los elementos coincidentes se consideran predicciones verdaderas y los otros, predicciones incorrectas.

3.8.0.1. Exactitud

En las tareas de clasificación, la exactitud [3] es una de las medidas de rendimiento más utilizadas. Esta métrica corresponde a la relación de entradas clasificadas correctamente con respecto al número total de entradas.

Formalmente, la exactitud se puede definir como:

$$\text{exactitud} = \frac{VP + VN}{VP + VN + FP + FN}$$

donde VP=Verdaderos positivos,VN=Verdaderos negativos,FP=Falsos positivos y FN=Falsos negativos.

3.8.0.2. BLEU

BLEU [74] es una métrica que sirve para comparar las correcciones de texto candidato generado por un modelo seq2seq en archivos de texto con respecto a una o más correciones de referencia. El puntaje BLEU consta de dos partes, precisión modificada y la penalidad de brevedad.

Formalmente se define BLEU como :

$$\texttt{BLEU} = \rho \left(\prod_{i=1}^{N} \texttt{precision}_i \right)^{\frac{1}{N}}$$

donde ρ es la penalidad de brevedad [4], donde n es el orden de n-gramas que suele ser 4, y la $\texttt{precision}_i$ y ρ se calculan de la siguiente manera:

$$\text{precision}_i = \frac{\sum_{t_i} \text{mín}\{C_h(t_i), \text{máx}_j\, C_{h_j}(t_i)\}}{H(i)}$$

en la ecuación anterior $H(i)$ es el número de tuplas de i-gramas en los candidatos, t_i es una tupla i-gramas en el candidato h, $C_h(t_i)$ es el número de veces que ocurre t_i en el candidato y $C_{h_j}(t_i)$ es el número de veces que t_i ocurre en la referencia j de este candidato.

$$\rho = \exp\{\text{mín}(0, \frac{n - L}{N})\}$$

aquí n se refiere a la longitud de la hipótesis y L se refiere a la longitud de la referencia dada.

[3] accuracy
[4] brevity penalty

La tarea principal de BLEU es comparar n-gramas del candidato con los n-gramas de la referencia y contar el número de coincidencias. Estas coincidencias son independientes de la posición. Cuantas más coincidencias, mejor será la corrección propuesta del candidato. Los puntajes de BLEU varían de 0 a 100.

El puntaje es para comparar oraciones, pero también se propone una versión modificada que normaliza los n-gramas por su aparición para mejorar la puntuación de bloques de oraciones.

3.8.0.3. CharacTER

CharacTER [75] es una métrica a nivel caracter de otra métrica llamada TER [92] y se define como el número mínimo de ediciones de caracteres requeridas para ajustar una hipótesis, de modo que coincida absolutamente con la referencia, normalizada por la longitud de la oración de la hipótesis, como la indica la siguiente ecuación:

$$\texttt{CharacTER} = \frac{\text{distancia edicion} + \text{costo shift}}{\#\text{caracteres en la oración de hipótesis}}$$

Donde la longitud de palabra promedio de la frase desplazada se define como el `costo shift`, ya que los costos de inserciones, eliminaciones y sustituciones se hacen mucho mayores a nivel de caracteres a diferencia del cálculo de TER, donde el desplazamiento de una frase completa tiene un costo de 1, no importa cuán lejos se mueva esta frase [75].

Los puntajes de CharacTER varían de 0 a 1. Sin embargo, a diferencia de los resultados con otras métricas, con CharacTER, un puntaje más alto es una señal de mayor esfuerzo post-edición y por lo tanto, un puntaje bajo es mejor, ya que esto indica que se requiere menos post-edición.

Revisión del Estado del Arte

Para la revisión del estado del arte se identificaron las bases de datos ACL [1] y arXiv[2] y se realizaron dos importantes grupos de búsqueda. En la primera búsqueda orientada a la corrección ortográfica, se utilizaron términos como: `text correction`, `automatic spelling correction`, `deep learning`, `seq2seq`, `sequence-to-sequence`, `attention mechanism`, `resource scarse language` y `low-Resource languages`.

Y como un sistema de corrección de errores gramaticales (GEC) tiene como tarea de corregir distintos tipos de errores en texto, como los ortográficos, de puntuación, gramaticales y de elección de palabras, realizamos el mismo procedimiento orientado a la corrección ortográfica gramatical y en la búsqueda utilizamos los siguientes términos como: `grammatical error correction`, `low-resource languages`, `deep learning`, `sequence-to-sequence`.

Se seleccionaron los siguientes artículos que se consideraron relevantes con respecto al problema a resolver en este trabajo.

4.1. Corrección ortográfica usando aprendizaje profundo

Ghosh y Kristensson [60] propusieron un modelo de aprendizaje profundo para la corrección y completación de texto en la decodificación de teclado para inglés. La importancia de mencionar este trabajo, radica en que es un primer intento de corrección de texto usando aprendizaje profundo con resultados prometedores.

El enfoque en este trabajo es abordar el problema de corrección y completación en una red correctora separada como codificador y y un modelo de lenguaje implícito como decodificador en una arquitectura secuencia a secuencia que se entrena de extremo a extremo y un mecanismo de atención, CCED (Correction and Completion Encoder Decoder Attention network).

Además los autores proponen una medida de evaluación alternativa más uniforme sobre la tasa de error de caracteres (CER)[3] para evaluar las predicciones del modelo basadas en la intención subyacente de la oración y aprovechar la información contextual mientras se evalúa el rendimiento del decodificador.

Se presenta en [61] un modelo de reconocimiento de palabras (por ejemplo un corrector ortográfico) basado en una red neuronal recurrente de nivel de semi-carácter (scRNN), que está inspirado en el robusto mecanismo de reconocimiento de palabras conocido en la literatura de la psicolingüística como el efecto Cmabrigde Uinervtisy.

Los autores demuestran que el scRNN tiene un rendimiento significativamente mejor en la

[1] https://www.aclweb.org/portal/
[2] https://arxiv.org/
[3] The Character Error Rate

corrección ortográfica de palabras, en comparación con los correctores ortográficos existentes y una red neuronal convolucional basada en caracteres: charCNN.

Además, se indica según la revisión de la literatura en psicolingüística, la posición de los caracteres mezclados afecta la carga cognitiva del reconocimiento de palabras humanas, se experimenta el modelo scRNN mediante la manipulación de la estructura del vector de entrada. Se replica el paradigma del movimiento ocular de Rayner, pero utilizando scRNN en lugar de sujetos humanos.

El resultado es que el scRNN replica (al menos una parte) el mecanismo de reconocimiento de palabras humanas.

Etoori, Chinnakotl y Radhika Mamidi [62], muestran cómo construir un corrector ortográfico automático para lenguas de escasos recursos. Ellos proponen un modelo de aprendizaje profundo seq2seq basados en caracteres para lenguas índicos (SCMIL). De acuerdo al artículo ellos proponen lo siguiente:

- Una arquitectura de seq2seq recurrente basada en caracteres con un codificador LSTM y un decodificador LSTM para la corrección ortográfica de las lenguas índicas.
- Creación de conjuntos de datos sintéticos para Hindi y Telugu mediante la recopilación de errores ortográficos muy probables e induciendo ruido en corpus limpios.
- Comparación del rendimiento de SCMIL con otros enfoques como la traducción automática estadística (SMT), métodos basados en reglas y varios modelos de aprendizaje profundo.

4.2. Corrección de errores gramaticales usando aprendizaje profundo

Los autores en [63] consideran el GEC como un problema de traducción automática neural de bajos recursos o como máximo de recursos medios. Para esto se adaptan varios métodos de MT de bajo recurso a una GEC neuronal y se establecen pautas para obtener resultados confiables. Los métodos propuestos incluyen, técnicas de adaptación de dominio, transfer learning con datos monolingües. Además en el trabajo se propone un conjunto de métodos independientes que se pueden aplicar fácilmente en la mayoría de los entornos de GEC, entornos GEC neuronales más eficientes.

Uno de los aspectos más importantes es que el NMT de bajos recursos se basan en el aprendizaje de transferencia, ya que los enfoques basados en entrenamiento previo con datos monolingües parecen ser particularmente adecuados para la tarea GEC. Para obtener los parámetros del decodificador de pre-entrenamiento se entrena un modelo de lenguaje basado en GRU en los datos monolingües. La arquitectura del modelo de lenguaje corresponde lo más posible a la estructura del decodificador de un modelo seq2seq.

En general, el modelado de lenguaje requiere que se entrene el modelo para una tarea específica que se está tratando de resolver. Entonces, si se desea traducir o corregir texto del inglés al francés, se debe entrenar el modelo con muchos ejemplos de texto en francés e inglés. Si se quiere entrenar el modelo para el alemán, se debe hacer lo mismo y así sucesivamente. Obviamente, esto requiere muchos datos, ya que los necesita para cada lenguaje para la tarea que está tratando de resolver.

El aprendizaje zero-shot entrena un modelo universal en un conjunto de datos muy grande o en un conjunto de datos muy variado. Luego se puede aplicar este modelo a cualquier tarea. En el ejemplo de traducción o corrección de errores gramaticales, se debe entrenar un modelo y lo usaría como una especie de traductor o corrector gramatical universal para otros lenguajes. El siguiente artículo [64] hizo exactamente esto y pudo aprender representaciones de oraciones para 93 lenguajes diferentes.

El trabajo titulado Grammatical Error Correction and Style Transfer via Zero-shot Monolingual Translation [66] los autores, presentan un enfoque para realizar GEC y Transferencia de Estilo con el mismo modelo entrenado, sin utilizar ningún dato de entrenamiento supervisado para ninguna de las tareas. Se basan en la traducción automática neuronal (NMT) de zeroshot [65], y como tal, el único tipo de datos que utiliza son corpus paralelos regulares (con textos y sus traducciones). De acuerdo al artículo las contribuciones para ambas tareas son los siguiente:

- Un método único para la Transferencia de Estilo y GEC, sin utilizar datos anotados para ninguna de las tareas.
- Soporte para ambas tareas en varios lenguajes dentro del mismo modelo. Los autores utilizan en sus experimentos, inglés, estonio y letón.
- Un análisis cuantitativo exhaustivo y evaluación manual cualitativa del modelo en ambas tareas.
- Resaltar los aspectos de confiabilidad del modelo en ambas tareas.

En [67] se presenta un nuevo conjunto de datos para el GEC en checo. Este conjunto de datos AKCES-GEC, un proyecto general que comprende varios recursos: CzeSL (corpus de estudiantes de checo como segundo lenguaje), ROMi (etnolecto romaní y adolescentes checos romaníes) y SKRIPT y SCHOLA (lenguaje escrito y hablado recogido de alumnos checos nativos, respectivamente). Se realizan experimentos en lenguas de bajos recursos como el alemán, ruso y checo. Para cada lenguaje se utiliza el modelo Transformer en datos sintéticos y se ajusta con una mezcla de datos sintéticos y auténticos, obteniendo resultados de vanguardia. El objetivo del artículo es concentrarse en lenguajes de bajos recursos en lugar del inglés.

Los autores de [68], partiendo de que el GEC se puede ver como un modelo seq2seq de bajos recursos, los autores generan versiones erróneas de corpus grandes sin anotaciones utilizando una función de ruido reales. Los cuerpos paralelos resultantes se usan para pre-entrenar los modelos Transformer y luego aplicando el Aprendizaje de Transferencia adaptado a un dominio y estilo de un conjunto de prueba y un corrector ortográfico neuronal, se logra resultados competitivos en escenarios de bajos recursos en `ACL 2019 BEAShared Task`.

Un aspecto importante que muestra el artículo es que antes de introducir texto con corrección ortográfica en un modelo seq2seq, se aplica BPE [69] [4] usando SentencePiece [70]. Primero se entrena un modelo SentencePiece con un tamaño de vocabulario de $32K$ en el corpus Gutenberg original, y se aplica este modelo a todo el texto de entrada. Esto permite evitar tokens `<unk>` en la mayoría de los conjuntos de entrenamiento y validación, incluido el conjunto de desarrollo $W\&I + L$.

[4] byte-pair encoding

Experimentación y Resultados

En el presente capítulo se describen los resultados obtenidos luego de cumplir con los objetivos propuestos.

5.1. Procesamiento de datos

En esta sección se desarrolla el resultado esperado para el objetivo 1. Se realiza el procesamiento de los datos recolectados.

Dentro del ámbito de las lenguas originarias peruanas, existen un grupo de páginas con información a analizar de documentos en formato PDF, como es el caso del Ministerio de Cultura, la Oficina de Medición de la Calidad del Aprendizaje (UMC), los voluntarios del SIL Internacional, conocido en el Perú como el Instituto Lingüístico de Verano (ILV), o el portal PeruEduca.

El Sistema Digital para el Aprendizaje PerúEduca permite a los profesores, directivos, alumnos y padres de familia acceder a herramientas, servicios y recursos educativos a través de internet. Dentro de este portal existe información de cada una de las lenguas a estudiar en este trabajo. Por ejemplo podemos encontrar guías de alfabetos, relatos de tradiciones, manuales de escritura y herramientas de trabajo que permiten a docentes del EIB conocer más de su lengua originaria o contar con orientaciones para el uso del alfabeto oficial proporcionado por el Ministerio de Educación. Para esta tesis utilizaremos los documentos que se encuentran en el portal de PeruEduca.

Con esta información se realizó un web scraper para encontrar documentos de esa página. Todos los documentos utilizados para formar un corpus de trabajo fueron descargados en formato PDF y se utilizaron herramientas como PDFMiner, para extraer información en texto simple.

Se utiliza PyMongo para crear un MongoClient para la instancia `mongod` en ejecución y se utiliza una colección [1] que es lo equivalente de una tabla en una base de datos relacional, de los documentos almacenados en MongoDB con estilo de acceso a diccionarios. En este proceso se guardaron todos los documentos desde el portal PeruEduca en formato PDF identificados con un link, asociado a cada de una de las lenguas objetivo.

Se escribe una función que devuelve todos los documentos de cada lengua utilizando un identificador, un url para cada lengua seleccionada que se encuentra dentro de la base de datos, así como el número de documentos obtenidos por cada lengua.

El script pdf2txt.py [2] extrae todo el texto de un PDF y utilizamos valores predeterminados razonables para ordenar y agrupar el texto para cada una de las lenguas a tratar.

Utilizamos un filtrado de oraciones basados en reglas utilizando expresiones regulares.

[1] collections
[2] https://pdfminersix.readthedocs.io/en/latest/api/commandline.html#api-pdf2txt

Un resumen obtenidos después del procesamiento de datos se presenta en la siguiente tabla:

	Lenguaje	Numero archivos	Numero oraciones	Numero_tokens	abecedario	iso-639-3
0	Yanesha	26	13242	23626	28	yanesha
1	Yine	11	7659	14142	21	yine
2	Ashaninka	33	12630	23721	19	ashaninka
3	Shipibo Konibo	32	22033	22944	19	shipibo-konibo

Figura 5.1: Resumen de los datos extraídos desde PerúEduca.

A partir de la información obtenida podemos obtener las distribuciones de palabras y oraciones por caracteres para cada lengua:

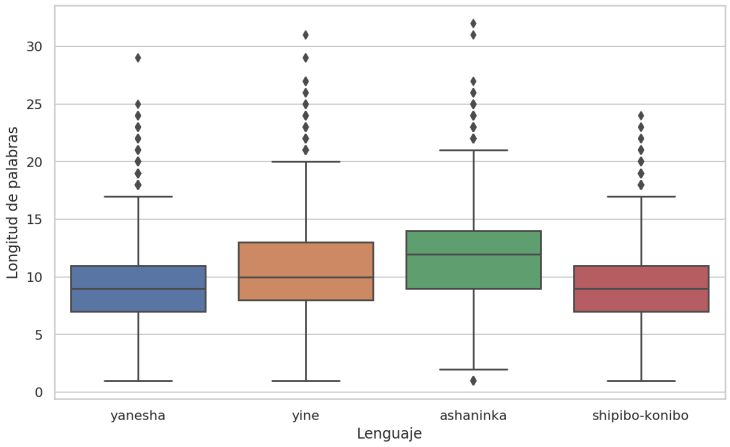


Figura 5.2: Distribución de longitud de palabras en número de caracteres para cada lengua estudiada.

En la figura 5.2, observamos que la longitud promedio de un token está en un rango de 7 a 13 caracteres y hay tokens con incluso más de 30 caracteres.

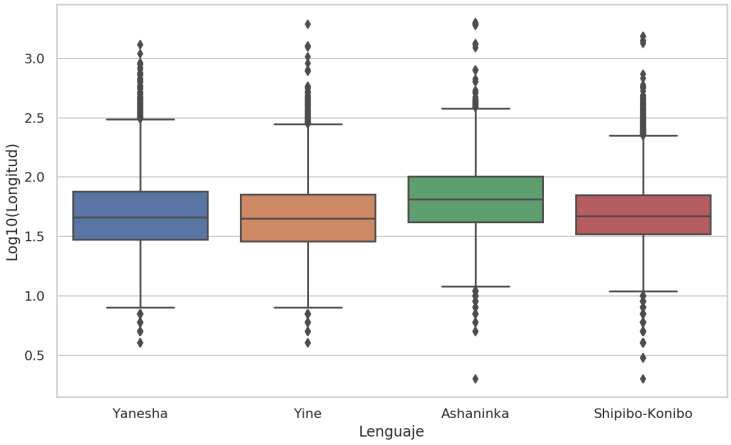


Figura 5.3: Distribución de longitud de oraciones en número de caracteres para cada lengua estudiada.

La figura 5.3 muestra la distribución de las longitudes de oraciones en caracteres con escala logarítmica donde el valor promedio de longitud para todos las lenguas relacionadas está entre 1,70 y 1,75, siendo la lengua Ashaninka la que tiene un valor ligeramente más alto. La diferencia podría ser la mayor longitud de palabra promedio en Ashaninka en contraste con las otras tres lenguas, de acuerdo al gráfico anterior.

El trabajo de Bustamante, Oncevay y Zariquiey [78] mejora obstensiblemente, el corpus mediante la identificación de los límites de las oraciones utilizando un tokenizador de oraciones no supervisado (NLTK Punkt Sentence Tokenize [3]) y el diseño de expresiones regulares en un enfoque basado en reglas: por ejemplo una oración está contenida entre una letra mayúscula y un punto, los títulos de las secciones se encuentran dentro de una letra mayúscula y dos saltos de línea, entre otros.

Asimismo, los autores realizan una selección y filtrado de oraciones basado en una herramienta automática para la identificación del lenguaje y heurísticas basada en reglas, como se muestra en las figuras 5.4 y 5.5 de aceptación (`accepted`) y rechazo (`rejected`) para palabras y oraciones.

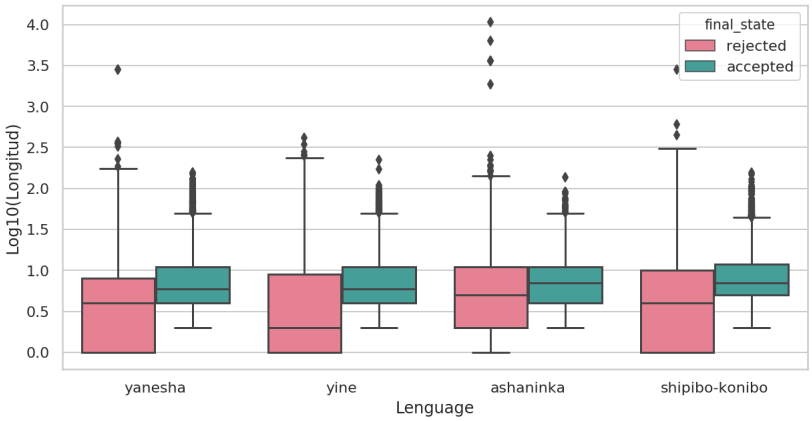


Figura 5.4: Filtrado de palabras en número de caracteres para cada lengua estudiada.

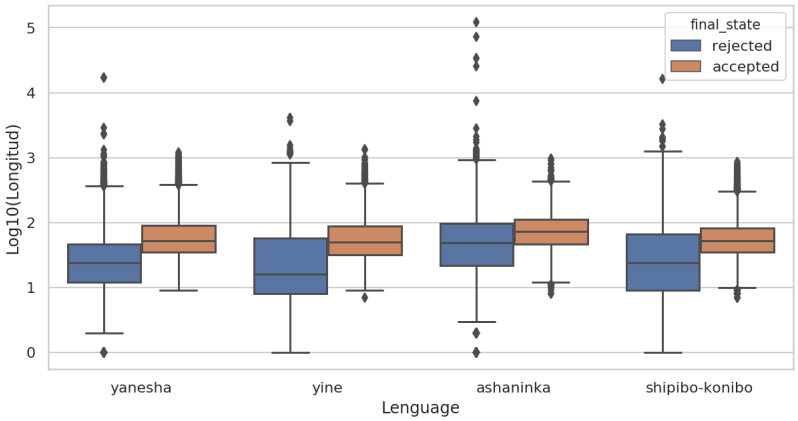


Figura 5.5: Filtrado de oraciones en número de caracteres para cada lengua estudiada.

Se realiza un análisis estadístico de los corpus procesados, utilizando el modelado LNRE [4] y comparan el proceso de filtrado presentado con un filtro aleatorio mediante el muestreo de

[3] https://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize.punkt
[4] Large number or rare events

oraciones con el mismo tamaño que el corpus final por lengua estudiada. Las oraciones extraídas automáticamente por el filtro son el conjunto de datos.

Se realizan experimentos diferentes con cada tipo de filtro empleando diferentes semillas aleatorias para realizar una comparación estadística válida y garantizar que los resultados no sean coincidentes.

5.2. Algoritmo de generación de errores

De acuerdo a [79], los errores de ortográficos pueden ocurrir de muchas formas diferentes; desde puntuación simple hasta errores que incluyen tiempos y la forma de palabras, colocaciones incorrectas, modismos erróneos y de información lingüística. Identificar automáticamente todos estos errores es una importante tarea especialmente porque la cantidad de datos anotados disponibles es muy limitada.

Con el propósito de construir un corpus paralelo de palabras a palabras con errores y oraciones a oraciones errores que es el resultado esperado para el objetivo 2, se desarrolla un algoritmo elemental de generación de errores. Este algoritmo genera los siguientes tipos de errores:

- Reemplaza un caracter con un caracter aleatorio.
- Elimina un caracter
- Agrega un caracter de forma aleatoria
- Transpone dos caracteres
- Elimina los espacios en blanco

La probabilidad de que ocurra cualquiera de los errores es la misma y se agrega un cantidad de ruido al algoritmo.

Algunos resultados obtenidos se muestra en la siguiente tabla:

	Palabra correcta	**Palabra incorrecta**
Ashaninka	asbihtashi	pishitashi
Shipibo-Konibo	koyribobixribi	koríkibobiribi
Yanesha	seyepre	seyerpa
Yine	genpkahlu	genekashlu

Cuadro 5.1: Resultados del algoritmo de generación de errores a nivel palabra.

5.3. Modelo seq2seq

5.3.1. Modelo seq2seq base

Para poder construir los modelos seq2seq, filtramos la distribución de las longitudes de las oraciones de acuerdo al criterio de dividir por cuantiles, tal como se indica en el cuadro 5.6.

	longitud_oraciones	log_longitud_oraciones
0.10	21.0	1.322219
0.25	31.0	1.491362
0.50	46.0	1.662758
0.75	74.0	1.869232

Figura 5.6: Cuantiles de la distribución de la longitud de las oraciones.

Nuestros datos de entrenamiento son pares de longitud 21, 31 y 46 de oraciones con errores ortográficos, generadas por el algoritmo propuesto de generación de errores y oraciones sin errores ortográficos.

Construiremos un modelo simple seq2seq (sin atención), como se muestra en el diagrama 5.7, que nos servirá de modelo base.

La entrada de nuestro modelo es una secuencia de caracteres. Sin embargo, necesitamos transformar los caracteres de manera que puedan ser entendidos por nuestra red neuronal. Para hacer esto, necesitamos construir un diccionario de codificación y una función inversa para decodificar los resultados, además de completar la entrada del decodificador con un símbolo `INICIO` y transformar la salida del decodificador en una codificación one-hot.

El primer paso para transformar cada caracter de entrada en un vector denso es mediante una capa embedding. Utilizamos una capa recurrente LSTM para codificar los vectores de entrada. La salida de este paso de codificación será la salida del LSTM en el paso de tiempo final.

Similar al paso de codificación, la entrada del decodificador es una secuencia de caracteres. También pasamos la entrada del decodificador a una capa de embedding para transformar cada caracter en un vector. Nuevamente usamos el LSTM tomando una entrada pasada por la capa embedding un vector a la vez. Sin embargo, en esta fase, usamos la salida del codificador como *initial_state* para el LSTM, que usará tanto la salida del codificador como la entrada para determinar la salida.

Finalmente, usamos una capa densa (time distributed) con activación softmax para transformar la salida de cada LSTM en el resultado final.

Después de entrenar el modelo, realizamos una decodificación codiciosa e implementamos una función de decodificación de caracteres, para mostrar resultados a través de archivos de textos: un archivo de texto de candidatos que devuelve la red neuronal y un archivo de referencia que es parte de nuestro conjunto de pruebas.

5.3.2. Modelo seq2seq y regularización

Construimos un nuevo modelo seq2seq apilando más capas LSTM en el codificador y utilizando técnicas de regularización. El gráfico 5.8 muestra la arquitectura de este modelo.

Aplicamos mecanismos de regularización como medida de evitar el sobreajuste al agregar una penalización a la salida de la capa con una regularización L1.

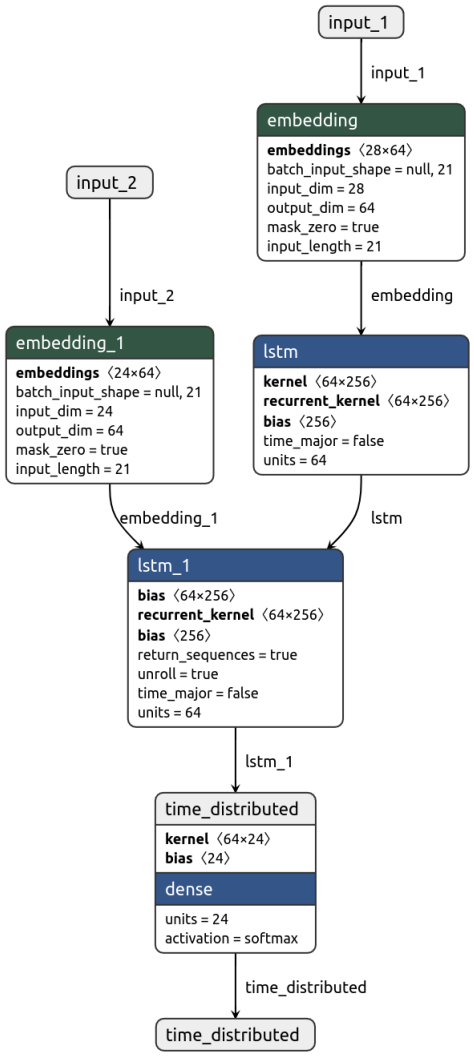


Figura 5.7: Modelo seq2seq base .

5.3.3. Modelo seq2seq con atención

El mecanismo de atención es una extensión de los modelos seq2seq que mejora su rendimiento. En efecto, una deficiencia del modelo base anterior, es que tiene que codificar, comprimir y recordar toda la secuencia de entrada en un solo vector. Esto hace que el vector se convierta en un cuello de botella de información cuando queremos generar palabras o oraciones largas.

En ese sentido, la idea del mecanismo de atención es hacer que el decodificador `mire hacia atrás` en la información del codificador en cada entrada y use esa información para tomar una decisión.

Para la implementación de este modelo utilizaremos un mecanismo de atención global propuesto en el paper de Luong, Pham y Manning [59]. La arquitectura del modelo propuesto se presenta en el gráfico 5.9.

Además de usar la salida del decodificador, también usamos la información del `contexto`. Este

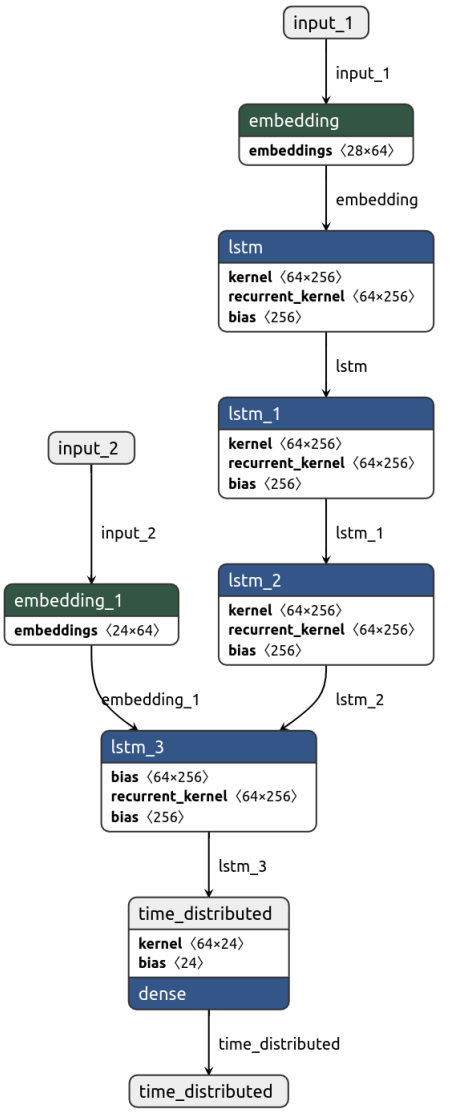


Figura 5.8: Modelo seq2seq con capas apiladas y regularización.

vector de contexto se calcula `mirando hacia atrás` en el lado del codificador y resulta de combinar cada salida del codificador con cada uno de los valores de atención.

Utilizamos la puntuación por puntos para calcular el valor de atención del modelo, que representa cuánto influye relativamente la entrada *i*-ésima en la salida *j*-ésima, es decir, cuando la red escribe la salida *j*-ésima, cuánta atención se centra en la entrada *i*-ésima en comparación con otras entradas.

Luego, podemos entrenar el modelo en pares de oraciones con errores y sin errores ortográficos. El proceso de transformación de datos y los archivos de texto generados es similar al modelo base.

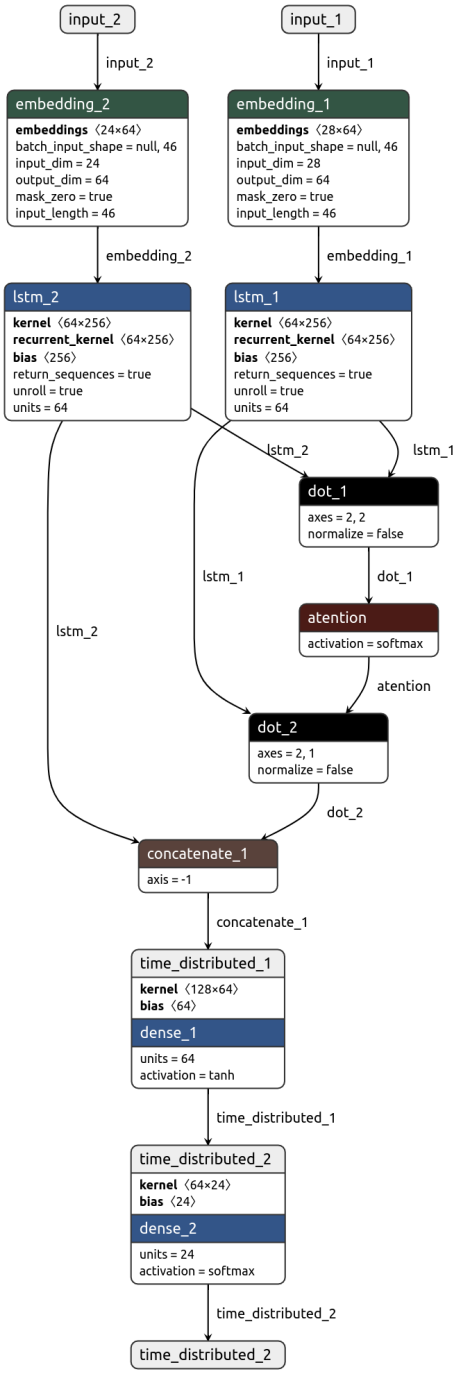


Figura 5.9: Modelo seq2seq con un mecanismo de atención.

5.4. Resultados obtenidos

En esta sección mostramos los resultados obtenidos, desde el modelo seq2seq base, el modelo seq2seq con capas LSTM apiladas y técnicas de regularización y el modelo seq2seq con un mecanismo de atención global a nivel caracter.

Para medir la predicción de los experimentos, utilizamos métricas como la exactitud, BLEU, utilizando sacreBLEU [80][5], que funciona sobre texto plano y CharacTER[6], que mide el número de ediciones (eliminación, adición y sustitución de palabras) necesarias para que una traducción automática coincida exactamente con la traducción de referencia más cercana en fluidez y semántica a nivel caracter, utilizando archivos de texto en la línea de comandos.

En las dos últimas métricas, comparamos los resultados en archivos de texto que proporciona la red neuronal y el conjunto de oraciones sin errores ortográficos que es parte de nuestro conjunto de pruebas. Todo esto se realiza en la línea de comandos.

Empecemos describiendo algunas propiedades importantes de cada una de las lenguas nativas analizadas y los resultados obtenidos utilizando las métricas mencionadas:

5.4.1. Ashaninka

La información para esta sección se han extraído del trabajo de David Payne sobre la gramática Ashaninka [81], el diccionario Asheninka-Castellano del mismo autor [82] y el trabajo de Judith Payne [83] para el aprendizaje del Asháninca.

- Se caracteriza por ser una lengua aglutinante, por lo que presenta prefijos y sufijos que se adhieren a la raíz nominal y verbal.
- La categoría verbal admite prefijos y sufijos. Fundamentalmente, los prefijos se refieren a la persona que realiza la acción. Las oraciones y los sufijos, se refieren al tiempo factual (no futuro: presente o pretérito) que se marca con el sufijo $-i$.
- La estructura sintáctica es VSO (Verbo-Sujeto-Objeto), pero se puede tomar el tipo SVO, como es el caso del Asheninka del Apuracayali.

El siguiente cuadro 5.2, muestra los resultados obtenido desde 484 oraciones con una longitud a lo más de 21 caracteres y un entrenamiento de 100 épocas.

Modelos	**CharacTER**	**BLEU**	**Accuracy**
seq2seq	0.739	≈ 0	0.7032
+ capas + regularización	0.765	≈ 0	0.6296
+ atención	0.682	≈ 0	0.7688

Cuadro 5.2: Resultados para Ashaninka de longitud a lo más 21 caracteres.

El siguiente cuadro 5.3, muestra los resultados obtenido desde 1497 oraciones con una longitud a lo más de 31 caracteres y un entrenamiento de 100 épocas.

Modelos	**CharacTER**	**BLEU**	**Accuracy**
seq2seq	0.634	0.7	0.6848
+ capas + regularización	0.732	1.9	0.6994
+ atención	0.302	5.2	0.8966

Cuadro 5.3: Resultados para Ashaninka de longitud a lo más 31 caracteres.

[5]https://github.com/mjpost/sacreBLEU/blob/master/sacrebleu.py
[6]https://github.com/rwth-i6/CharacTER/blob/master/CharacTER.py

Por último, el siguiente cuadro 5.4, muestra los resultados obtenido desde 3851 oraciones con una longitud a lo más de 46 caracteres y un entrenamiento de 100 épocas.

Modelos	CharacTER	BLEU	Accuracy
seq2seq	0.641	0.4	0.7123
+ capas + regularización	0.677	0.4	0.7294
+ atención	0.182	12.4	0.9532

Cuadro 5.4: Resultados para Ashaninka de longitud a lo más 46 caracteres.

En la siguiente figura 5.10, vemos un ejemplo del funcionamiento de la red neuronal y el mecanismo de atención para Ashaninka de la oración `Icheki Irene inchatopaye -Irene corta árboles`.

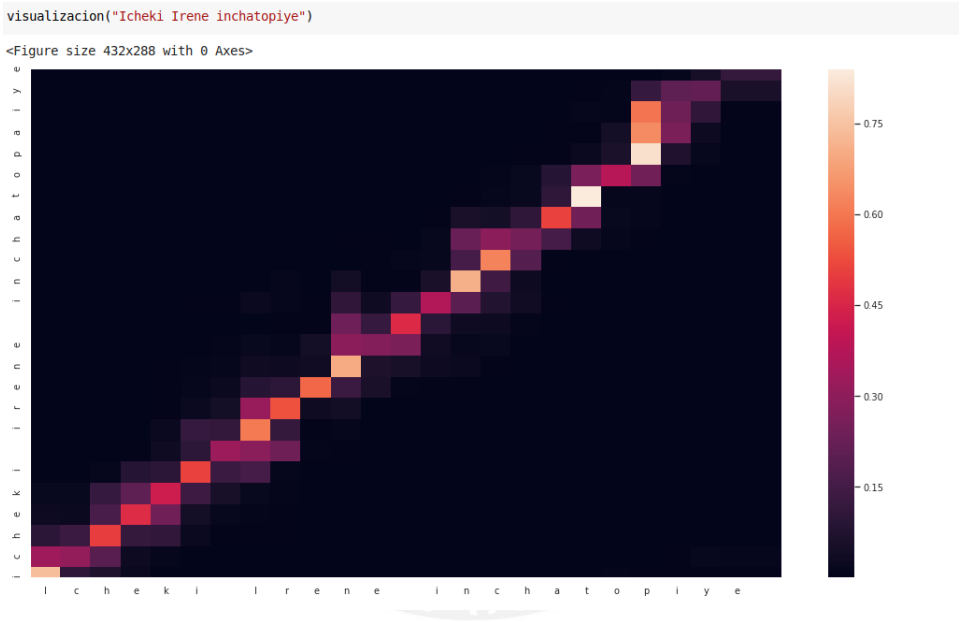


Figura 5.10: Ejemplo de la red con mecanismo de atención para Ashaninka.

Ejemplo

En esta parte mostremos los resultados obtenidos para Ashaninka con el modelo seq2seq y el modelo seq2seq con atención para la oración `nopimantiro pancotsi -- compartir casa`. Siendo el modelo con atención el que genera mejor corrección.

```
nopmiantiro pancttsi --> oshinka inka antakir
```

Figura 5.11: Salida del corrector para Ashaninka con un modelo seq2seq.

```
nopmiantiro pancttsi --> nopimantiro pantsi
```

Figura 5.12: Salida del corrector para Ashaninka con un modelo seq2seq + atención.

5.4.2. Shipibo-Konibo

La información para esta sección se han extraído del trabajo de Renzo Aguirre Santa [84], el diccionario de Shipibo [85] y los trabajos de Pilar Valenzuela [86, 87].

- Es sumamente aglutinante. Presenta afijos y clíticos y debido a esto las palabras tienden a poseer varios morfemas.
- Un rasgo sintáctico resaltante, es la presencia de un sistema de cambio de referencia (switch-reference), las cuales se unen a cláusulas subordinadas para indicar si los eventos de estas propiedades tienen lugar antes, al mismo tiempo o después del de la cláusula principal.
- Los órdenes sintácticos son AOV/SV, aunque existe flexibilidad al respecto.

El siguiente cuadro 5.5, muestra los resultados obtenido desde 1480 oraciones con una longitud a lo más de 21 caracteres y un entrenamiento de 100 épocas.

Modelos	**CharacTER**	**BLEU**	**Accuracy**
seq2seq	0.653	0.7	0.7032
+ capas + regularización	0.658	0.8	0.7239
+ atención	0.405	1.4	0.8597

Cuadro 5.5: Resultados para Shipibo-Konibo de longitud a lo más 21 caracteres.

El siguiente cuadro 5.6, muestra los resultados obtenido desde 4986 oraciones con una longitud a lo más de 31 caracteres y un entrenamiento de 100 épocas.

Modelos	**CharacTER**	**BLEU**	**Accuracy**
seq2seq	0.632	0.3	0.6961
+ capas + regularización	0.756	0.4	0.7594
+ atención	0.241	11.2	0.9299

Cuadro 5.6: Resultados para Shipibo-Konibo de longitud a lo más 31 caracteres.

El siguiente cuadro 5.7, muestra los resultados obtenido desde 11010 oraciones con una longitud a lo más de 46 caracteres y un entrenamiento de 100 épocas.

Modelos	**CharacTER**	**BLEU**	**Accuracy**
seq2seq	0.636	0.3	0.6695
+ capas + regularización	0.718	0.4	0.6763
+ atención	0.193	18.5	0.9429

Cuadro 5.7: Resultados para Shipibo-Konibo de longitud a lo más 46 caracteres.

En la siguiente figura 5.13, vemos un ejemplo del funcionamiento para Shipibo-Konibo de la palabra `chabi -llave`.

Figura 5.13: Ejemplo de la red con mecanismo de atención para Shipibo-Konibo.

Ejemplo

En esta parte mostremos los resultados obtenidos en Shipibo-Konibo con el modelo seq2seq y el modelo seq2seq con atención para la oración `ainbo metsa --mujer hermosa`. Siendo el modelo con atención el que genera una mejor corrección.

```
ainbo mttsa --> jati janeti
```

Figura 5.14: Salida del corrector para Shipibo-Konibo con un modelo seq2seq.

```
ainbo mttsa --> ainbo metsa
```

Figura 5.15: Salida del corrector para Shipibo-Konibo con un modelo seq2seq + atención.

5.4.3. Yanesha

La información para esta sección se han extraído del trabajo de Anna Luis Daigneault [88] y el texto de gramática de la lengua Yanesha de Martha Duff-Trip [89].

- Presenta un número elevado de afijos que se adhieren a una raíz, en comparación a otras lenguas aglutinantes.
- Las palabras a menudo se componen de muchos elementos morfológicos y léxicos diferentes, es decir es polisintética.
- Se caracteriza por presentar la estructura sintáctica VSO.
- Las palabras a menudo se pueden modificar para que sean específicas del contexto.

El siguiente cuadro 5.8, muestra los resultados obtenido desde 1646 oraciones con una longitud a lo más de 21 caracteres y un entrenamiento de 100 épocas.

Modelos	CharacTER	BLEU	Accuracy
seq2seq	0.604	1.5	0.7475
+ capas + regularización	0.605	2.3	0.7539
+ atención	0.520	5.2	0.8030

Cuadro 5.8: Resultados para Yanesha de longitud a lo más 21 caracteres.

El siguiente cuadro 5.9, muestra los resultados obtenido desde 3682 oraciones con una longitud a lo más de 31 caracteres y un entrenamiento de 100 épocas.

Modelos	CharacTER	BLEU	Accuracy
seq2seq	0.628	0.4	0.6798
+ capas + regularización	0.644	0.4	0.6810
+ atención	0.299	5.4	0.9209

Cuadro 5.9: Resultados para Yanesha de longitud a lo más 31 caracteres.

El siguiente cuadro 5.10, muestra los resultados obtenido desde 5682 oraciones con una longitud a lo más de 46 caracteres y un entrenamiento de 100 épocas.

Modelos	CharacTER	BLEU	Accuracy
seq2seq	0.649	0.4	0.6848
+ capas+ regularización	0.637	0.3	0.6889
+ atención	0.202	12.4	0.9276

Cuadro 5.10: Resultados para Yanesha de longitud a lo más 46 caracteres.

En la figura 5.16 vemos un ejemplo del funcionamiento para Yanesha de la oración `él/ella descansa - amesen .`

Ejemplo

En esta parte mostremos los resultados obtenidos en Yanesha con el modelo seq2seq y el modelo seq2seq con atención para la oración `mecharr--miedoso`. Siendo el modelo con atención quien genera una mejor corrección.

5.4.4. Yine

La información para esta sección se han extraído del trabajo de Rebecca Hanson [90] y el diccionario Piro(Yine) [91].

- Es polisintética, aglutinante y casi totalmente con sufijos con solo un grupo de prefijos.
- Una oración está compuesta por un verbo transitivo y frases nominales. Una oración transitiva está compuesta por un verbo transitivo y sus dos argumentos (sujeto y complemento).

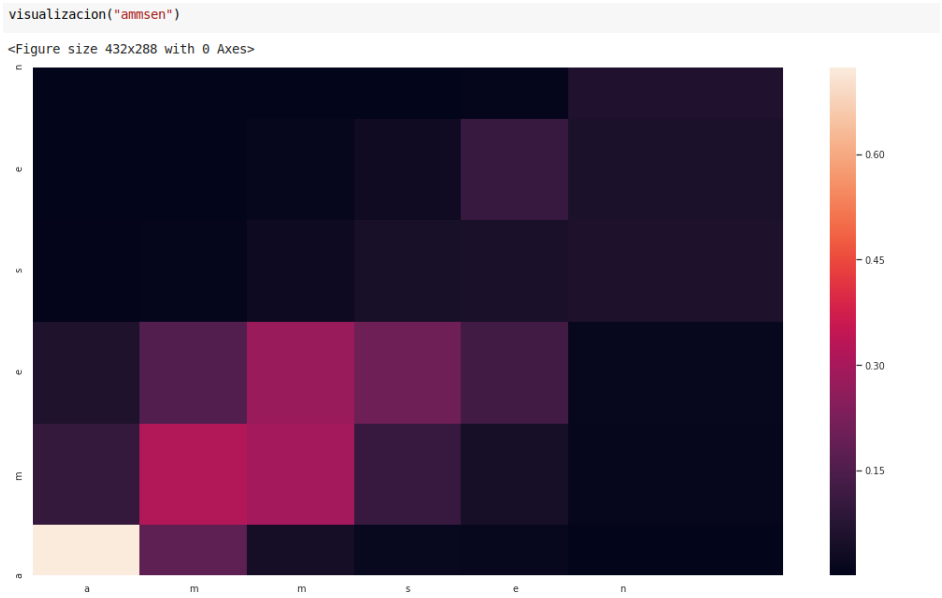


Figura 5.16: Ejemplo de la red con mecanismo de atención para Yanesha.

Figura 5.17: Salida del corrector para Yanesha con un modelo seq2seq.

Figura 5.18: Salida del corrector para Yanesha con un modelo seq2seq + atención.

- Las oraciones intransitivas están conformadas por un verbo intransitivo y un único argumento (sujeto).
- El orden de las palabras de una oración no es rígido, pero el orden sintáctico más común es SOV.

El siguiente cuadro 5.11, muestra los resultados obtenido desde 802 oraciones con una longitud a lo más de 21 caracteres y un entrenamiento de 100 épocas.

Modelos	CharacTER	BLEU	Accuracy
seq2seq	0.667	2.6	0.6467
+ capas+ regularización	0.752	2.7	0.6729
+ atención	0.345	4.6	0.8702

Cuadro 5.11: Resultados para Yine de longitud a lo más 21 caracteres

El siguiente cuadro 5.12, muestra los resultados obtenido desde 2227 oraciones con una longitud a lo más de 31 caracteres y un entrenamiento de 100 épocas.

Modelos	CharacTER	BLEU	Accuracy
seq2seq	0.681	1.2	0.6906
+ capas+ regularización	0.647	1.3	0.703
+ atención	0.299	8.4	0.9139

Cuadro 5.12: Resultados para Yine de longitud a lo más 31 caracteres

El siguiente cuadro 5.13, muestra los resultados obtenido desde 4025 oraciones con una longitud a lo más de 46 caracteres y un entrenamiento de 100 épocas.

Modelos	CharacTER	BLEU	Accuracy
seq2seq	0.667	1.3	0.7077
+capas + regularización	0.686	1.4	0.7125
+atención	0.202	15.2	0.9606

Cuadro 5.13: Resultados para Yine de longitud a lo más 46 caracteres

En la siguiente figura 5.19, vemos un ejemplo del funcionamiento para Yine de la palabra `kiruka-libro` .

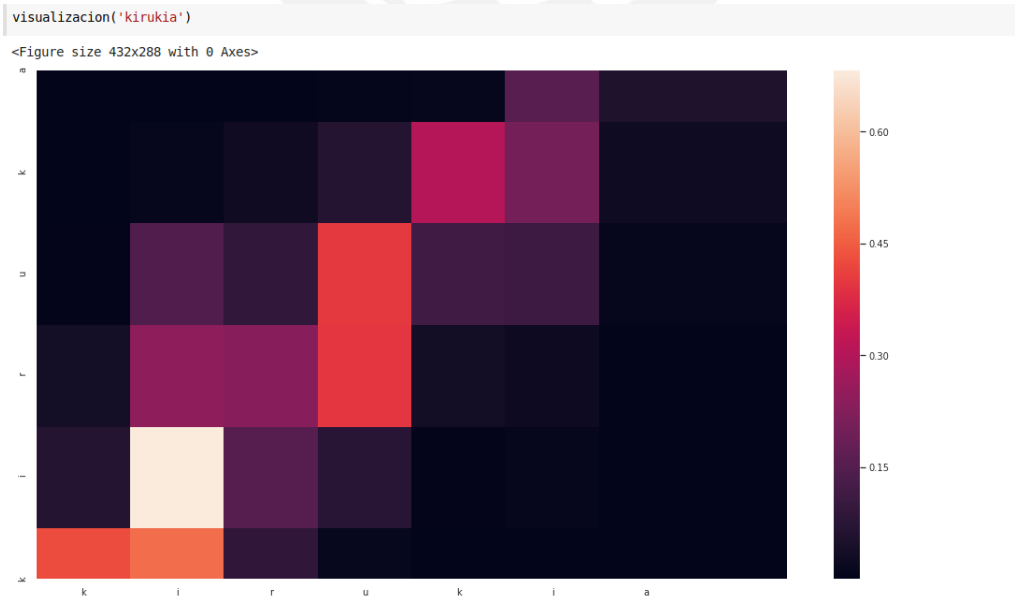


Figura 5.19: Ejemplo de la red con mecanismo de atención en Yine.

Ejemplo

En esta sección mostremos los resultados obtenidos en Yine con el modelo seq2seq y el modelo seq2seq con atención para la oración `yompompota--camina cabizbajo`. Siendo el modelo con atención quien genera una mejor corrección.

```
yampompata --> katu gimaka
```

Figura 5.20: Salida del corrector para Yine con un modelo seq2seq.

```
yampompata --> yampompota
```

Figura 5.21: Salida del corrector para Yine con un modelo seq2seq + atención.

5.5. Discusión de resultados

- La exactitud aumenta en una red con un mecanismo de atención global. En efecto, hay una mayor número de predicciones que el modelo realizó correctamente en todos los escenarios y esto se debe a que todas las entradas tienen importancia y se consideran todos los estados ocultos tanto del codificador y del decodificador LSTM para calcular un vector de contexto de longitud variable, como indica el gráfico 5.22 de la exactitud con respecto a la longitud de las oraciones de lenguas estudiadas:

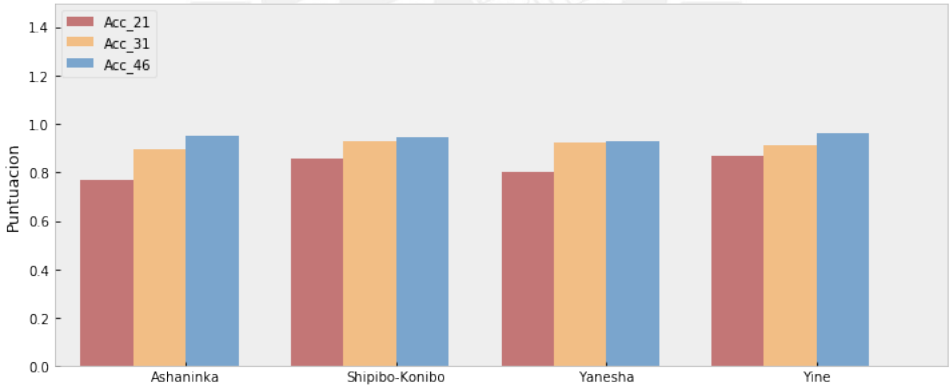


Figura 5.22: Exactitud de las oraciones de las lenguas estudiadas.

El gráfico muestra las exactitudes para oraciones de longitud 21(`Acc_21`), 31(`Acc_31`)y 46 (`Acc_46`) en cada una de las lenguas escogidas y que aumentan a medida que la longitud es mayor, ya que puede entender mejor las oraciones con un mayor contexto, ya que la cantidad de oraciones consideradas aumenta por la longitud y la red puede aprender mejor.

- Agregar capas y realizar un proceso de regularización al modelo seq2seq no conduce a una mejora en los valores obtenidos de la exactitud, con respecto al modelo base, lo que sugiere, que ambos casos la proporción de casos cuando se asigna una etiqueta correcta es casi la misma. Los gráficos 5.23, 5.24 y 5.25 muestran esa tendencia para todas las oraciones consideradas.

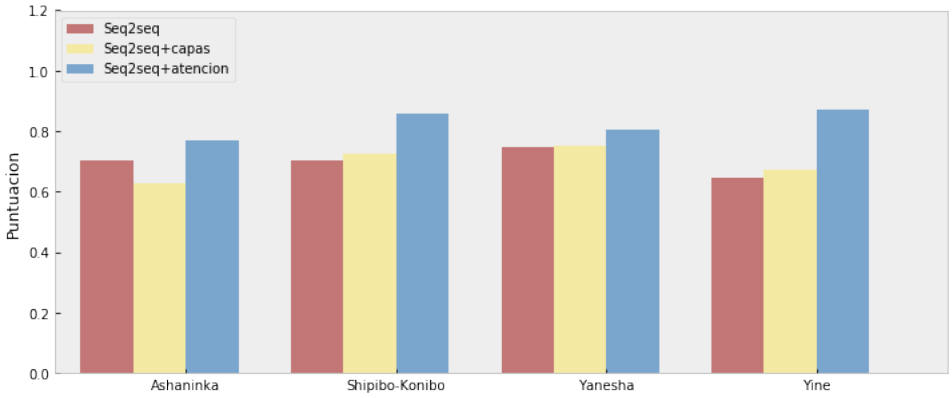


Figura 5.23: Exactitud para las oraciones de longitud 21 con los modelos estudiados.

El único caso donde no se cumple es en las oraciones de longitud 21 de Ashaninka donde se tiene un conjunto reducido de oraciones y el aumento de capas produjo que la red no pueda generalizar mejor al tener un modelo algo más complejo para tan pocos datos.

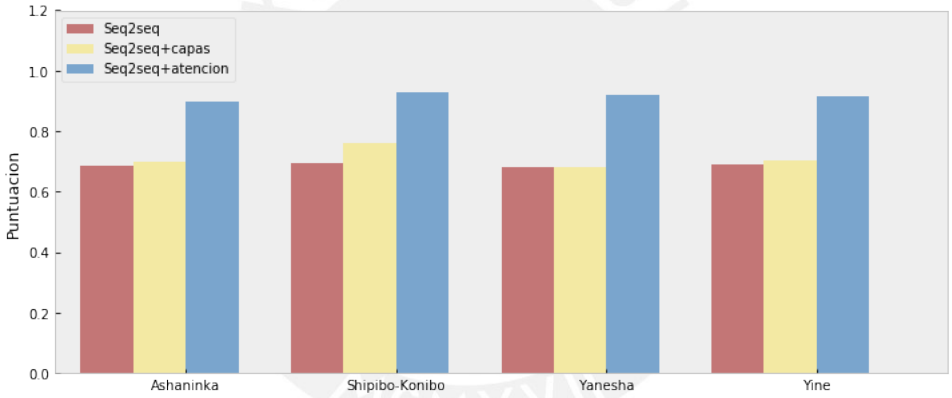


Figura 5.24: Exactitud para las oraciones de longitud 31 con los modelos estudiados

En el caso del modelo seq2seq con un mecanismo de atención, se tiene una mejor exactitud, ya que se asignan más etiquetas correctas y esto es por que la atención permite mirar la totalidad de la oración para establecer conexiones entre una palabra en particular y su contexto relevante ignorando otras palabras que simplemente no tienen mucho que ver con la palabra sobre la que está tratando de hacer una predicción.

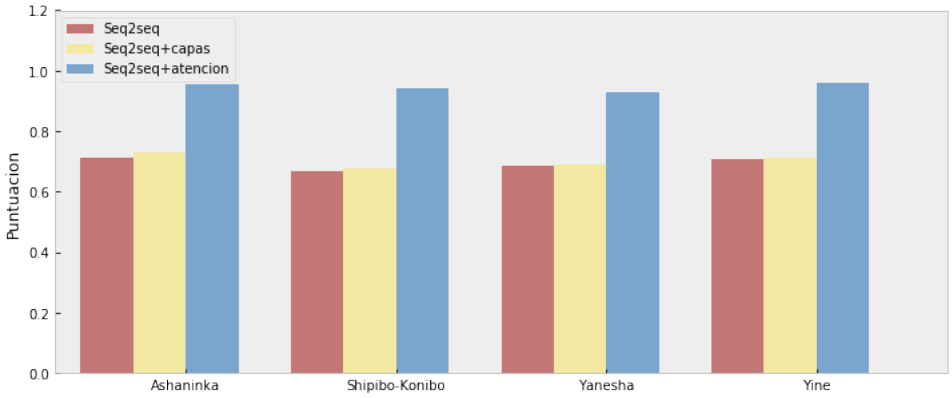


Figura 5.25: Exactitud para las oraciones de longitud 46 con los modelos estudiados

- BLEU es una métrica a nivel de palabras y tener un valor de casi cero en el modelo base y el modelo con regularización significa que la red no puede reconstruir palabras enteras correctamente cuando la longitud de las oraciones es pequeña.

  Utilizando un mecanismo de atención global, el valor de BLEU crece para cada una de las lenguas a medida que la longitud de las oraciones crece (BLEU_21), 31(BLEU_31)y 46 (BLEU_46)), como se muestra en el gráfico 5.26 de los valores BLEU, con respecto a la longitud de las oraciones de las lenguas estudiadas.

  Esta tendencia puede deberse a que BLEU otorga puntuaciones más altas a las palabras coincidentes en secuencia. Es decir, si una cadena de cuatro palabras resultante de la ejecución del modelo coincide con una oración de referencia en el mismo orden exacto, tendrá un impacto positivo en la puntuación BLEU que una cadena de dos palabras coincidentes.

  Esto demuestra entonces que un modelo seq2seq con un mecanismo de atención construye mejor palabras enteras coincidentes a medida que la longitud aumenta.

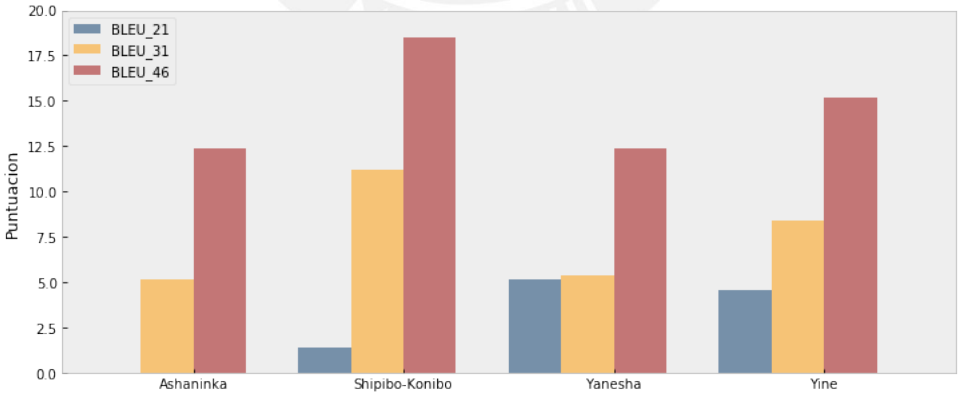


Figura 5.26: BLEU de las oraciones de las lenguas estudiadas.

- El puntaje BLEU obtenido demuestra un aumento en la correlación de los resultados obtenidos y las referencias dadas a medida que aumenta la longitud de la oración en la red con mecanismo de atención. Esto se debe a que la principal ventaja de los modelos basados en la atención es su capacidad para manejar oraciones largas de manera eficiente. A medida que aumenta el tamaño de la entrada, los modelos que no usan la atención perderán información si solo usan una representación final.

Los gráficos 5.27, 5.28 y 5.29, refuerzan la idea de que la red neuronal a nivel caracter construye mejor palabras enteras a medida que la longitud aumenta en cada una de las lenguas estudiadas.

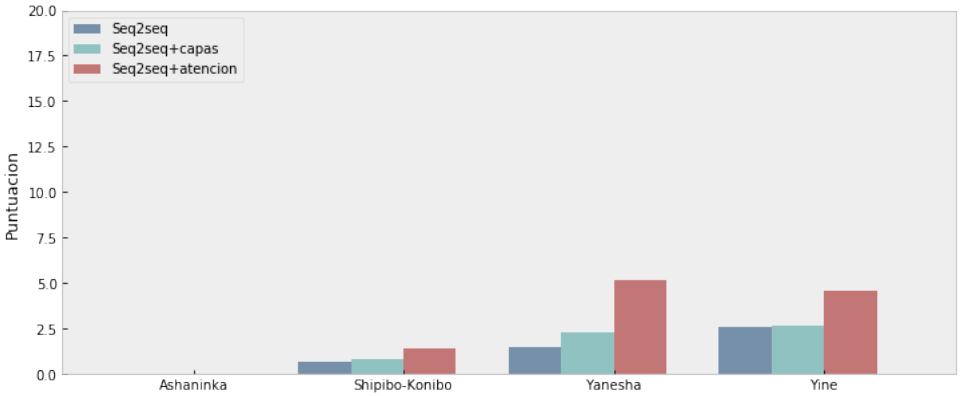


Figura 5.27: BLEU para las oraciones de longitud 21 con los modelos estudiados.

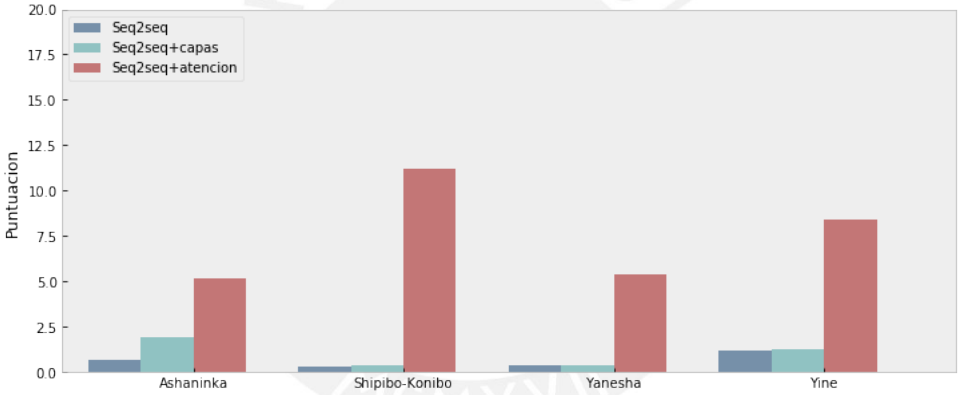


Figura 5.28: BLEU para las oraciones de longitud 31 con los modelos estudiados.

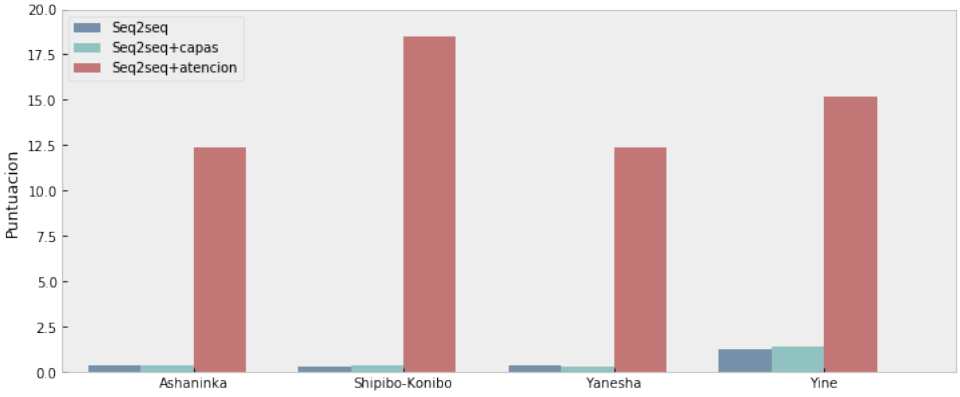


Figura 5.29: BLEU para las oraciones de longitud 46 con los modelos estudiados.

- Los valores de CharacTER, mejoran con un mecanismo de atención. Con la ayuda de la

atención, las dependencias entre las secuencias de entrada y salida ya no están restringidas por una distancia intermedia.

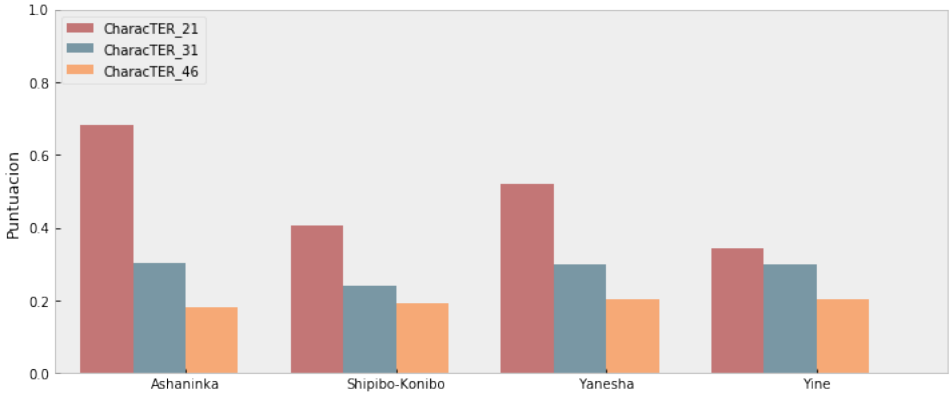


Figura 5.30: CharacTER de las oraciones de las lenguas estudiadas.

El gráfico 5.30 de los valores de CharacTER, con respecto a la longitud de las oraciones de las lenguas estudiadas, muestra que a una menor longitud de las oraciones indicadas por (`CharacTER_21`), 31(`CharacTER_31`)y 46 (`CharacTER_46`) se necesita una cantidad menor de edición posterior (inserciones, eliminaciones y sustituciones de una caracter, así como cambios de secuencias de caracteres.)

- La puntuación CharacTER es menor utilizando un mecanismo de atención, siendo una señal de menor esfuerzo para la edición de los resultados obtenidos de la red neuronal, para cambiar y coincidir exactamente con el archivo de texto de referencia.

  Este proceso se puede mejorar si es que el modelo predice primero una posición alineada única para la palabra objetivo actual y una ventana centrada alrededor de la posición de entrada que se utiliza para calcular un vector de contexto.

  Los gráficos 5.31, 5.32 y 5.33 muestra esta tendencia en todas las lenguas estudiadas y en la disminución progresiva del valor de la métrica a medida que la longitud de las oraciones aumenta.

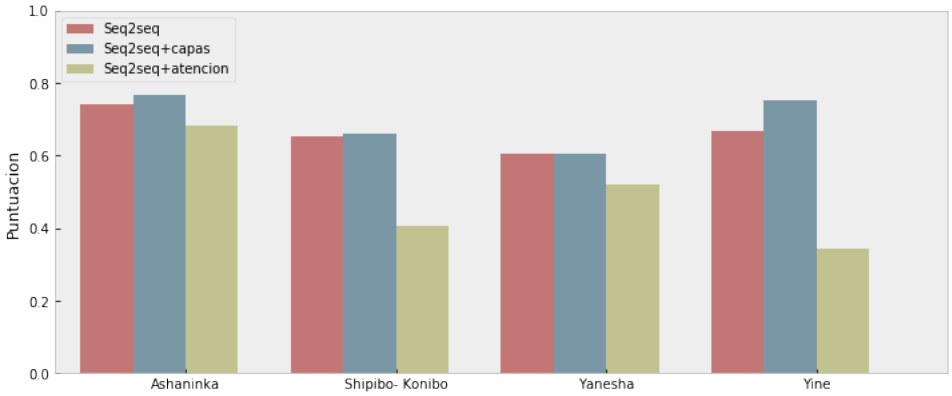


Figura 5.31: CharacTER para las oraciones de longitud 21 con los modelos estudiados.

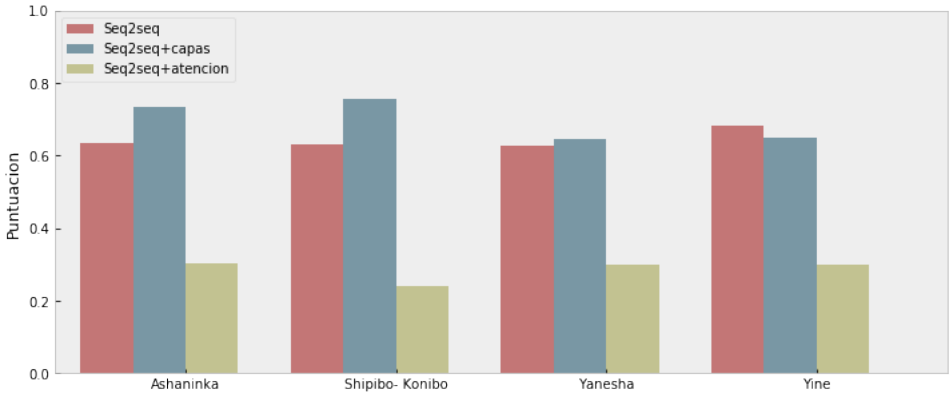


Figura 5.32: CharacTER para las oraciones de longitud 31 con los modelos estudiados.

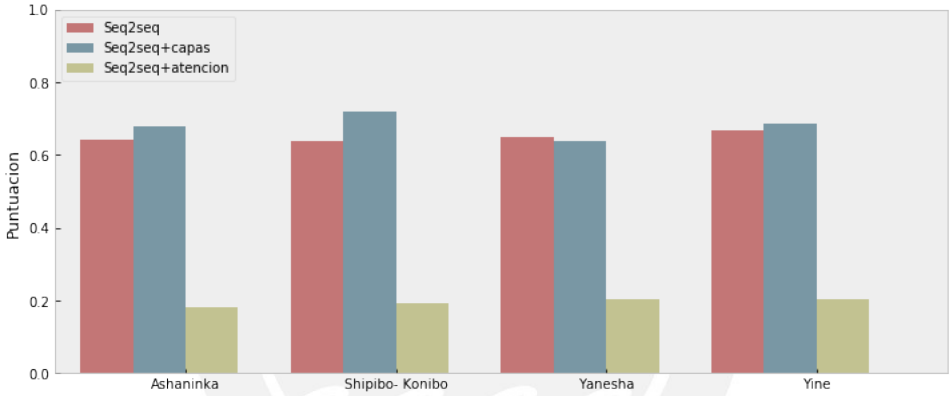


Figura 5.33: CharacTER para las oraciones de longitud 46 con los modelos estudiados.

- Las secuencias generadas por los modelos seq2seq y los valores de la métrica CharacTER obtenidos indican que las secuencias sí siguen patrones de caracteres consistentes y las palabras deben ser cercanas a lo esperado a menudo cuando la longitud de las oraciones aumenta en cada una de las lenguas estudiadas.
- Todas las inferencias sobre el BLEU se ha realizado en el conjunto de oraciones de prueba, comparando las cadenas de textos de dos archivos de texto generados por la red neuronal y algunos scripts de procesamiento inicial.

Conclusiones y trabajos futuros

6.1. Conclusiones

- Construimos un programa para descargar documentos en formato PDF y texto relacionados a cada una lenguas estudiadas de este trabajo. Empezamos analizando los documentos desde la página de PerúEduca que contiene información educativa que puede ser procesada.
- Creamos un corpus paralelo de oraciones con errores ortográficos y sin errores ortográficos, para entrenamiento, validación y prueba, introduciendo errores ortográficos, a través de un algoritmo de generación de errores sintéticos en las oraciones.
- Tomando como modelo base un modelo seq2seq se realizaron diversas pruebas con oraciones de distinta longitud. Se agrega una forma simple de `sampling` como una búsqueda greedy: en cada punto, escogemos la palabra más probable.

  Este enfoque tiene la ventaja de que es muy rápido, pero puede que la calidad de las secuencias de salida finales puede no ser la mejor.
- Se propone un modelo red neuronal con un mecanismo de atención global que mejora las métricas como BLEU y CharacTer con respecto al modelo base y modelos donde se apilan capas LSTM y se agregan condiciones de regularización para los pesos obtenidos y la salida obtenida. La mejora en estas métricas se da a medida que las longitudes de las oraciones aumenta.

6.2. Trabajos futuros

- Evaluar los modelos de seq2seq con otro tipo de métricas como es el caso de Maxmatch (M^2) [71] o GLEU [72] y realizar implementaciones a nivel de subpalabras y palabras.
- Presentar un conjunto de opciones a un usuario nativo, de manera que pueda escoger una palabra o oración correcta dentro de un conjunto de opciones, utilizando técnicas como Beam Search que expande todos los pasos siguientes posibles y controla las búsquedas paralelas a través de secuencia de probabilidades.
- Utilización de técnicas como BPE, una forma simple de compresión de datos en la que el par más común de bytes consecutivos de datos se reemplaza con un byte que no ocurre dentro de esos datos. En términos simples, lo que hace el BPE es dividir las palabras en sub-palabras y genera nuevos tokens que se utilizan de entrada en modelos a desarrollar, especialmente de las lenguas nativas peruanas que tienen propiedades aglutinantes como se describe en la sección de resultados.
- Implementación detallada de modelos seq2seq para cada una de las lenguas, ya que cada una de estas conserva ciertas propiedades lingüísticas a tener en cuenta para su diseño e implementación.

- Explorar otros mecanismos de atención, ya que la desventaja de un modelo de atención global que utilizamos, es que tiene que atender a todas las palabras en el lado de las entradas para cada palabra objetivo, lo que es computacionalmente costoso. Por ejemplo, podemos utilizar en el siguiente nivel una atención local que elige una posición en la oración de entrada que determinará una ventana de palabras a las que atiende el modelo.
- Conocer e implementar técnicas para manejar acentos, ingresar caracteres como `ch, sh` del alfabeto Ashaninka, `ch` del alfabeto Yine o `hu` de Shipibo-Konibo y conocer las estructuras sintácticas de cada una de las lenguas, así como el análisis del contexto de las oraciones, en escenarios como el `switch-reference` que presenta la lengua shipibo-Konibo o en Yanesha, donde las palabras a menudo se pueden modificar para que sean específicas del contexto.
- Mejorar el modelo que se utilizó para tratar las dependencias de largo alcance ya que las oraciones que se han obtenido son largas y se necesita un mejor análisis del contexto.

  Una primera opción sería utilizar, el modelo de red neuronal Transformer [76] y sus variantes [77] que usa solo el mecanismo de atención en lugar de las RNN para codificar cada posición, que puede ser paralelizado, acelerando así el entrenamiento y al tiempo de ejecución y que establece resultados de vanguardia en en diversas tareas del procesamiento de lenguaje natural.

Bibliografía

[1] *Base de Datos de Pueblos Indígenas u Originarios*: Diponible en http://bdpi.cultura.gob.pe/.

[2] *Documento nacional de lenguas originarias del Perú*. Ministerio de Educación, Perú 2013. Disponible en: http://repositorio.minedu.gob.pe/handle/123456789/3549.

[3] *Proceso de Consulta Previa del Plan Nacional de Educación Intercultural Bilingüe al 2021* Ministerio de Educacion, Peru 2016. Disponible en: http://www.minedu.gob.pe/campanias/pdf/eib-planes/rm-629-2016-minedu-plan-nacional-eib.pdf.

[4] Los pueblos shipibo-konibo, kakataibo e isconahua. Serie Nuestros pueblos indígenas $N°3$, Ministerio de Cultura, 2017.

[5] Los pueblos ashaninka, kakinte, nomatsigenga y yanesha. Serie Nuestros pueblos indígenas $N°1$, Ministerio de Cultura, 2014.

[6] Goldberg Yoav, Neural Network Methods for Natural Language Processing, A Publication in the Morgan & Claypool Publishers series, 2017.

[7] Forcada Mikel, *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) (organizado junto con LREC), 2006.

[8] Streiter Oliver S y Scannell Kevin P., *Implementing NLP projects for noncentral languages:instructions for funding bodies, strategies for developers*, Machine Translation 20(4):267-289, 2006.

[9] Faust Norma, *Lecciones para el aprendizaje del idioma shipibo-conibo. Documento de trabajo 1*. Instituto Lingüístico de Verano,1973.

[10] CHANA: Traducción automática entre español y shipibo-konibo. Lima, Perú: CHANA. Disponible en http://chana.inf.pucp.edu.pe. Pontificia Universidad Católica del Perú, 2017.

[11] Alva Carlos, Oncevay-Marcos Arturo, *Spell-Checking based on Syllabification and Character-level Graphsfor a Peruvian Agglutinative Language*, Association for Computational Linguistics, 2017.

[12] Cárdenas Ronald, Zeman Daniel, *A Morphological Analyzer for Shipibo-Konibo*, Association for Computational Linguistics 2018.

[13] Galarreta Ana Paula, Melgar Andres y Oncevay-Marcos Arturo, *Corpus Creation and Initial SMT Experiments between Spanish and Shipibo-konibo*, Conference: RANLP 2017 - Recent Advances in Natural Language Processing Meet Deep Learning.

[14] Madhvi Soni, Jitendra Singh Thakur, A Systematic Review of Automated Grammar Checking in English Language 2018. Disponible https://arxiv.org/pdf/1804.00540.pdf.

[15] Jurafsky Daniel y Martin James H., Speech and Language Processing (2nd Edition). Prentice-Hall, Inc., 2009.

[16] Kukich Karen, *Techniques for automatically correcting words in text*, ACM Computing Surveys (CSUR), 1992.

[17] Ahmadi Sina, *Attention-based Encoder-Decoder Networks for Spelling and Grammatical Error Correction*, Master Thesis, Paris Descartes University, 2017.

[18] Hamming Richard W., *Error detecting and error correcting codes*, Bell Labs Technical Journal, 1950.

[19] Winkler William E., *String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage*, 1990.

[20] Wagner Robert A. y Fisher Michael J., *The string-to-string correction problem*, Journal of the ACM (JACM), 1974.

[21] Damerau Fred J., *A technique for computer detection and correction of spelling errors*. Commun. ACM, 1964.

[22] Levenshtein Vladimir I., *Binary codes capable of correcting deletions, insertions, and reversals*, Soviet Physics Doklady, Vol 10, 1966.

[23] Daniel Naber, A rule-based style and grammar checker. (2003).

[24] Yannakoudakis Emmanuel J. y Fawthrop David, *The rules of spelling errors*. Information Processing & Management, 19(2):87-99, 1983.

[25] Means Linda G., *Cn yur cmputr raed ths?* En Proceedings of the second conference on Applied natural language processing, pages 93-100. Association for Computational Linguistics, 1988.

[26] Angell R. C., Freund, G. E., y Willet P., *Automatic spelling correction using a trigram similarity measure*. Inf. Process. Manage. 19, 255-256, 1983.

[27] Cherkassky V., Vassilas N., Brodt G., y Wechler H., *Conventional and associative memory approaches to automatic spelling checking*, Eng. Appl. Artif. Intell, 1982.

[28] Kernighan M. D., Church K. W. y Gale W. A., *A spelling correction program based on a noisy channel model*. In Proceedings of COLING-90, The 13th International Conference on Computational Linguistics, vol. 2 (Helsinki, Finland). Hans Karlgren, Ed. 205- 210, 1990.

[29] Church K. W. y Gale W. A., *Probability scoring for spelling correction*. Stat. Comput. 1, 93-103, 1991.

[30] Rosenfeld Ronal, *A maximum entropy approach to adaptive statistical language modeling*. In Computer, Speech and Language, 1996.

[31] Lottaz Claudio, Iseli Christian, Jongeneel Victor C. y Bucher Philipp, *Modeling sequencing errors by combining hidden markov models*. Bioinformatics, 19(suppl 2):ii103-ii112, 2003.

[32] Karpathy Andrej, Johnson Justin y Li Fei-Fei, *Visualizing and understanding recurrent networks*. CoRR, abs/1506.02078, 2015.

[33] Müller Andreas C. y Guido Sarah, Introduction to Machine Learning with Python: A Guide for Data Scientists, O'Reilly Media, 1 edition, 2016.

[34] Deep Learning LeCun Yann, Bengio Yoshua y Hinton Geoffrey, Deep learning. Nature Vol. 521, 2015.

[35] Bengio Yoshua , Courville Aaron y Pascal Vincent,
Representation Learning: A Review and New Perspectives IEEE transactions on pattern analysis and machine intelligence, 2013. Disponible en https://arxiv.org/abs/1206.5538.

[36] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, Multilayer feedforward networks are universal approximators. In: Neural networks 2.5 (1989), pp. 359-366.

[37] Kamath Uday, Liu John, Whitaker James, Deep Learning for NLP and Speech Recognition, Springer, 1 edition, 2019.

[38] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer, Deep contextualized word representations . NAACL 2018. Disponible en https://arxiv.org/pdf/1802.05365.pdf.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA. Disponible en https://arxiv.org/abs/1706.03762.pdf.

[40] Duyu Tang, Bing Qin, Ting Liu, Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1422-1432,Lisbon, Portugal, 17-21 September 2015.

[41] Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio. NIPS 2014 Deep Learning and Representation Learning Workshop. Disponible en https://arxiv.org/pdf/1412.3555.pdf.

[42] Mikolov Tomáš, Karafiát Martin, Burget Lukáš, Černocký Jan , Khudanpur Sanjeev, *Recurrent neural network based language model*. Interspeech, 2010.

[43] Vasilev Ivan, Slater Daniel, Spacagna Gianmario, Roelants Peter y Zocca Valentino, Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, Packt Publishing; 2 edition, 2019.

[44] Werbos Paul J., *Backpropagation through time: What it does and how to do it*. Proc. of the IEEE, 78(10):1550-1560, 1990.

[45] Stanislau Semeniuta, Aliaksei Severyn y Erhardt Barth. Recurrent Dropout without Memory Loss, 2016. Disponible en : https://arxiv.org/pdf/1603.05118.pdf.

[46] Pascanu Razvan, Çaglar Gülçehre, Cho Kyunghyun y Bengio Yoshua, *How to construct deep recurrent neural networks*. CoRR, abs/1312.6026, 2013.

[47] Graves Alex, Fernandez Santiago y Shmidhuber Jürgen, *Multidimensional recurrent neural networks*. arXiv 2007.

[48] Graves Alex, *Generating sequences with recurrent neural networks*. arXiv, 2013.

[49] Rudder Sebastian, Deep Learning for NLP Best Practices. In: (2017). Disponible en: http://ruder.io/deep-learning-nlp-best-practices/.

[50] Danijar Hafner. Tips for Training Recurrent Neural Networks. In: (2017). Disponible en: https://danijar.com/tips-for-training-recurrent-neural-networks/.

[51] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: arXiv preprint arXiv:1406.1078, 2014.

[52] Ilya Sutskever, Oriol Vinyals y Quoc V. Le, *Sequence to Sequence Learning with Neural Networks*, Advances in neural information processing systems, 2014. Disponible en https://arxiv.org/pdf/1409.3215.pdf.

[53] Allen Schmaltz, Yoon Kim, Alexander Rush, Stuart Shieber, Adapting Sequence Models for Sentence Correction. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017.

[54] Zheng Yuan y Ted Briscoe, Grammatical error correction using neural machine translation. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 380-386, San Diego, California. Association for Computational Linguistics, 2016.

[55] Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, Andrew Y. Ng, Neural Language Correction with Character-Based Attention, 2016. Disponible en https://arxiv.org/pdf/1603.09727.pdf.

[56] Tao Ge, Furu Wei, Ming Zhou, Fluency Boost Learning and Inference for Neural Grammatical Error Correction. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1055-1065. Melbourne, Australia, July 15-20, 2018.

[57] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, Neural machine translation by jointly learning to align and translate, 2016. Disponible en https://arxiv.org/pdf/1409.0473.pdf.

[58] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus, End-To-End Memory Networks, 2015. Disponible en https://arxiv.org/pdf/1503.08895.pdf.

[59] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation, 2015. Disponible en https://arxiv.org/pdf/1508.04025.pdf.

[60] Shaona Ghosh and Per Ola Kristensson, Neural networks for text correction and completion in keyboard decoding, 2017. Disponible en https://arxiv.org/pdf/1709.06429.pdf.

[61] Keisuke Sakaguchi, Kevin Duh, Matt Post, Benjamin Van Durme, Robsut Wrod Reocginiton via semi-Character Recurrent Neural Network.Published en AAAI, 2016. Disponible en https://arxiv.org/pdf/1608.02214.pdf.

[62] Pravallika Etoori, Manoj Chinnakotla, Radhika Mamidi, Automatic Spelling Correction for Resource-Scarce Language using Deep Learning. Proceedings of ACL, 2018, Student Research Workshop, pages 146-152. Melbourne, Australia, July 15-20, 2018.

[63] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, Kenneth Heafield, Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task. Proceedings of NAACL-HLT, 2018, pages 595-606 New Orleans, Louisiana, June 1-6, 2018. Disponible en https://arxiv.org/pdf/1804.05940.pdf.

[64] Mikel Artetxe, Holger Schwenk, Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond, 2018. Disponible en https://arxiv.org/pdf/1812.10464.pdf.

[65] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes y Jeffrey Dean, 2017. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. Transactions of the Association for Computational Linguistics, 5:339-351.

[66] Elizaveta Korotkova, Agnes Luhtaru, Maksym Del, Krista Liin, Daiga Deksne, Mark Fishel, Grammatical Error Correction and Style Transfer via Zero-shot Monolingual Translation, 2019. Disponible en https://arxiv.org/pdf/1903.11283.pdf. Rochester, April 2007.

[67] Jakub Náplava, Milan Straka. Grammatical Error Correction in Low-Resource Scenarios, 2019. Disponible en https://arxiv.org/abs/1910.00353.

[68] Yo Joong Choe, Jiyeon Ham, Kyubyong Park, Yeoil Yoon. A neural grammatical error correction system built on better pre-trainingand sequential transfer learning, 2019. Disponible en https://arxiv.org/abs/1907.01256.

[69] Rico Sennrich, Barry Haddow, Alexandra Birch. Neural machine translation of rare words with subword units, 2016. Disponible en https://arxiv.org/abs/1508.07909.

[70] Taku Kudo, John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. ACL,2018.

[71] Daniel Dahlmeier, Hwee Tou Ng, Better evaluation for grammatical error correction. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 568-572, 2012.

[72] Courtney Napoles, Keisuke Sakaguchi, Matt Post, Joel Tetreault, Ground Truth for Grammatical Error Correction Metrics. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2015.

[73] Martin Chodorow, Markus Dickinson, Ross Israel, and Joel R Tetreault. Problems in evaluating grammatical error detection systems. In COLING, pages 611-628, 2012.

[74] Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu.Bleu: a Method for Automatic Evaluation of Machine Translation. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (2002).

[75] Weiyue Wang, Jan-Thorsten Peter, Hendrik Rosendahl, Hermann Ney. CharacTer: Translation Edit Rate on Character Level. Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers,pp. 505-510, 2016.

[76] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pp. 5998-6008, 2017. Disponible en https://arxiv.org/abs/1706.03762.

[77] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Jamie Brew. Transformers: State-of-the-art Natural Language Processing (2019), HuggingFace Inc. Disponible en https://arxiv.org/abs/1910.03771.

[78] Bustamante Gina, Oncevay Arturo and Zariquiey Roberto. No data to crawl? Monolingual corpus creation from PDF files of four truly low-resource languages in Peru. Proceedings of the Twelfth International Conference on Language Resources and Evaluation (LREC 2020). In Press, 2020.

[79] Marek Rei, Mariano Felice, Zheng Yuan, Ted Briscoe. Artificial Error Generation with Machine Translation and Syntactic Patterns. Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, 2017.

[80] Matt Post. A Call for Clarity in Reporting BLEU Scores. Proceedings of the Third Conference on Machine Translation: Research Papers, pages186-191 (2018). Disponible en https://www.aclweb.org/anthology/W18-6319.

[81] Payne David L., Payne Judith K. Sanchez Ramos Jorge, Morfología, Fonología y Fonética del Ashéninca del Apurucayali. Instituto Lingüístico de Verano, 2011.

[82] Payne David, Diccionario Asheninka-Castellano, Documento de Trabajo $N°18$. Instituto Lingüístico de Verano, Edición Preliminar, 1980.

[83] Payne Judith,Lecciones para el aprendizaje del idioma Ashéninca, Serie Lingüística Peruana $N°28$, Instituto Lingüístico de Verano, 2008.

[84] Ego Aguirre Santa Cruz, Renzo Alberto, Desambiguación de morfemas polifuncionales en la traducción automática de lenguas minoritarias: el caso del enclítico $= n$ en el shipibo-konibo, Tesis para optar el grado de Magíster en Lingüística, Pontificia Universidad Católica del Perú Lima, 2018.

[85] Loriot James, Lauriout Erwin, Day Dwight, Diccionario Shipibo-Castellano, Serie Lingüística Peruana $N°31$ Ministerio de Educación, Instituto Lingüístico de Verano. Segunda edición, Perú, 2008.

[86] Valenzuela Pilar, Transitivity in Shipibo-Konibo Grammar. Tesis doctoral. Portland: Universidad de Oregon, 2003.

[87] Valenzuela Pilar, *Aspectos morfosintácticos de la relativización en shipibo-konibo (pano)*, Boletim do Museu Paraense Emílio Goeldi Ciencias Humanas. Belén, Volumen 1, número 1, pp. 123-134, 1998.

[88] Daigneault Anna Luisa, An ethnolinguistic study of the Yanesha? (Amuesha) language and speech community in Peru's Andean Amazon, and the traditional role of Ponapnora, a female rite of passage, Universita de Montréal (Faculté des arts et des sciences), Master's Thesis, 2009.

[89] Duff-Tripp Martha, Gramática del Idioma Yanesha (Amuesha). Serie Lingüística Peruana $N°43$ Ministerio de Educación, Instituto Lingüístico de Verano. Segunda edición, Perú, 2008.

[90] Hanson Rebecca , A grammar of Yine (Piro), La Trobe University. Research Centre for Linguistic Typology, 2010.

[91] Joyce Nies, Diccionario Piro(Yine), Serie Lingüística Peruana $N°22$ Instituto Lingüístico de Verano. Segunda edición, Lima Perú, 2008.

[92] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A Study of Translation Edit Rate with Targeted Human Annotation. Proceedings of Association for Machine Translation in the Americas, 2006.