PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Implementación de algoritmos para la identificación automática de lenguas originarias peruanas en un repositorio digital

Tesis para optar por el Título de Ingeniera Informática que presenta la bachillera:

Alexandra Mercedes Espichán Linares

20121825

Asesor: Mg. Félix Arturo Oncevay Marcos

Lima, Febrero de 2019

Resumen

Debido a la revitalización lingüística en el Perú a lo largo de los últimos años, existe un creciente interés por reforzar la educación bilingüe en el país y aumentar la investigación enfocada en sus lenguas nativas. Asimismo, hay que considerar que en el Perú actualmente alrededor de 4 millones de personas hablan alguna de las 47 lenguas nativas conservadas. Por tanto, hay una gran variedad de lenguas con las cuales trabajar, por lo que sería de utilidad contar con herramientas automáticas que permitan agilizar algunas tareas en el estudio e investigación de dichas lenguas.

De este modo, y desde el punto de vista de la informática, una de las primeras y principales tareas lingüísticas que incorporan métodos computacionales es la identificación automática de lenguaje, la cual se refiere a determinar el lenguaje en el que está escrito un texto dado, el cual puede ser un documento, un párrafo o incluso una oración. Este además es un paso esencial en el procesamiento automático de los datos del mundo real, donde una multitud de lenguajes pueden estar presentes, ya que las técnicas de procesamiento del lenguaje natural típicamente presuponen que todos los documentos a ser procesados están escritos en un lenguaje dado.

Por lo tanto, este trabajo se enfoca en tres pasos: (1) en construir desde cero un corpus anotado digital para 49 lenguas y dialectos indígenas peruanos, (2) en adaptarse a los enfoques de aprendizaje de máquina estándar y profundo para la identificación de lenguas, y (3) en comparar estadísticamente los resultados obtenidos.

Los resultados obtenidos fueron prometedores, el modelo estándar superó al modelo de aprendizaje profundo tal como se esperaba, con una precisión promedio de 95.9%. En el futuro, se espera que se aproveche el corpus y el modelo para tareas más complejas.

Tabla de Contenido

Índice de Ilustraciones

Índice de Tablas

Capítulo 1. Generalidades

1.1 Problemática

En la última década, el cambio social en el Perú está avanzando tan rápidamente que los hablantes de lenguas originarias tienen la necesidad de priorizar el español en su vida cotidiana. Esto ocurre especialmente como resultado de la migración urbana, incluso en casos en los que existen fuertes conexiones emocionales con el idioma indígena. Sin embargo, al mismo tiempo, en algunas regiones andinas y amazónicas parecen estar tomando fuerza las identidades indígenas, lo cual puede constituir la condición necesaria para una eventual revitalización lingüística en las futuras generaciones (Heggarty & Pearce, 2011).

De esta manera, dado este proceso de revitalización lingüística en la actualidad, se ha intensificado el interés por seguir mejorando la enseñanza bilingüe en las escuelas, lo cual está a cargo del Plan Nacional de Educación Intercultural Bilingüe (EIB) de acuerdo a la Resolución Ministerial N°629-2016-MINEDU (2016). Es por esto que algunas universidades y el gobierno peruano han preparado una diversidad de material para la enseñanza bilingüe, incluyendo folletos de diversos temas como vocabulario, matemáticas y ciencias, así como cuentos tradicionales de diversos pueblos escritos en lenguas originarias (Ministerio de Educación, Perú, s.f).

Asimismo, hay que considerar que actualmente alrededor de 4 millones de personas en el Perú hablan alguna lengua originaria, destacando el quechua y el aimara, muchas de las cuales pertenecen a pueblos indígenas que se encuentran en situación de aislamiento voluntario. Además, se sabe que en el Perú existe una gran variedad de lenguas y que aún se conservan 47 lenguas nativas divididas en 19 familias lingüísticas (Ministerio de Educación, Perú, 2013). Por tanto, hay una gran variedad de lenguas con las cuales trabajar, por lo que

sería de utilidad contar con herramientas automáticas que permitan agilizar algunas tareas en el estudio e investigación de dichas lenguas.

De este modo, y desde el punto de vista de la informática, una de las primeras y principales tareas lingüísticas que incorporan métodos computacionales es la identificación automática de lenguaje (Malmasi & Dras, 2015), la cual se refiere a determinar el lenguaje en el que está escrito un texto dado, el cual puede ser un documento, un párrafo o incluso una oración. Este además es un paso esencial en el procesamiento automático de los datos del mundo real, donde una multitud de lenguajes pueden estar presentes, ya que las técnicas de procesamiento del lenguaje natural típicamente presuponen que todos los documentos a ser procesados están escritos en un lenguaje dado (por ejemplo: Español). Es por esto que la identificación automática del lenguaje es una de las funcionalidades elementales que tiene muchas aplicaciones útiles, entre las cuales se incluyen la traducción automática, el análisis de sentimientos, la lexicografía o la recuperación de la información (Malmasi & Dras, 2015). Por ejemplo, en el contexto de la recuperación de la información, la identificación del lenguaje puede ayudar a filtrar documentos por lenguaje e incluso por dialecto. Por esta razón, es un componente clave de muchos servicios web, donde conocer el lenguaje en el que una página o documento está escrito es una consideración importante para determinar si podría ser de interés para un usuario en particular de una herramienta de búsqueda (Lui & Baldwin, 2012).

No obstante, las lenguas originarias peruanas pueden ser catalogadas como lenguas minoritarias o de escasos recursos computacionales (Forcada, 2006). Es decir, estas lenguas no poseen suficientes textos digitalizados (o ninguno) y el corpus textual listo para el procesamiento que se ha encontrado es mínimo y pertenece a muy pocas lenguas. Por esta razón, si se pretende iniciar estudios en computación lingüística en varias lenguas originarias del Perú, será necesario implementar y digitalizar un repositorio propio, el cual debe incluir

una cantidad de documentos que alberguen una cantidad suficiente de palabras para trabajar. Este repositorio es necesario para realizar las tareas descritas de identificación automática, y posteriormente podría ser usado para llevar a cabo muchas más investigaciones en lingüística computacional en el país.

De este modo, a partir de la generación del repositorio indicado, se puede implementar una herramienta primaria como un identificador de lenguajes automático, que sería la primera barrera a romper para muchos de los lenguajes originarios peruanos que no han sido objeto de estudio de la lingüística computacional, siendo la excepción sólo algunos de ellos como el Quechua (Rios, 2015) o el Shipibo-konibo (Pontificia Universidad Católica del Perú, 2017).

Por otro lado, para construir un identificador de lenguaje usando métodos tradicionales es necesario representar las lenguas de una forma en la que la máquina pueda entenderlas y trabajar con ellas. Para esto, se seleccionan y extraen características de los lenguajes. Sin embargo, existen métodos de aprendizaje profundo en los que la máquina aprende automáticamente una manera de representar las lenguas. Por lo tanto, en este proyecto se pretende usar ambos métodos y probar cual tiene mejores resultados.

Por las razones descritas, en el presente proyecto se propone la construcción de un nuevo repositorio digital de textos que centralice al menos 10 lenguas originarias peruanas, la implementación de un algoritmo tradicional y un algoritmo de aprendizaje profundo que permitan identificar de manera automática el lenguaje de textos escritos en lenguas originarias peruanas. Luego, se seleccionará el mejor algoritmo y se implementará una aplicación web de acceso libre en la que los usuarios puedan consultar a qué lenguaje pertenece un documento, párrafo u oración. Asimismo, a partir del repositorio se podrán realizar búsquedas por palabra, donde se les mostrará las oraciones de los documentos en las que esta palabra ha sido encontrada, así como una cita que los dirija a la fuente del documento. De esta manera, se espera contribuir en las investigaciones de computación

lingüística en el país, dando los primeros pasos para algunas lenguas originarias peruanas que aún no han sido objeto de estudio de esta área de trabajo.

1.2 Objetivos

1.2.1 Objetivo general

Implementar algoritmos para la identificación automática de lenguas originarias peruanas

1.2.2 Objetivos específicos

O 1. Recolectar y procesar datos léxicos y corpus textuales de diversas lenguas originarias peruanas.

O 2. Implementar un modelo algorítmico tradicional de clasificación automática para lenguas originarias peruanas.

O 3. Implementar un modelo algorítmico de aprendizaje profundo para la clasificación automática de lenguas originarias peruanas.

O 4. Implementar una aplicación web que contenga el mejor modelo algorítmico y componentes desarrollados para la presentación de resultados.

1.2.3 Resultados esperados

R 1. Componente de pre procesamiento de los textos en lenguas originarias peruanas. (O1)

R 2. Base de datos de lenguas originarias peruanas. (O1)

R 3. Caracterización de las oraciones. (O2)

R 4. Modelo algorítmico tradicional de aprendizaje supervisado entrenado. (O2)

R 5. Validación del modelo algorítmico tradicional. (O2)

R 6. Representación directa de las oraciones en una red neuronal. (O3)

R 7. Modelo algorítmico de aprendizaje profundo entrenado. (O3)

R 8. Validación del modelo algorítmico de aprendizaje profundo. (O3)

R 9. Resultados de la experimentación numérica para comparar ambos modelos algorítmicos. (O4)

R 10. Aplicación web que integre el mejor modelo algorítmico. (O4)

1.3 Herramientas y Métodos

1.3.1 Herramientas

Python

Python[1] es un lenguaje de programación de alto nivel de código abierto, sencillo de usar y con la capacidad de integrarse bien con otros lenguajes. Posee gran cantidad de librerías de terceros para distintas tareas como, por ejemplo, aprendizaje de máquina. En la actualidad Python es muy usado para tareas de aprendizaje de máquina. Esto se debe a que tiene una buena proporción complejidad-rendimiento y ofrece un conjunto completo de herramientas para producir aprendizaje automático. Por estas razones se ha elegido este lenguaje para el desarrollo de este proyecto.

Jupyter Notebook

Según el sitio oficial de Jupyter Notebook (2019), esta es una aplicación web de código abierto que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto explicativo. Sus usos incluyen: limpieza y transformación de datos, simulación numérica, modelado estadístico, aprendizaje de máquina y más. Jupyter notebook tiene soporte para una variedad de lenguajes de programación, entre ellos Python. En este

[1] https://www.python.org

proyecto se usará esta aplicación para mostrar de manera más ordenada y entendible el código a desarrollar y para visualizar resultados.

Scikit-learn

Scikit-learn[2], también conocida como sklearn, es una librería enfocada en aprendizaje de máquina en Python. Provee herramientas para minería de datos y análisis de datos. Incluye diversos algoritmos de clasificación, regresión, clustering, reducción de la dimensionalidad, selección de modelo y pre procesamiento de datos. En este proyecto se usará esta librería para probar diversos métodos de clasificación tradicionales, en la caracterización de oraciones y en la generación de reportes.

Enchant

Enchant[3] es una librería de Python que usa diccionarios para verificar si una palabra, frase u oración pertenece a un lenguaje dado. Incluye funcionalidades para proveer palabras propias que se pueden incluir como parte de un diccionario en un lenguaje específico. En este proyecto se usará esta librería para filtrar frases en español o inglés como parte del pre procesamiento de los textos.

Seaborn

Seaborn[4] es una librería de Python usada para la visualización de resultados. Provee una interfaz de alto nivel para mostrar gráficos estadísticos. En este proyecto se usará esta librería para mostrar gráficos en los reportes de evaluación de los modelos implementados y para realizar una exploración de datos de manera visual.

---

[2] http://scikit-learn.org/stable/

[3] https://github.com/AbiWord/enchant

[4] https://seaborn.pydata.org

BeautifulSoup

BeautifulSoup[5] es una librería de Python que provee métodos para navegar, buscar y extraer datos de estructuras HTML de manera sencilla. Permite extraer solo texto, limpiando elementos como etiquetas propias de los archivos HTML y seleccionar etiquetas dados los valores de sus atributos. En este proyecto se usará esta librería para extraer texto de páginas web de manera automática dado que algunos textos están divididos en varias páginas y hacerlo de manera manual tomaría mucho tiempo.

Tensorflow

Tensorflow[6] es una librería de código abierto disponible para Python y otros lenguajes de programación. Es usado para la computación numérica usando gráficos de flujo de datos. Su arquitectura permite desplegar la computación en uno o más CPUs o GPUs. Es útil para el desarrollo de redes neuronales. En este proyecto se usará esta librería para la implementación de redes neuronales profundas.

Keras

Keras[7] es una librería de Python enfocada en el desarrollo y experimentación rápida de redes neuronales. Es capaz de ejecutarse sobre Tensorflow, CNTK y Theano. Permite un sencillo y rápido desarrollo de prototipos. En este proyecto se usará esta librería para la implementación de redes neuronales profundas.

---

[5] https://www.crummy.com/software/BeautifulSoup/

[6] https://www.tensorflow.org

[7] https://keras.io

Django

Django[8] es un framework de desarrollo web de Python de código abierto. Permite un rápido desarrollo y un diseño limpio y pragmático. En este proyecto se usará este framework para el desarrollo de la página web de acceso libre que contendrá el mejor modelo de identificación de lenguas originarias peruanas.

Sublime Text

Sublime Text[9] es un editor de texto avanzado para código. Tiene soporte para el marcado de diversos lenguajes de programación, entre ellos Python. Además, posee diversas características útiles para el desarrollo de aplicaciones. Debido a ello su uso es muy popular en el desarrollo de aplicaciones web en diversos lenguajes. En este proyecto se usará Sublime Text para el desarrollo de la página web de acceso libre.

Microsoft Excel

Según el sitio oficial de Microsoft Excel (2019), este software permite organizar datos, numéricos o de texto, en hojas o libros de cálculo. Además, permite realizar complejos análisis automáticamente. Asimismo, resume los datos con vistas previas de opciones de tablas dinámicas para compararlas. Recomienda diagramas y gráficos que ilustran los patrones de los datos. En este proyecto se usará esta aplicación para realizar la experimentación numérica en la comparación de los dos algoritmos a implementar.

---

[8] https://www.djangoproject.com/

[9] https://www.sublimetext.com/

PDFMiner

PDFMiner[10] es una herramienta disponible para Python, para extraer información de archivos PDF. Permite obtener la posición exacta del texto en una página, además de otra información como la fuente o líneas. Incluye un convertidor que puede transformar archivos PDF en otros formatos de texto. Además, posee un analizador de PDF que puede ser usado para otros propósitos además del análisis de textos. En este proyecto se usará esta herramienta para extraer el texto de archivos PDF.

Base de datos documental

Según el sitio oficial de MongoDB (2019), cuando se introdujeron las bases de datos relacionales, los esquemas de datos eran bastante simples y directos, y tenía sentido concebir los objetos como conjuntos de relaciones. Por ejemplo, un objeto de artículo podría estar relacionado con una categoría (un objeto), un comentario (otro objeto), etc. Estas bases de datos relacionales podrían consultarse con un lenguaje de consulta estructurado estándar o SQL. Pero el entorno para los datos, así como la programación, ha cambiado desde entonces debido a la aparición de la computación en la nube, la necesidad de guardar data no estructurada y los métodos de desarrollo ágil. En respuesta a estos cambios, surgieron nuevas formas de almacenar datos (por ejemplo, las bases de datos NoSQL) que permiten agrupar los datos de forma más natural y lógica, y que disminuyen las restricciones en el esquema de la base de datos. Una de las formas más populares de almacenar datos es una base de datos documental donde cada registro y sus datos asociados se consideran un "documento". Los beneficios de guardar la data de esta manera son que los documentos son unidades independientes, la lógica de la aplicación es más fácil de escribir y los datos no estructurados

[10] https://github.com/euske/pdfminer

se pueden almacenar fácilmente. En este proyecto se usará la base de datos MongoDB, la cual la base de datos NoSQL más usada en la actualidad.

1.3.2 Métodos

KDD

Knowledge Discovery in Databases (KDD) o descubrimiento de conocimiento en bases de datos es un proceso de extracción de conocimiento que se refiere al proceso no trivial de descubrir conocimiento e información potencialmente útil dentro de los datos contenidos en algún repositorio de información. Es un proceso iterativo que exhaustivamente explora volúmenes muy grandes de datos para determinar relaciones (Han et al. 2011). Este proceso se puede resumir en:

- Limpieza de datos para remover el ruido y datos inconsistentes.
- Integración de datos, donde múltiples fuentes de datos podrían ser combinadas.
- Selección de datos, donde la data relevante para la tarea de análisis es recuperada de la base de datos.
- Transformación de datos, donde la data es transformada y consolidada en formas apropiadas para la minería mediante la realización de operaciones de resumen o agregación.
- Minería de datos, el cual es un proceso esencial donde se aplican métodos inteligentes para extraer patrones en la data.
- Evaluación de patrones para identificar los patrones de real interés, representando el conocimiento basado en medidas interesantes.
- Presentación de conocimiento, donde las técnicas de visualización y representación del conocimiento son usadas para presentar el conocimiento minado a los usuarios.

Se usará esta metodología a lo largo de todo el proyecto.

Bootstrap sampling

Simpson & Mayer-Hasselwander (1986) definen *bootstrap sampling* como un método para obtener varios conjuntos de datos simulados directamente de la data medida. Estos conjuntos de datos simulados o muestras de bootstrap son usados para estimar la varianza estadística en el parámetro de interés. Una muestra de bootstrap es formada seleccionando un subconjunto de datos de la data medida a través de un muestreo aleatorio con reemplazo. En otras palabras, las muestras de bootstrap se forman seleccionado datos de manera aleatoria sin tomar en cuenta si un dato ya ha sido seleccionado o no. Por tanto, un dato puede no aparecer, aparecer una vez o varias veces en un conjunto de datos simulado mediante este método. En este proyecto se usará este método para realizar la experimentación numérica ya que se necesita probar cada algoritmo sobre varios conjuntos de datos.

Contraste de hipótesis estadísticas

Según Ross (2007), un contraste de hipótesis estadísticas consiste en determinar si una hipótesis o teoría formulada sobre una población deba ser rechazada o no, evaluando la población seleccionada a través de una muestra aleatoria de la misma. Esto requiere el establecer una hipótesis previa (hipótesis nula H0) y compararla con una hipótesis alternativa (H1). Al establecer las hipótesis relacionadas a un parámetro de la población, se requiere encontrar un estadístico pivote que esté relacionado al parámetro, con ese estadístico se obtendrá el criterio de decisión según el valor que alcance la muestra. Además, es necesario determinar el porcentaje de riesgo de equivocarse (nivel de confianza o significancia), que se está dispuesto a asumir, con ese valor se logra determinar el valor crítico (frontera que marca el cambio de decisión). Al observar la región en la que se centra el valor de la muestra respecto al valor crítico, se determina si se acepta o se rechazan las hipótesis.

Método de comparación entre dos muestras relacionadas

Demšar (2006) afirma que para comparar múltiples muestras relacionadas existen diversas pruebas estadísticas que se usan en base al tipo y cantidad de muestras que se compararán. En este caso, dado que se compararán dos modelos algorítmicos, estas muestras relacionadas serían los rendimientos de ambos modelos medidos sobre los mismos conjuntos de datos, preferentemente usando las mismas particiones en los conjuntos de entrenamiento y prueba.

De esta manera, según Montgomery (2017) existen pruebas estadísticas para asegurar que hay una diferencia estadísticamente relevante entre dos muestras y determinar cuál tiene una media mayor. Sin embargo, estas pruebas en su mayoría trabajan bajo el supuesto de que las muestras tienen una distribución normal. Por lo tanto, el primer paso será verificar si las muestras tienen distribución normal.

La prueba de Kolmogorov es una prueba muy usada para verificar si es razonable la hipótesis de que una función F(x) es una distribución normal con una media y varianza específicas, basado en las observaciones que están disponibles (Wilcox, 2005). Por lo tanto, la prueba de Kolmogorov será usada como primer paso para determinar que ambas muestras poseen una distribución normal.

Una vez se haya asegurado que las muestras tienen una distribución normal, se usará la prueba F (Fisher, 1956), para determinar si las varianzas de ambas muestras son significativamente homogéneas (hipótesis nula). Esta prueba divide la variabilidad total en la variabilidad entre los clasificadores, la variabilidad entre los conjuntos de datos y la variabilidad residual (error). Si la variabilidad entre los clasificadores es significativamente mayor que la variabilidad del error, podemos rechazar la hipótesis nula y concluir que existen algunas diferencias entre las varianzas de las muestras.

Finalmente, si las varianzas de las muestras son significativamente homogéneas, se usará la prueba Z, la cual evalúa las medias de las muestras y determina en la prueba a una cola cuál de las muestras posee una media mayor. Por el contrario, si las varianzas de las muestras son significativamente diferentes, se usará la prueba t-student para muestras con varianza distinta. Esta prueba, al igual que la prueba Z, evalúa las medias de las muestras y determina en la prueba a una cola cuál de las muestras posee una media mayor. (Montgomery, 2017)

1.3.3 Mapeo de objetivos, resultados y verificación

En las Tablas Tabla *1*, Tabla *2*, Tabla *3* y Tabla *4* se detallan las herramientas o métodos a usar para lograr cada resultado esperado de cada objetivo específico, así como el medio de verificación correspondiente.

Tabla 1. Herramientas o métodos a usar y medio de verificación de los resultados esperados del primer objetivo

Objetivo: Recolectar y procesar datos léxicos y corpus textuales de diversas lenguas originarias peruanas			
Resultado	**Meta física**	**Medio de verificación**	**Herramientas o métodos a usar**
Componente de pre procesamiento	Software	- Funcionalidad de extracción de textos de páginas web. - Funcionalidad de extracción de textos de archivos PDF.	Python Enchant BeautifulSoup Pdfminer Jupyter Notebook KDD

		- Funcionalidad de limpieza de textos. - Oraciones en lenguas originarias peruanas pre procesadas.	
Base de datos de lenguas originarias peruanas	Base de datos	- Esquema de la base de datos. - Base de datos que contiene oraciones y palabras en lenguas originarias peruanas. - Exploración de los datos de la base de datos.	Base de datos documental Seaborn KDD

Tabla 2. Herramientas o métodos a usar y medio de verificación de los resultados esperados del segundo objetivo.

Objetivo: Implementar un modelo algorítmico tradicional de clasificación automática para lenguas originarias peruanas			
Resultado	**Meta física**	**Medio de verificación**	**Herramientas o métodos a usar**
Caracterización de las oraciones	Software	- Matriz de características de las	Python Scikit-learn

		oraciones en lenguas originarias peruanas.	Jupyter Notebook KDD
Modelo algorítmico tradicional entrenado	Software	- Prueba sobre distintos parámetros de caracterización y métodos de clasificación. - Clasificador entrenado.	Python Scikit-learn Jupyter Notebook KDD
Validación del modelo algorítmico tradicional	Software	- Reporte del funcionamiento del clasificador sobre datos de prueba.	Python Scikit-learn Seaborn Jupyter Notebook KDD

Tabla 3. Herramientas o métodos a usar y medio de verificación de los resultados esperados del tercer objetivo.

Objetivo: Implementar un modelo algorítmico de aprendizaje profundo para la clasificación automática de lenguas originarias peruanas			
Resultado	**Meta física**	**Medio de verificación**	**Herramientas o métodos a usar**

Representación directa en una red neuronal	Software	- Función que convierta las oraciones a una estructura que pueda ser usada en una red neuronal. - Oraciones convertidas en vectores.	Python Keras Jupyter Notebook KDD
Modelo algorítmico de aprendizaje profundo entrenado	Software	- Clasificador entrenado - Función que clasifique un texto dado.	Python Tensorflow Keras Jupyter Notebook KDD
Validación del modelo algorítmico de aprendizaje profundo	Software	- Reporte del funcionamiento del clasificador sobre datos de prueba.	Python Scikit-learn Seaborn Jupyter Notebook KDD

Tabla 4. Herramientas o métodos a usar y medio de verificación de los resultados esperados del cuarto objetivo.

Objetivo: Implementar una aplicación web que contenga el mejor modelo algorítmico y componentes desarrollados para la presentación de resultados

Resultado	**Meta física**	**Medio de verificación**	**Herramientas o métodos a usar**
Resultados de la experimentación numérica para comparar ambos modelos algorítmicos	Reporte	- Reporte de comparación de ambos modelos	Microsoft Excel Bootstrap sampling Contraste de hipótesis estadísticas Método de comparación entre dos muestras relacionadas
Aplicación web	Software	- Página web de acceso libre. - Pruebas de la funcionalidad de identificación del lenguaje sobre textos de prueba.	Python Django Sublime Text KDD

1.4 Alcance y limitaciones

Este proyecto se llevará a cabo a partir de la construcción de un repositorio digital de textos en lenguas originarias peruanas.

Primero, los datos recolectados serán pre procesados en un componente de software que comprende la limpieza (corregir errores en el texto, filtrar caracteres raros) de las oraciones que se encuentran en un texto y extraer únicamente las oraciones en lenguas originarias.

Luego, se probarán modelos algorítmicos tradicionales de aprendizaje supervisado. Para esto, se implementará un programa que permita extraer las características por oración de todos los textos para cada lengua originaria, creándose de esta manera vectores de características por cada lengua. A continuación, se realizarán pruebas usando estos vectores de características sobre los modelos algorítmicos tradicionales de aprendizaje supervisado.

Después, se realizarán pruebas usando los datos pre procesados sobre una red neuronal recurrente a diseñar.

Luego, se compararán los resultados de ambos métodos con el fin de seleccionarse las técnicas de procesamiento y el algoritmo que será usado en el identificador automático de lenguaje.

Finalmente, se implementará una aplicación web de libre acceso en la que los usuarios puedan consultar el repositorio de textos y la herramienta de identificación del lenguaje. Esta aplicación permitirá que un usuario ingrese un documento, párrafo u oración e identifique de qué lengua originaria se trata, mostrando un informe detallado que muestre las lenguas originarias a las que más se acerca con sus respectivas probabilidades. Además, los usuarios podrán tener acceso a través de la aplicación a las fuentes de los textos del repositorio, y podrán realizar una búsqueda por palabras que muestre las oraciones del repositorio en las que esta palabra se encuentra con un enlace a su respectiva fuente original.

1.5 Viabilidad

1.5.1 Viabilidad Técnica

Se posee acceso libre a los lenguajes y a las herramientas que se van a usar. Quien suscribe posee experiencia usando el lenguaje de programación Python. Además, hay la posibilidad de usar el repositorio git[11] y los servidores locales del grupo de investigación GRPIAA (Grupo de Reconocimiento de Patrones e Inteligencia Artificial Aplicada) de la Pontificia Universidad Católica del Perú para realizar las pruebas.

1.5.2 Viabilidad Temporal

Para el desarrollo de las tareas de este proyecto se estima una duración de 6 meses. En el Anexo 1 se presenta una planificación en el tiempo de estas tareas, observándose que sí se puede ejecutar en un tiempo apropiado.

1.5.3 Viabilidad Económica

Se posee acceso libre a los lenguajes y a las herramientas que se van a usar, lo cual evita que se tengan limitaciones financieras y de accesibilidad en el proyecto.

1.6 Alcance, Limitaciones y Riesgos

En la Tabla 5 se presentan los riesgos identificados en el proyecto, así como las medidas de mitigación y contingencia planificadas para hacer frente a estos riesgos en caso se materialicen.

[11] https://github.com/grpiaa-pucp

Tabla 5. Riesgos identificados en el proyecto

Descripción	**Síntomas**	**Probabilidad**	**Impacto**	**Severidad**	**Mitigación**	**Contingencia**
No lograr obtener resultados óptimos para el modelo de aprendizaje profundo pues demora mucho tiempo entrenar el modelo.	Los modelos de aprendizaje profundo suelen demorar en entrenarse.	0.5	0.8	0.4	Se puede usar el servidor de la sección de informática que posee un GPU potente.	Entrenar un solo modelo de aprendizaje profundo, el cual sea comparado con el modelo tradicional.
No encontrar suficientes documentos escritos en lenguas originarias peruanas.	Las lenguas originarias peruanas son lenguas de escasos recursos.	0.1	0.8	0.08	Usar datos léxicos ya recolectados a los que se nos brindó acceso.	Ya se han recolectado documentos de lenguas originarias con anticipación

Capítulo 2. Marco Legal/Regulatorio/Conceptual/otros

2.1 Marco conceptual

A continuación, se explicarán conceptos relacionados con el problema planteado y las técnicas y términos de reconocimiento de patrones e inteligencia artificial utilizados en las investigaciones revisadas y en la propuesta de solución.

En cuanto a los conceptos generales relacionados a lingüística, la RAE (2016) define a la **lengua** como el sistema de comunicación verbal y casi siempre escrito, propio de una comunidad humana, al **lenguaje** como el conjunto de sonidos articulados con que el hombre manifiesta lo que piensa o siente, y a la **familia lingüística** como el conjunto de lenguas que derivan de una misma lengua, por ejemplo: la familia de lenguas románicas.

Por otro lado, en cuanto a los conceptos relacionados al reconocimiento de patrones e inteligencia artificial, Murphy (2012) define **aprendizaje de máquina** o aprendizaje automático como un conjunto de métodos que pueden detectar automáticamente patrones en datos y luego usar los patrones no cubiertos para predecir datos futuros o para llevar a cabo otros tipos de toma de decisiones bajo incertidumbre. Los algoritmos de aprendizaje de máquina se agrupan en función de la salida de los mismos, de esta manera, se tiene el aprendizaje supervisado y el aprendizaje no supervisado.

El **aprendizaje supervisado**, también llamado predictivo, tiene como meta aprender a llegar desde entradas “x” a salidas “y” dado un set de pares entrada-salida llamado el set de entrenamiento. Por tanto, el objetivo principal del aprendizaje supervisado es crear una función capaz de predecir el valor de salida correspondiente a cualquier entrada válida, después de haber visto una serie de ejemplos, los datos de entrenamiento. La salida de la función puede ser un valor numérico, como en los problemas de **regresión**, o una etiqueta de clase, como en los problemas de **clasificación**. La identificación automática del lenguaje es

un problema de aprendizaje supervisado, y específicamente es un problema de clasificación, en el que se tiene un conjunto de datos de entrenamiento (un repositorio de textos en diferentes lenguajes) y se trata de entrenar un modelo en base a estos datos que prediga el lenguaje de un nuevo texto (data de prueba), para finalmente asignar a este texto una clase (el lenguaje al que pertenece).

Por otro lado, LeCun et al. (2015) afirman que las **técnicas de aprendizaje de máquina tradicionales** son limitadas en su habilidad de procesar los datos naturales en su forma cruda. Por tanto, construir un sistema de aprendizaje de máquina requiere una considerable experiencia en el dominio para diseñar un extractor de características que transformen la data cruda a una adecuada representación interna o vector de características, desde el cual el sistema de aprendizaje pueda detectar o clasificar patrones en los datos.

Asimismo, LeCun et al. (2015) definen **aprendizaje de la representación** como un conjunto de métodos que permiten alimentar a una máquina con datos crudos y descubrir automáticamente representaciones necesarias para la clasificación. Los métodos de **aprendizaje profundo** son métodos de aprendizaje de la representación con múltiples niveles de representación. Estos métodos son obtenidos mediante la composición de módulos simples, pero no lineales que transforman la representación en un nivel a una representación en un nivel mas alto y abstracto. De esta manera, con tales transformaciones se pueden aprender funciones muy complejas. Para tareas de clasificación, los niveles más altos de representación amplifican los aspectos de los datos de entrada que son importantes para la discriminación y suprimen las variaciones irrelevantes.

Las **redes neuronales recurrentes** (RNN) son útiles para tareas que involucran datos de entrada secuenciales, tales como texto y audio. Este tipo de redes neuronales procesan un elemento a la vez de la secuencia de entrada, manteniendo en sus unidades ocultas un vector de estado que implícitamente contiene información acerca de la historia de todos los

elementos pasados de la secuencia. En la Ilustración 1 se puede observar la representación de una red neuronal recurrente, las neuronas artificiales (unidades ocultas agrupadas bajo el nodo s, con valores s_t en el tiempo t) obtienen datos de entrada de otras neuronas en pasos anteriores. De esta manera, una RNN puede mapear una secuencia de entrada con elementos x_t en una secuencia de salida con elementos o_t, con cada o_t dependiendo de todo el anterior $x_{t'}$ (para $t' \leq t$). Los mismos parámetros U, V, W son usados en cada instante de tiempo.

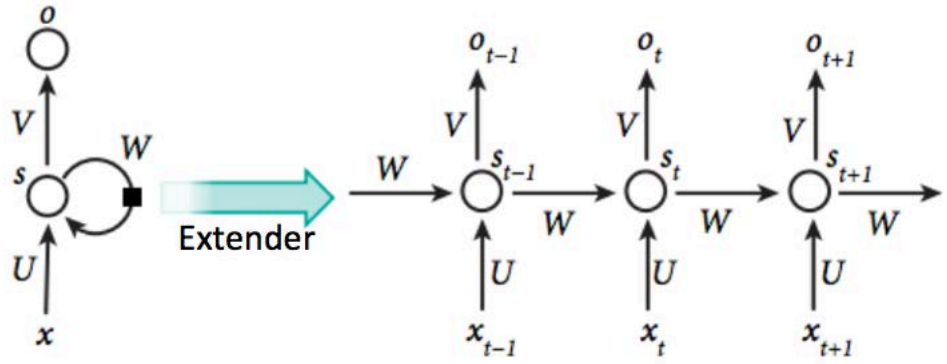


Ilustración 1. Representación de una red neuronal recurrente y su extensión en el tiempo.

Adaptado de (*LeCun et al., 2015)*

Sin embargo, LeCun et al. (2015) afirman que a pesar de que el objetivo principal de las RNN es aprender dependencias a largo plazo, evidencia teórica y empírica ha mostrado que es difícil aprender a almacenar información durante mucho tiempo.

Para corregir esto, una idea es aumentar la red con una memoria explícita. La primera propuesta de este tipo son las redes ***Long short-term memory*** (LSTM), las cuales usan unidades ocultas especiales, cuyo comportamiento natural es recordar los datos de entrada durante mucho tiempo. Una unidad especial llamada la célula de memoria actúa como un acumulador, tiene una conexión a sí misma así que copia su propio estado y acumula la señal externa, pero esta auto-conexión es multiplicativamente cerrada por otra unidad que aprende a decidir cuándo borrar el contenido de la memoria.

Por otro lado, comúnmente se usa una capa anterior a la capa LSTM que es denominada ***embedding***, en este caso se usará un *embedding* de caracter a vector. De esta manera, esta capa aprende a representar como un vector cada caracter que aparece en los datos de entrenamiento. Por lo tanto, cuando cada caracter en una palabra de longitud l se reemplaza por su vector de *embedding*, obtenemos una matriz con dimensiones dxl, donde d representa el tamaño del *embedding* elegido. (Jaech et al., 2016)

Finalmente, el **procesamiento de lenguaje natural** (PLN) es un área de investigación y aplicación que explora cómo las computadoras pueden ser usadas para entender y manipular el lenguaje natural o el habla para hacer cosas útiles. Los investigadores de PLN tienen como objetivo recoger información sobre cómo los seres humanos entienden y usan el lenguaje, de manera que herramientas y técnicas apropiadas pueden ser desarrolladas para hacer que los sistemas computacionales entiendan y manipulen el lenguaje natural para realizar tareas deseadas como, por ejemplo: la identificación automática de un lenguaje a partir de un texto dado. (Chowdhury, 2003)

Capítulo 3. Estado del Arte

Para implementar un identificador automático de lenguaje, es necesario contar con corpus textuales (conjunto de textos digitalizados) anotado en los lenguajes que se usará. Sin embargo, no todos los lenguajes cuentan con corpus anotados que sean los suficientemente extensos, por lo que son conocidos como lenguajes de pocos recursos (*low-resourced languages*) (Forcada, 2006). Dado que se va a desarrollar un identificador de lenguaje basado en textos para lenguas de pocos recursos, se realizó una búsqueda de artículos en Scopus[12] sobre identificación automática de lenguaje en un contexto de escasos recursos. La cadena de búsqueda utilizada es la siguiente:

```
TITLE-ABS-KEY ( language ( identification OR classification ) AND text AND ( less OR under ) resource )
```

Por otro lado, dado que se pretende aplicar técnicas de aprendizaje profundo para comparar el rendimiento de este tipo de métodos versus los métodos tradicionales, se hizo una búsqueda en Scopus con la siguiente cadena:

```
TITLE-ABS-KEY ( language ( identification OR classification ) AND text AND deep ( learning OR network ))
```

A partir de los resultados obtenidos, se seleccionaron los estudios que fueron considerados como más relevantes para el proyecto a desarrollar tomando en cuenta las lenguas con las que se trabajaron, dando prioridad a aquellos estudios que trabajaron con lenguas de escasos recursos, y la fecha de publicación. A continuación, se presenta una descripción de cada uno de ellos.

[12] https://www.scopus.com/

3.1 Revisión y discusión

A continuación, se presentan cuatro estudios en los cuales se han aplicado métodos tradicionales de aprendizaje supervisado para realizar la identificación automática del lenguaje sobre lenguas de escasos recursos.

Malmasi y Dras (2015) presentan el primer estudio empírico para distinguir entre textos Persa y Dari (Persa del este, hablado mayormente en Afganistán) al nivel de oraciones. Como Dari es un lenguaje con pocos recursos, se desarrolló un corpus de 28 mil oraciones extraídas de textos de noticias (14 mil por lenguaje) para esta tarea. Usando n-grams de caracteres y palabras, lo cual consiste en separar subsecuencias de "n" elementos de una secuencia dada, se logra diferenciar estas lenguas con un 96% de precisión. Al usar n-grams, desde el punto de vista lingüístico, las subcadenas capturadas por esta función, dependiendo del orden, pueden capturar implícitamente diversas características sublexicales incluyendo letras simples, fonemas, sílabas, morfemas y sufijos. En este estudio se usa un algoritmo de aprendizaje supervisado o clasificación denominado SVM para realizar la clasificación multiclase, en particular se usa el paquete LIBLINEAR SVM (Chang y Lin, 2011) que ha demostrado ser eficiente para los problemas de clasificación de textos con un gran número de características y documentos. Se prueban distintas características como unigrams, bigrams y trigrams de caracteres y unigrams y bigrams de palabras obteniéndose la mejor precisión al combinar todas las características.

Botha y Barnard (2012) investigan los factores que determinan el rendimiento de la identificación del lenguaje basada en textos, con un especial énfasis en las 11 lenguas oficiales de Sudáfrica usando las estadísticas de n-grams como características para la clasificación. Para un valor fijo de "n", SVM generalmente supera a los otros clasificadores, pero los clasificadores más simples son capaces de manejar valores mayores de "n". En este estudio se listan distintos enfoques para la identificación de lenguaje. La colección de

métodos más simples incluye ordenación de ranking de n-grams, el algoritmo de aprendizaje supervisado Naïve Bayes, producto punto normalizado, basado en centroides y entropía relativa. También se puede aplicar una serie de algoritmos más complejos del área de reconocimiento de patrones como: máquinas vectoriales de soporte (SVM), muestreo Monte Carlo, modelos de Markov, árboles de decisión, redes neuronales y regresión lineal múltiple. Sin embargo, solo se ponen a prueba tres de ellos: SVM (3-gram), Naïve Bayes (3-gram y 6-gram) y ordenamiento de ranking de n-grams (3-gram y 6-gram) con distintos tamaños de textos de entrenamiento: 200K, 400K, 800K, 1.6M y 2M caracteres y tres cantidades de caracteres de prueba: 15 caracteres (2-3 palabras), 100 caracteres (oración larga) y 300 caracteres (párrafo). De esta manera, se encuentra que un clasificador Naïve Bayes usando 6-gram es la mejor opción de clasificador global, con un mejor rendimiento con cantidades grandes de caracteres. Para cadenas de 100 caracteres o más se obtiene una precisión de 99.4%, para cadenas más cortas, que consisten de 15 caracteres, la mejor precisión obtenida fue de 83%.

Selamat y Akosu (2016) proponen un algoritmo basado en características léxicas que es capaz de realizar la identificación del lenguaje utilizando una mínima cantidad de datos de entrenamiento. Para este estudio se utilizó un conjunto de datos extraídos de la Declaración Universal de los Derechos Humanos en 15 idiomas: 4 lenguas nigerianas (Hausa, Igbo, Tiv y Yoruba), dos lenguas sudafricanas (Ndebele y Zulu), Swahili del este de África, dos lenguas de Ghana (Akuapem y Asante), dos lenguas del este de Asia (Bahasa Melayu y Bahasa Indonesia), Croata, Serbio, Eslovaco e Inglés. De la lista de lenguas que se usaron, las dos lenguas asiáticas no son de bajos recursos, pero son lenguas cercanamente relacionadas, al igual que el Serbio y el Croata, estas lenguas fueron añadidas para probar el rendimiento del sistema en lenguas relacionadas. El inglés, a pesar de ser la lengua con más recursos del mundo, es incluida en este estudio para probar la viabilidad de la propuesta a lenguas con

muchos recursos. La técnica que se usó se basa en el método del corrector ortográfico de Pienaar y Snyman (2010). Esta técnica consiste en comparar las palabras del texto objetivo con las palabras que existen en los vocabularios de las lenguas disponibles. Si un lenguaje en particular tiene en su léxico el mayor porcentaje de palabras del texto objetivo y este porcentaje supera a un benchmark fijado por los autores, el sistema concluye que este texto debe estar escrito en ese lenguaje. La mejora que se propuso en este estudio fue indexar las palabras de los vocabularios de acuerdo a su longitud. De esta manera, si en el texto objetivo se tiene una palabra de tres letras, esta palabra solo se comparará con las palabras de tres letras de los vocabularios, reduciendo así tiempos de búsqueda innecesaria. Finalmente, la precisión promedio del método fue del 93%, donde se logró identificar correctamente lenguas cercanamente relacionadas en la mayoría de casos, y se obtuvo una mejora del 73% en términos de tiempo respecto al algoritmo original.

Grothe et al. (2008) presentan dos experimentos para comparar diferentes algoritmos de identificación del lenguaje a partir del corpus Leipzig Corpora Colletion (LCC) (Quasthoff et al., 2006) y de artículos seleccionados aleatoriamente de Wikipedia. Para el primer experimento se usó data de entrenamiento de los siguientes lenguajes: catalán, danés, holandés, inglés, francés, alemán, italiano, noruego y sueco. Para el segundo experimento se usó la misma data de entrenamiento anterior, pero se extendió con los siguientes lenguajes: estonio, finlandés, serbio y turco, los cuales, a pesar de no ser lenguajes de escasos recursos a nivel general, sí lo son a nivel relativo comparado con varios lenguajes del primer grupo mencionado. En este estudio, se consideran los enfoques basados en palabras cortas (SW), palabras frecuentes (FW) y n-grams (NG), y son combinados con el método de clasificación “Ad-Hoc Ranking”. Este método consiste en la comparación de dos modelos ordenados en frecuencia descendente. Para usar este método, se extraen características textuales de cada documento no clasificado en un modelo del documento de la misma manera en la que se

extraen características en un modelo del lenguaje a partir de la data de entrenamiento. Luego, las características contenidas en el modelo del documento son sucesivamente buscadas en los modelos del lenguaje, para esto se usa la medida *out-of-place* la cual es calculada comparando los dos modelos y su posición asignando un valor, si una característica no se encuentra en el modelo del lenguaje se le asigna un valor máximo. La distancia total resultante de las medidas *out-of-place* es usada para asignar el lenguaje del documento, de esta manera, el modelo con la distancia más pequeña es elegida como el lenguaje que describe el documento. En el primer experimento se usó un valor máximo de la medida *out-of-place* fijo, los mejores resultados que se tuvieron para cada enfoque fueron: FW 25% (99.2%), SW 4 (94.1%) y NG 3 (79.2%). En el segundo experimento se usó un máximo de la medida *out-of-place* dinámico obteniéndose cerca de 100% de precisión para los tres enfoques.

Por otro lado, estudios más recientes han realizado la tarea de identificación del lenguaje usando métodos de aprendizaje profundo. A continuación, se presentan tres estudios que han aplicado estas técnicas para resolver este problema.

Jaech et al. (2016) introducen un modelo jerárquico para la identificación de lenguaje en mensajes de redes sociales. El modelo que presentan aprende representaciones del lenguaje a nivel de caracteres y palabras contextualizadas. Este modelo tiene dos componentes principales que son entrenados juntos, de inicio a fin. El primero es un "char2vec" (caracter a vector), este aplica una red neuronal convolucional (CNN) a una secuencia de caracteres Unicode de una palabra delimitada por espacios en blanco, retornando un vector por palabra. El segundo componente es una red neuronal recurrente LSTM bidireccional que mapea una secuencia de dichos vectores de palabras a una etiqueta (un lenguaje). Los autores prueban su método sobre 2 conjuntos de datos de tweets: El conjunto de datos TweetLID (Zubiaga et al., 2014), el cual se enfoca en seis lenguas

comúnmente habladas en la península Ibérica: Español, Portugués, Catalán, Galicio, Inglés y Basque; y el conjunto de datos Twitter70, el cual fue publicado por el equipo de ingeniería del lenguaje de Twitter en 2015. Los lenguajes de este conjunto de datos vienen de las familias afroasiáticas, dravidias, indoeuropeas, sino-tibetanas y tao kadai. Finalmente, este modelo obtiene 77.1 y 91.2 % de precisión en cada conjunto de datos respectivamente, mejorando el resultado frente a otros métodos.

Bjerva (2016) trabaja en la tarea de discriminación entre lenguas similares, usando dialectos de diversas lenguas tales como Español, Francés y Portugués. El sistema que propone usa representaciones de bytes en una red residual profunda (ResNet). Las ResNets son construidas apilando unidades llamadas residuales. Estas unidades pueden ser vistas como una serie de capas convolucionales con un atajo que facilita la propagación de señales en la red neuronal. El sistema desarrollado por el autor, llamado ResIdent, es probado sobre 3 conjuntos de datos distintos. De esta manera, se obtiene 84.88% de precisión en el conjunto de datos A, 68.80% de precisión en el conjunto de datos B1 y 69.80% de precisión en el conjunto de datos B2. Aunque se logró un rendimiento aceptable, un mayor ajuste de las representaciones de entrada y la arquitectura del sistema mejorarían probablemente el rendimiento.

Kocmi y Bojar (2017) proponen un método para la identificación lingüística multilingüe. Un ejemplo sencillo de este tipo de identificación es cuando el documento puede estar en dos o tres lenguas y sólo queremos sus nombres. Sin embargo, los autores dan un paso más allá y proponen un método en el que el objetivo es identificar los tramos de cada una de las lenguas. El método propuesto se basa en Redes Neuronales Recurrentes Bidireccionales y fue construido usando capas con células “Gated Recurrent Unit” (GRU), recientemente propuestas por Cho et al. (2014). También se probó con células *Long short-term memory* (LSTM), pero lograron resultados ligeramente peores. Luego, el modelo genera

una distribución de probabilidad sobre todas las etiquetas de lenguajes. Para determinar el idioma de un caracter, se toma la etiqueta con el valor máximo. De esta manera, el modelo toma letras como datos de entrada y provee una etiqueta de lenguaje a cada una de ellas. Luego, siempre que se necesita reconocer el lenguaje de un documento, tomamos el lenguaje asignado por el modelo a la mayoría de las letras. Este modelo fue entrenado utilizando el método de descenso de gradiente estocástica de primer orden Adam (Kingma & Ba, 2015) y el criterio de entrenamiento es la función de pérdida de entropía cruzada. Para evitar el sobre entrenamiento, se usó el *dropout* (Srivastava et al., 2014). La idea clave es eliminar aleatoriamente las conexiones. Esto evita que las neuronas se co-adapten demasiado, es decir, empiecen a depender demasiado de las salidas de otras neuronas. Finalmente, el modelo funciona bien en tareas de identificación lingüística monolingües y multilingües en seis conjuntos de datos que cubren 131 idiomas. El método mantiene la precisión también para documentos cortos y entre distintos dominios, por lo que es ideal para su uso sin necesidad de preparación de datos de entrenamiento.

3.2 Conclusiones

De los estudios presentados anteriormente, se puede observar que se han usado diversos métodos de aprendizaje de máquina para realizar la identificación del lenguaje en distintas lenguas del mundo.

Asimismo, se observa que el rendimiento de estos métodos muchas veces depende de la cantidad de datos que se posea para cada lenguaje. Sin embargo, se demuestra que a pesar de contar con pocos datos para el entrenamiento de un identificador automático, como sucedería en el caso de las lenguas peruanas ya que son lenguas de escasos recursos, se puede obtener precisiones aceptables.

Finalmente, a partir de la revisión del estado del arte, se identifica a los n-grams como la característica más usada al entrenar clasificadores con métodos tradicionales. Por otro lado, el método de aprendizaje profundo más usado para la tarea de identificación del lenguaje son las redes neuronales recurrentes (RNN) usando capas LSTM o GRU.

Por lo tanto, para este proyecto, se usará distintas combinaciones de n-grams como características de los lenguajes sobre distintos métodos de aprendizaje supervisado tradicionales para identificar el mejor modelo de aprendizaje supervisado tradicional. Además, se probará el rendimiento de las redes neuronales recurrentes con capas LSTM y distintos parámetros en la identificación automática de lenguas peruanas. Finalmente, se seleccionará el mejor modelo para ser usado en la aplicación web.

Capítulo 4. Componente de pre procesamiento

4.1 Introducción

En este capítulo, se presentará el desarrollo del resultado esperado 1. Para este fin, primero se recolectaron algunos documentos en lenguas nativas peruanas. Las principales fuentes de obtención de estos documentos son: la Biblia, la Declaración Universal de los Derechos Humanos, cuentos y diccionarios. Algunos de estos documentos son archivos PDF, otros tuvieron que ser generados a partir de un componente desarrollado para la extracción automática de textos de páginas web. Los archivos que están en formato PDF primero deben pasar por una herramienta que extraiga el texto en archivos de texto. Estos textos muchas veces poseen oraciones en español e inglés. Además, se pueden encontrar caracteres raros, correos electrónicos, direcciones URL de páginas web y otros elementos que deberían ser eliminados.

4.2 Descripción del resultado

Este componente de software tiene funcionalidades para la extracción de textos de páginas web, en específico para la extracción de textos de la página web de la biblia, para extraer texto de archivos PDF y para realizar la limpieza de los textos.

En la limpieza de los textos, se elimina caracteres no alfanuméricos, correos, páginas web y otros patrones específicos. Además, elimina las oraciones en español e inglés y extrae las oraciones en lenguas nativas peruanas.

4.3 Desarrollo del resultado

4.3.1 Funcionalidad de extracción de textos de páginas web

En la página www.bible.com la biblia está traducida a muchas lenguas, entre ellas muchas lenguas originarias peruanas. Las direcciones URL de la biblia tienen el siguiente formato:

www.bible.com/en-GB/bible/<*identificador de cada biblia*>/<*código del libro*>.<*capítulo*>.<*código iso 639-3 de la lengua*>NT

Por ejemplo, el capítulo 1 del libro de Mateo de la biblia en la lengua Achuar se encuentra en la siguiente dirección:

https://www.bible.com/en-GB/bible/4/MAT.1.acuNT

Dado que extraer el texto de todos los capítulos de todos los libros de todas las lenguas de manera manual puede resultar un trabajo que tome mucho tiempo, se implementó esta funcionalidad.

Primero, se desarrolló una función con los siguientes parámetros: ruta de archivo de destino, el número identificador de la biblia y el código iso 639-3. Esta función hace uso de la librería BeautifulSoup4 y está desarrollada en base a la estructura de la página web de la biblia y por tanto es específica para esta página. El objetivo de esta función es extraer la biblia de una lengua en el archivo de destino.

Luego, para automatizar aún más la tarea, se creó un archivo csv que contiene información sobre todas las lenguas originarias peruanas que poseen una biblia traducida en la página web mencionada. Este archivo contiene información como el nombre de la lengua, el número identificador de cada biblia y el código iso 639-3 de la respectiva lengua. El nombre de archivo destino fue construido usando el código iso 639-3 y la terminación NT.

Finalmente, bastó crear un bucle que extraiga cada biblia en base a cada línea del archivo csv. De esta manera, se pudo extraer automáticamente todas las biblias en lenguas originarias peruanas que se encontró en esa página web.

4.3.2 Funcionalidad de extracción de textos de archivos PDF

Dado que en la web se encontraron algunos archivos PDF que contenían cuentos o diccionarios en lenguas originarias peruanas, fue necesario desarrollar una funcionalidad que extraiga el texto de archivos PDF.

Por ello, se implementó una función con los siguientes parámetros: ruta de archivo de origen y ruta de archivo de destino. Esta función hace uso de la librería PDFMiner y extrae el texto del archivo PDF de origen al archivo de texto de destino.

Para automatizar aún más la tarea, se desarrolló una función que busque todos los archivos PDF del corpus y los extraiga en archivos de texto en una carpeta denominada “txt” con el mismo nombre del archivo original, pero con la extensión txt.

4.3.3 Lenguas recolectadas

Además de la extracción de las biblias y archivos PDF de internet, también se encontró la Declaración Universal de los Derechos Humanos en muchas lenguas originarias peruanas en la página web de la UDHR, para esto no se necesitó una funcionalidad extra ya que bastaba con copiar el texto y guardarlo en un archivo pues todo estaba en una sola página. De esta manera, en total se logró recolectar datos de 49 lenguas, incluyendo 19 dialectos de Quechua y 3 dialectos de Asháninka. En el Anexo 2 se detalla las fuentes de cada uno de los archivos recolectados. En la Tabla 6 se presenta información sobre las lenguas de las cuales se logró recolectar datos. Se muestra la familia lingüística a la que pertenece cada lengua, su código ISO-639-3 y la cantidad aproximada de hablantes.

Tabla 6. Información de las lenguas recolectadas. Información extraída del Documento nacional de lenguas originarias del Perú (Ministerio de Educación, 2013)

Familia lingüística	Lengua	ISO-639-3	Hablantes
Arawak	Asháninka	cni	88,703
	Ashéninka del Pajonal	cjo	10,000
	Ashéninka del Pichis	cpu	8,774
	Kakinte	cot	439
	Matsigenka	mcb	11,275
	Nomatsigenga	not	8,016
	Yanesha	ame	7,523
	Yine	pib	3,261
Aru	Aymara	aym	443,248
Bora	Bora	boa	748
Cahuapana	Shawi	cbt	21,650
Huitoto	Murui	huu	1,864
Jibaro	Achuar	acu	11,087
	Awajún	agr	55,366
	Wampis	hub	10,163
Kandozi	Kandozi	cbu	3,255
Pano	Amahuaca	amc	301
	Capanahua	kaq	348
	Cashinahua	cbs	2,419
	Kakataibo	cbr	1,879
	Matsés	mcf	1,724
	Sharanahua	mcd	486
	Shipibo	shp	22,517
	Yaminahua	yaa	600
Peba-yagua	Yagua	yad	5,679
Quechua	Quechua de Ambo-Pasco	qva	90,000
	Quechua de Ayacucho	quy	850,050
	Quechua de Cajamarca	qvc	30,000
	Quechua de Huallaga	qub	40,000
	Quechua de Huamalies	qvh	72,500
	Quechua de Lambayeque	quf	21,496
	Quechua de Margos	qvm	114,000
	Quechua de Panao	qxh	50,000
	Quechua de San Martin	qvs	44,000
	Quechua de Yauyos	qux	6,500
	Quechua del Callejón de Huaylas	qwh	300,000
	Quechua del Cusco	quz	1,115,000
	Quechua del Este de Apurimac	qve	218,352

Familia lingüística	Lengua	ISO-639-3	Hablantes
	Quechua del Norte de Conchucos	qxn	250,000
	Quechua del Norte de Junin	qvn	60,000
	Quechua del Norte de Pastaza	qvz	2,100
	Quechua del Sur de Conchucos	qxo	250,000
	Quechua del Sur de Pastaza	qup	2,200
	Quechua Huaylla Wanca	qvw	298,560
Shimaco	Urarina	ura	4,854
Tacana	Ese eja	ese	588
Tikuna	Tikuna	tca	6,982
Tucano	Secoya	sey	921
Záparo	Arabela	arl	403

4.3.4 Funcionalidad de limpieza de textos

Los textos obtenidos en los pasos anteriores necesitan ser pre procesados, esta funcionalidad incluye la limpieza de caracteres no alfanuméricos, números, patrones específicos usando expresiones regulares, y la eliminación de oraciones en Español e Inglés usando la librería Enchant.

Los patrones que fueron eliminados son los siguientes:

- Citas bíblicas. Ejemplo: (Mr. 1.22; Lc. 6.47-49)
- Indicadores de elementos de sintaxis. Ejemplo: say-1.OBJ-3 Tana-PL-TOP
- Correos.
- Tiempo de grabación. Ejemplo: 00:04:35.743
- Rutas de archivos. Ejemplo: file:/Users/Documents/Quechua/Yauyos/Bird.eaf
- Dinero. Ejemplo: S/.200,000
- Rutas de páginas web. Ejemplo: http://creativecommons.org/licenses/by-nc-nd/3.0

Esta funcionalidad además separa el texto en oraciones, considerando al punto como el elemento que separa cada oración, elimina las comas, punto y coma y demás signos de

puntuación, solo deja el caracter comilla (‘) dado que este es parte del abecedario de muchas lenguas originarias peruanas. Además, se verifica que la primera palabra de cada oración sea mayúscula, pues muchas veces esta primera palabra era algún indicador, tal como un inciso, entonces debía ser removida hasta encontrar la primera palabra en mayúscula. Luego, todas las oraciones se convierten en minúsculas.

Finalmente, se guardan todas las oraciones pre procesadas en un archivo de texto por cada lengua dentro de la carpeta “sentences”. También se guardan todas las palabras únicas para cada lengua en la carpeta “words” y todos los caracteres distintos que se han encontrado en cada lengua en la carpeta “abecedary”. Además, todos estos datos también se guardarán en una base de datos, la cual será introducida más adelante. Por otro lado, también se guardó un archivo csv con detalles de cada lengua como la cantidad de oraciones, cantidad de palabras y cantidad de caracteres distintos encontrados.

En la Ilustración 2 se muestra un diagrama de los pasos seguidos para la limpieza de los textos, así como las herramientas usadas y las consideraciones que se tomaron. Los datos de entrada son los textos recolectados en cada lengua y la salida son las oraciones, palabras y caracteres distintos en lenguas originarias peruanas que se han encontrado en estos textos.

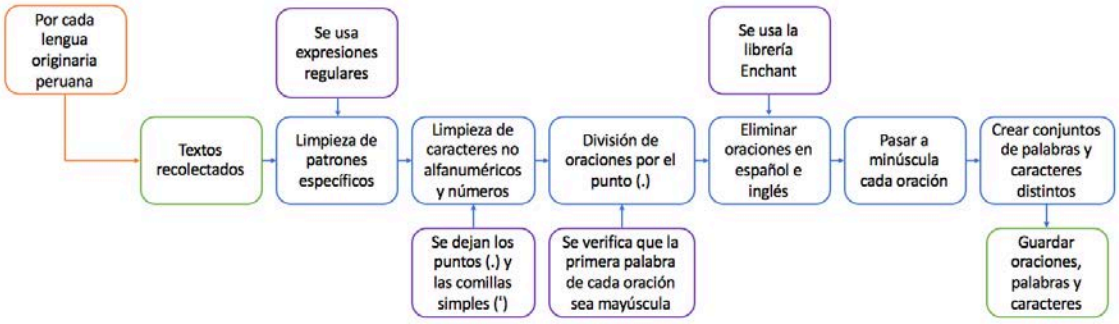


Ilustración 2. Diagrama de las fases de la funcionalidad de limpieza de textos.

Capítulo 5. Base de datos de lenguas originarias peruanas

5.1 Introducción

En este capítulo, se presentará el desarrollo del resultado esperado 2. De esta manera, luego del pre procesamiento de los textos mencionado en el capítulo anterior, se consiguió una base de datos de oraciones, palabras y caracteres. Esta base de datos será posteriormente usada para la tarea de la implementación de los modelos de identificación automática del lenguaje.

5.2 Descripción del resultado

Se diseña un esquema de la base de datos donde se guardarán las oraciones, palabras y caracteres obtenidas anteriormente y se presenta información sobre ella, además se realizó una exploración de los datos.

5.3 Desarrollo del resultado

5.3.1 Esquema de la base de datos

Para este proyecto se utilizó una base de datos documental. En la Ilustración 3 se muestra el esquema de la base de datos usada. Las oraciones, palabras y caracteres son diferenciadas de cada lengua a través del código iso-639-3. Además, se tiene información sobre la familia lingüística a la que pertenece cada lengua. El campo *family_order* de la tabla *Languages* es usado para especificar el orden en el que aparecerán las lenguas al mostrar los reportes. Por ejemplo, es de interés que ciertas lenguas aparezcan contiguas como en el caso de los dialectos del Asháninka que pertenecen a la familia Arawak. En este caso, los valores de este campo para los dialectos del Asháninka deberían ser números adyacentes para que en los reportes estas lenguas aparezcan una al lado de la otra.

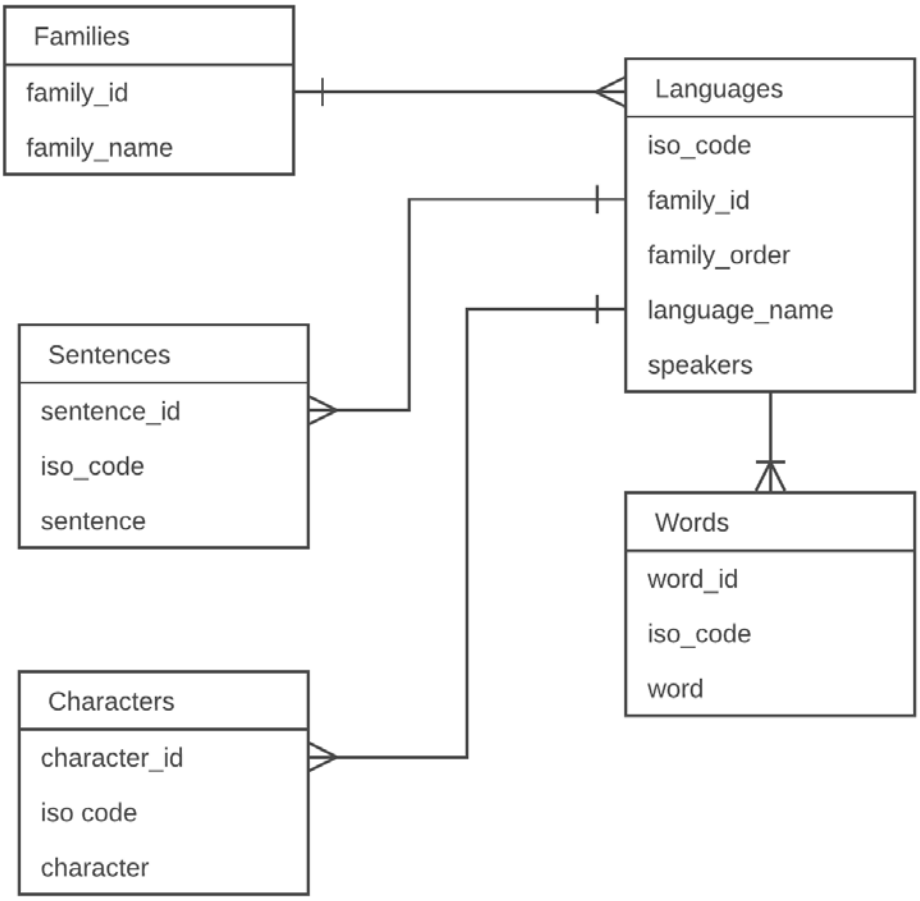


Ilustración 3. Esquema de la base de datos de lenguas originarias peruanas.

5.3.2 Implementación de la base de datos

Se levantó una instancia de base de datos en el gestor de base de datos MongoDB. Sobre esta instancia se construyó la base de datos siguiendo el esquema mostrado y se guardó todos los datos obtenidos anteriormente.

5.3.3 Información de la base de datos

En el Anexo 3 se muestra información sobre la base de datos, tales como la cantidad de archivos que se recolectó por cada lengua, el número de oraciones extraídas, la cantidad de palabras únicas, la cantidad de caracteres únicos y la cantidad de palabras en general.

5.3.4 Exploración de los datos

Se realizó una exploración de los datos a nivel de longitud de palabras y de oraciones. Esto se realizó con el fin de comparar la semejanza entre la distribución de longitudes de estos elementos contenidos en la base de datos. Para generar estos gráficos se hizo uso de la librería Seaborn.

En la Ilustración 4 se muestra la distribución de la longitud de las palabras en todas las lenguas presentes en la base de datos. Se observa que hay palabras de más de 40 caracteres pertenecientes a la lengua Kakinte (cot). En promedio, Kakinte (cot) es la lengua con las palabras más largas y Kakataibo (cbr) es la lengua con las palabras más cortas. Además, las distribuciones en algunas de las variantes de la familia quechua son bastante similares. En la Tabla 7 se muestran las palabras que poseen más de 40 caracteres, la mayoría pertenece a la lengua Kakinte (cot), mientras que una de ellas pertenece a Asheninka del Pichis (cpu). Estas palabras han sido encontradas así en su fuente original.

Tabla 7. Palabras con más de 40 caracteres de las lenguas originarias peruanas.

Lengua	**Palabra**	**Longitud**
cot	pimpishivoroquijacoshirentimentanaquempanijite	46
cot	ompitsaraquijacotimentamajatanaquempanijite	43
cot	irinetsanatashitimentajiavaquemparinijite	41
cot	anquenquetsaguishetacotsitantaquerogueti	40
cot	pintomicaguenquejanetapitsapanajanteroja	40
cpu	ipampithashiretakotapiintashivaitanakaro	40

En la Ilustración 5 se muestra la distribución de la longitud de las oraciones en todas las lenguas presentes en la base de datos. Aquí se aplicó una escala logarítmica pues había oraciones que eran muy largas. En general, la mayoría de las lenguas presenta una distribución similar respecto a la longitud de sus oraciones recolectadas. Un caso excepcional es el de Amahuaca (amc), el cual posee una distribución menos amplia, esto se puede deber a que es la lengua de la cual se encontró menos recursos.

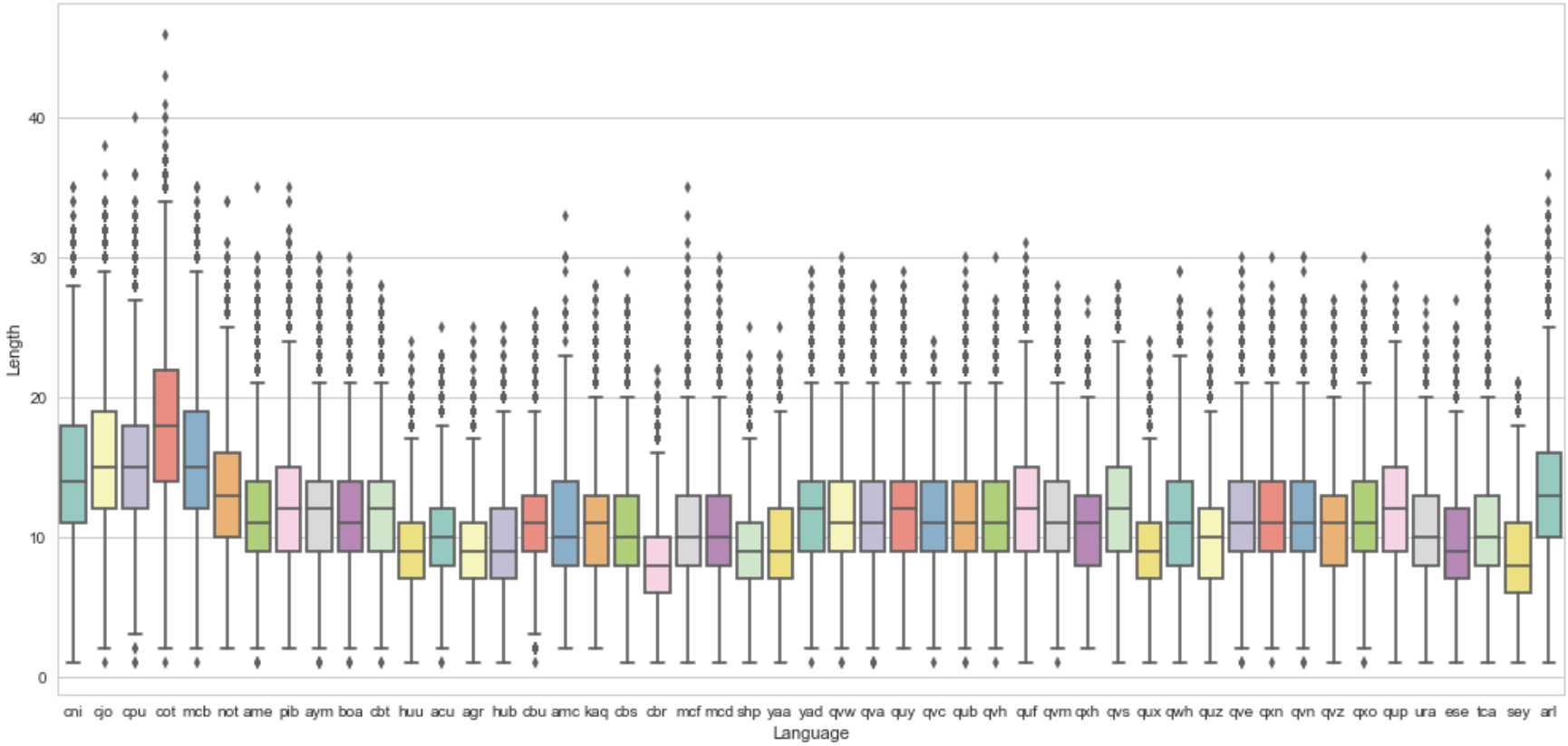


Ilustración 4. Distribución de la longitud de palabras en las lenguas originarias peruanas.

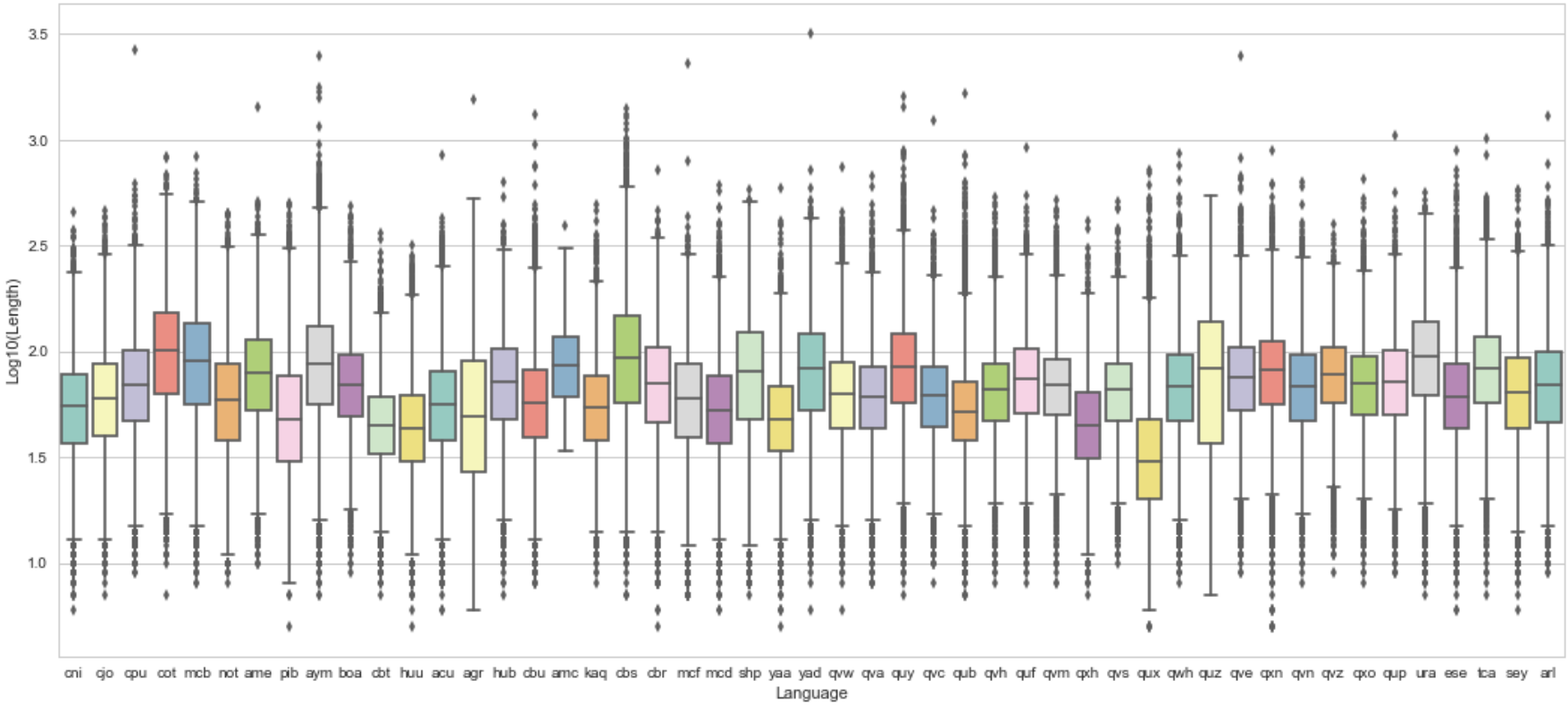


Ilustración 5. Distribución de longitud de las oraciones de las lenguas originarias peruanas.

Capítulo 6. Caracterización de las oraciones

6.1 Introducción

En este capítulo, se presentará el desarrollo del resultado esperado 3. Se busca una representación de los datos para representar a las oraciones de una manera en la que la máquina las pueda entender antes de entrenar un modelo tradicional de aprendizaje supervisado. De acuerdo a la revisión del estado del arte, una de las características más usadas para representar texto son los n-grams. Los cuales consisten en subsecuencias de “n” elementos de una secuencia dada. Por tanto, se usarán estas características para representar a las oraciones.

6.2 Descripción del resultado

Se obtiene una función para caracterizar el conjunto de datos, de manera que las oraciones sean representadas por vectores numéricos. Este conjunto de datos caracterizado estará listo para su uso en el entrenamiento del modelo tradicional de aprendizaje supervisado.

6.3 Desarrollo del resultado

6.3.1 Funcionalidad de caracterización del conjunto de datos

Para caracterizar el conjunto de datos se usó la clase TfidfVectorizer de la librería scikit-learn, esta clase permite calcular de manera automática los n-grams en un determinado rango en todo el conjunto de datos, generando una matriz de frecuencias (tf). Es decir, el valor que tendrá cada n-gram en una oración será su frecuencia de aparición. Esta clase también permite calcular la frecuencia inversa de documento (idf) de manera que el valor de cada n-gram en cada oración de la matriz resultante sea el tf-idf (*term frequency - inverse document frequency*). Los parámetros de la clase que se usaron son:

- analyzer, el tipo de analizador, el cual puede ser “char” para calcular n-grams en base a las secuencias de caracteres de las palabras o “word” para calcular los n-grams en base a la secuencia de palabras de una oración. En este caso, se uso el tipo “char”.
- ngram_range, el rango de longitudes de n-grams, debe ser una tupla que indique (“n” mínimo, “n” máximo). Por ejemplo, si se le pasa el valor (2,3), se calcularán bigrams y trigrams como características.
- use_idf, indica si se calculará la frecuencia inversa de documento.
- max_features, indica la cantidad máxima de características que se tomarán en cuenta, estas características serán las más relevantes al ordenarlas por su frecuencia de aparición en el conjunto de datos.

De esta manera, se desarrolló una función a la cual se le provean el rango de n-grams y la cantidad máxima de características y genere un objeto de la clase TfidfVectorizer que será el caracterizador de oraciones. Luego, haciendo uso de métodos propios de esta clase, se ajusta este caracterizador de acuerdo al conjunto de entrenamiento; es decir, el caracterizador calculará todos los n-grams en base a este conjunto y luego podrá buscar estos n-grams y calcular el tf-idf en cualquier nuevo conjunto que se le provea. Por otro lado, en el siguiente capítulo, se muestran los resultados obtenidos al usar diversos valores del rango de n-grams y de la cantidad máximas de características.

Capítulo 7. Modelo algorítmico tradicional entrenado

7.1 Introducción

En este capítulo, se presentará el desarrollo del resultado esperado 4. Por lo tanto, una vez se tenga las oraciones caracterizadas ya se puede entrenar el modelo, sin embargo, existen multitud de métodos de clasificación. En la revisión del estado del arte se encontró que los métodos más usados son SVM y Naive Bayes. Para este proyecto se decidió probar con estos dos métodos y algunos más. Además, dado que la caracterización de las oraciones tiene parámetros variables se probó con distintos parámetros sobre distintos métodos para elegir finalmente el mejor modelo.

7.2 Descripción del resultado

Se obtiene el mejor modelo algorítmico tradicional de aprendizaje supervisado con los mejores parámetros de caracterización para el conjunto de datos.

7.3 Desarrollo del resultado

7.3.1 Funcionalidad de partición del conjunto de datos en conjuntos de entrenamiento y prueba considerando palabras fuera de vocabulario

Para probar métodos de clasificación se suele partir los datos en conjuntos de entrenamiento y prueba. De esta manera, el método es entrenado con el conjunto de entrenamiento y se pone a prueba su rendimiento clasificando un conjunto diferente, el conjunto de prueba. Esta partición se suele hacer de manera aleatoria, sin embargo, hay casos en los que conviene probar sobre un conjunto determinado. Por ejemplo, en este caso en el que se quiere construir un identificador del lenguaje para lenguas de bajos recursos, es muy probable que gran cantidad de palabras existentes en cada lengua no esté en los datos que se pudo recolectar. Debido a esto, se quiere probar el rendimiento del modelo a construir sobre

palabras desconocidas. Por esta razón, se realizó una función que haga la partición del conjunto de datos total en conjuntos de entrenamiento y prueba considerando palabras fuera de vocabularios que el conjunto de prueba posea, por cada lengua, la mayor cantidad posible de palabras que no se encuentren en el conjunto de entrenamiento. Por tanto, se desarrolló una función cuyos parámetros son el conjunto de datos, las etiquetas o clases y el tamaño del conjunto de prueba en porcentaje.

Esta función genera un contador por cada lengua, el cual genera un diccionario en el que la llave es una palabra y el valor es la cantidad de veces que esa palabra aparece en todas las oraciones de la lengua actual. Luego, usando este diccionario, a cada oración de la lengua actual se le atribuye un peso, el cual será el promedio del valor de las palabras que la componen. Es decir, el peso será la suma del valor que cada palabra tiene en el diccionario entre la cantidad de palabras de la oración. Por tanto, las oraciones que poseen palabras menos comunes tendrán menor peso. Luego, en base al tamaño del conjunto de prueba requerido, se calcula la cantidad de oraciones de la lengua actual que deberá haber en el conjunto de prueba y se seleccionan las que tienen menor peso, el resto irá al conjunto de entrenamiento. Esto se realiza por cada lengua para obtener la partición deseada. Finalmente, retorna los conjuntos de entrenamiento y prueba.

7.3.2 Funcionalidad que prueba diversos métodos de clasificación usando validación cruzada

La librería scikit-learn incluye diversos métodos de aprendizaje supervisado, por lo tanto, se usará esta librería para probar el rendimiento de los métodos más usados según la revisión del estado del arte y algunos otros menos conocidos. De esta manera, los métodos que fueron incluídos en estas pruebas son los siguientes:

• Naive Bayes

• Support Vector Machine

• Stochastic Gradient Descent

• Perceptrón

• Passive Aggressive

Por otro lado, dado que se pretendía probar diversos métodos, se desarrolló una función que, empleando validación cruzada, prueba un método que es pasado como parámetro, sobre un conjunto de datos. La validación cruzada (*cross validation*) se usa para obtener un resultado aproximado del rendimiento de un método sobre un conjunto de datos. Para este fin se usó la función *cross_val_score* de scikit-learn, a esta función se le pasa el clasificador, el conjunto de datos, las etiquetas y el número de particiones o *folds*. Esta función hace el número de particiones indicado sobre los datos, de manera que en cada iteración una partición sea de prueba y el resto de entrenamiento, retornando finalmente los puntajes obtenidos en cada partición. Luego, estos puntajes son promediados para saber el rendimiento aproximado de cada método. En este proyecto se usó el valor de 5 particiones para realizar la validación cruzada y los resultados de las pruebas de los distintos métodos de aprendizaje supervisado se presentan en la siguiente sección.

7.3.3 Funcionalidad que prueba distintos modelos algorítmicos tradicionales

Esta funcionalidad se desarrolló para probar distintas caracterizaciones de las oraciones dados diversos parámetros, sobre los métodos tradicionales de clasificación. Esta prueba tiene el fin de encontrar el mejor modelo de caracterización y método de clasificación. Luego, este modelo será el modelo algorítmico tradicional que competirá contra el modelo de aprendizaje profundo.

La función desarrollada tiene como parámetros: el conjunto de datos, las etiquetas, una lista de los n-grams que se desea probar (se probarán combinaciones de estos n-grams) y una

lista de las máximas cantidades de características que se considerará para la caracterización, entre más alto el número más se demorará el modelo.

Para buscar los mejores parámetros no es necesario usar toda la data disponible, por tanto, se usa una partición denominada conjunto de desarrollo, la cual en este caso es una muestra del conjunto de entrenamiento.

En el Anexo 4 se muestran los resultados de probar los diferentes parámetros para la caracterización sobre 5 métodos tradicionales de aprendizaje supervisado. Se probaron n-grams con "n" entre 2 y 5 y en la cantidad de características se probaron los valores: 5000, 8000 y 10000.

Para simplificar la tabla, en la cantidad de características se puso el valor para el cual cada método obtuvo su mejor resultado, considerando el rango de n-grams correspondiente. Los valores que están en negrita son los mejores valores que se obtuvieron entre todos los métodos para cada rango de n-grams.

Se puede notar que para todos los rangos de n-grams, el método SVM con kernel linear obtuvo mejores resultados en comparación a los demás métodos. Además, el rango de n-grams en el que este método obtuvo el mejor resultado fue con n-grams entre 2 y 4 (bigrams, trigrams y tetragrams) y 10000 características, obteniendo 98.70% de precisión promedio. Esto está representado en la tabla con un sombreado de color azul. A manera de resumen se presenta la Tabla 8, donde se muestra solo los mejores resultados para cada rango de n-grams. Por lo tanto, estos serán los parámetros a usar para la caracterización de las oraciones y el método a usar será SVM con kernel linear.

Tabla 8. Mejor resultado en cada rango de n-grams al probar diversos métodos tradicionales.

"n" mínimo	**"n" máximo**	**Cantidad de características**	**Método**	**Precisión promedio**
2	2	5000	SVM (linear)	96.76

“n” mínimo	**“n” máximo**	**Cantidad de características**	**Método**	**Precisión promedio**
2	3	10000	SVM (linear)	98.47
2	**4**	**10000**	**SVM (linear)**	**98.70**
2	5	10000	SVM (linear)	98.67
3	3	10000	SVM (linear)	98.40
3	4	10000	SVM (linear)	98.59
3	5	10000	SVM (linear)	98.53
4	4	10000	SVM (linear)	98.36
4	5	10000	SVM (linear)	98.12
5	5	10000	SVM (linear)	97.32

7.3.4 Modelo algorítmico entrenado

Finalmente, se obtiene la matriz de características del conjunto de entrenamiento, considerando a los 10000 bigrams, trigrams y tetragrams más relevantes (con mayor frecuencia de aparición) como características. Luego, se entrena el clasificador con el método SVM con kernel linear sobre esta matriz de características.

Capítulo 8. Validación del modelo algorítmico tradicional

8.1 Introducción

En este capítulo, se presentará el desarrollo del resultado esperado 5. De esta manera, una vez que se tiene el modelo tradicional entrenado, este puede ser probado sobre el conjunto de prueba. Por tanto, en este capítulo se visualizará los resultados del uso del modelo algorítmico tradicional sobre el conjunto de prueba. Adicionalmente, se probó el modelo en la clasificación de palabras.

8.2 Descripción del resultado

Se obtiene el valor de diversas métricas que demuestran el rendimiento del modelo algorítmico tradicional sobre cada lengua. Además, se muestra la matriz de confusión obtenida al clasificar las oraciones del conjunto de prueba. Asimismo, se muestran los resultados del modelo al clasificar palabras.

8.3 Desarrollo del resultado

8.3.1 Reporte de clasificación de oraciones

Para obtener este reporte de clasificación se usó la función *classification_report* de la librería scikit-learn. De esta manera, en el Anexo 5 se muestra el reporte del rendimiento del modelo algorítmico tradicional sobre el conjunto de datos de prueba. Las métricas usadas son precisión, recall y f1-score. Las muestras son la cantidad de oraciones de cada lengua sobre las que se está probando el modelo algorítmico.

Además, se observa que el Quechua de Cuzco (quz) es la lengua que obtuvo los peores resultados. En esta lengua se obtiene un 81.8% de precisión y un 55% de recall. Lo cual quiere decir que de todas las oraciones que fueron clasificadas como Quechua del Cuzco, un

81.8% si pertenecían a esta lengua. Por otro lado, solo un 55% de todas las oraciones que si pertenecían al Quechua del Cuzco fueron clasificadas como tal.

Por otro lado, Amahuaca (amc), que es la lengua de la que se recolectó menos datos (solo 117 oraciones recolectadas), obtuvo 100% en las tres métricas al probarse el modelo sobre 35 oraciones de prueba para esta lengua. Esto podría significar que a pesar de tener muy pocos datos para esta lengua, es posible discriminarla correctamente de las demás lenguas. Sin embargo, también podría demostrar un sobre entrenamiento en esta lengua dado que solo se posee oraciones en un contexto: la Declaración Universal de los Derechos Humanos. En la Tabla 9 se presenta un resumen de los resultados, observándose que en promedio se obtiene un 95.9% de precisión, 95.8% de recall y 95.8% de f1-score.

Tabla 9. Resultados promedio del rendimiento del modelo tradicional al clasificar oraciones.

Precision	Recall	F1-score	Muestras
95.9	95.8	95.8	262041

8.3.2 Matriz de confusión de la clasificación de oraciones

Para obtener la matriz de confusión se usó la función *confusion_matrix* de la librería stro no era justcikit-learn. La matriz de confusión generada por esta función muestra cuántos elementos de cada clase fueron clasificados en otra clase. Luego, como se quería mostrar porcentajes, se implementó la función *normalize_confusion_matrix* que convierte estas cantidades en porcentajes. Luego usando la librería seaborn se generó un gráfico de la matriz de confusión.

De esta manera, en la Ilustración 6 se muestra la matriz de confusión del método tradicional de aprendizaje supervisado. En este gráfico, un color más oscuro quiere decir que el valor está más cerca al 100%, mientras que un color más claro indica que está más cerca al 0%.

En esta matriz de confusión los colores más oscuros están en la diagonal principal, lo cual significa que la mayoría de oraciones ha sido clasificada correctamente. Se observa además que hay una región en la que los colores están más dispersos. Esta región pertenece a la familia Quechua.

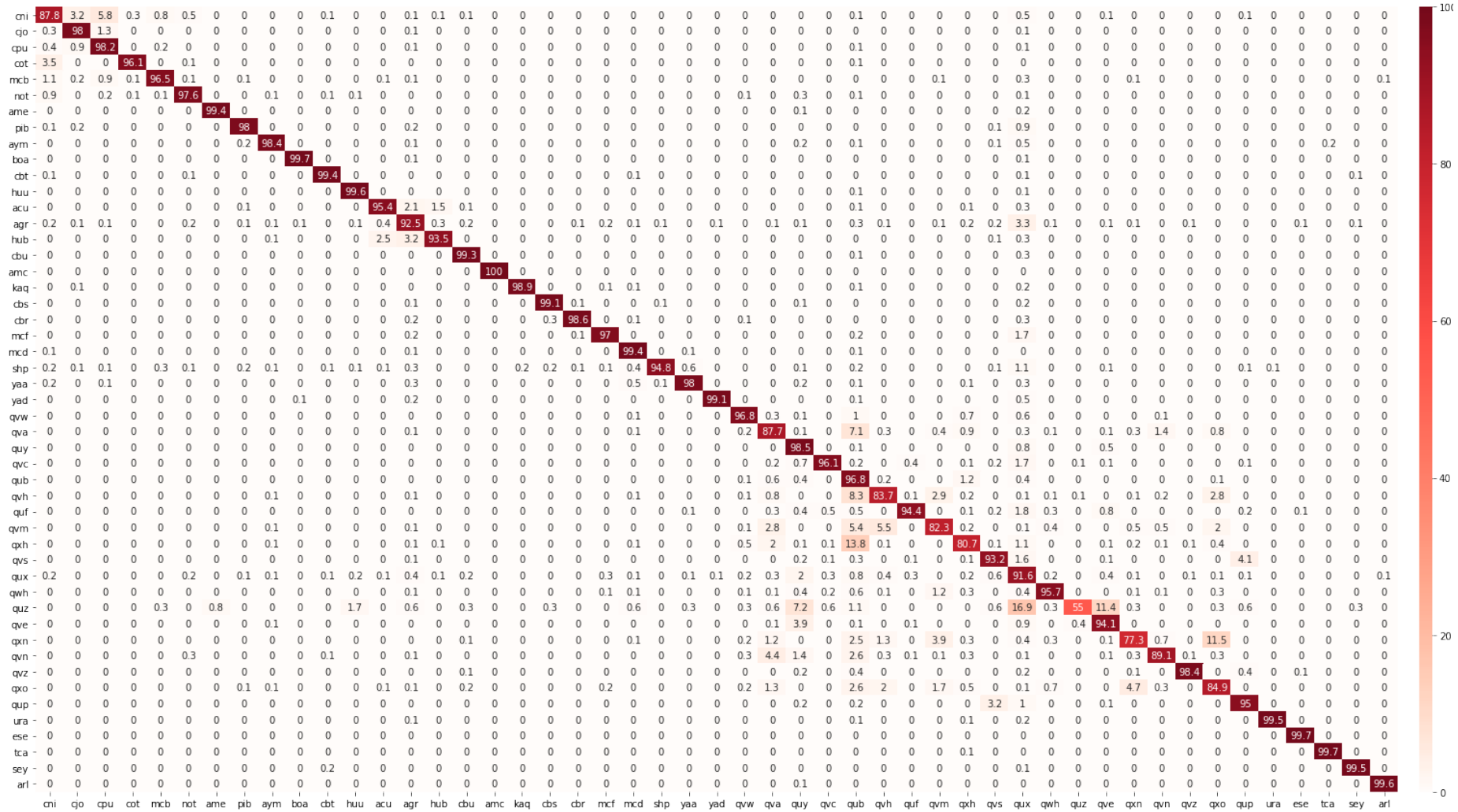


Ilustración 6. Matriz de confusión del modelo tradicional de aprendizaje supervisado en todas las lenguas.

En la Ilustración 7 se muestra la región mencionada anteriormente, es decir la matriz de confusión de la familia Quechua. Dado que los miembros de esta familia son dialectos de la lengua Quechua, tienen raíces en común y era de esperarse encontrar cierto nivel de confusión entre estas lenguas.

Se puede observar que el Quechua del Cuzco (quz) es la lengua que se ha visto más afectada, pues solo un 55% de sus oraciones de prueba fueron clasificados correctamente. Además, un 16.9% de las veces fue clasificado incorrectamente como Quechua de Yauyos (qux) y un 11.4% como Quechua del Este de Apurímac (qve).

Otro nivel considerable de confusión se ha dado entre el Quechua de Panao (qxh) y el Quechua de Huallaga (qub), dado que 13.8% de las oraciones de la primera lengua fueron clasificados como la segunda lengua. De igual manera ocurre con el Quechua del Norte de Conchucos (qxn) y el Quechua del Sur de Conchucos (qxo), donde 11.5% de las oraciones de la primera lengua fueron clasificados como la segunda lengua.

Ilustración 7. Matriz de confusión del modelo tradicional de aprendizaje supervisado en la familia Quechua.

Por otro lado, en la Ilustración 8 se muestra la matriz de confusión de las familias Arawak, Pano y Jíbaro, las cuales son otras familias aparte del Quechua que poseen más de una lengua como miembro.

En la familia Arawak, se puede observar que la mayor cantidad de confusión se acumula en las tres primeras lenguas. Estas tres lenguas son dialectos del Asháninka. La que se ve más afectada es la lengua Asháninka (cni), la cual es clasificada incorrectamente un 3.2% de las veces como Ashéninka del Pajonal (cjo) y un 5.8% como Ashéninka del Pichis (cpu).

En la familia Pano el nivel de confusión es mínimo. Entre sus miembros, la lengua que ha sido más confundida es la lengua Shipibo-konibo (shp). Esta lengua ha sido clasificada correctamente un 94.8% de las veces. Sin embargo, ha sido confundida en pequeñas proporciones con lenguas de su familia y sumando estos porcentajes no se llega al 100%, por lo tanto, al parecer ha sido confundida también con lenguas que no pertenecen a su familia.

Finalmente, la familia Jíbaro solo tiene tres miembros, de los cuales el que obtuvo la menor cantidad de oraciones clasificadas correctamente es la lengua Awajún (agr). Esta lengua ha sido clasificada correctamente un 92.5% de las veces. Sin embargo, al igual que el Shipibo-konibo, los porcentajes en los que ha sido confundido con lenguas de su familia son pequeños, por lo que se puede concluir que también ha sido confundido con lenguas fuera de su familia. Por el contrario, se puede notar que los otros dos miembros de Jíbaro han sido confundidos la mayoría de las veces con lenguas pertenecientes a su familia.

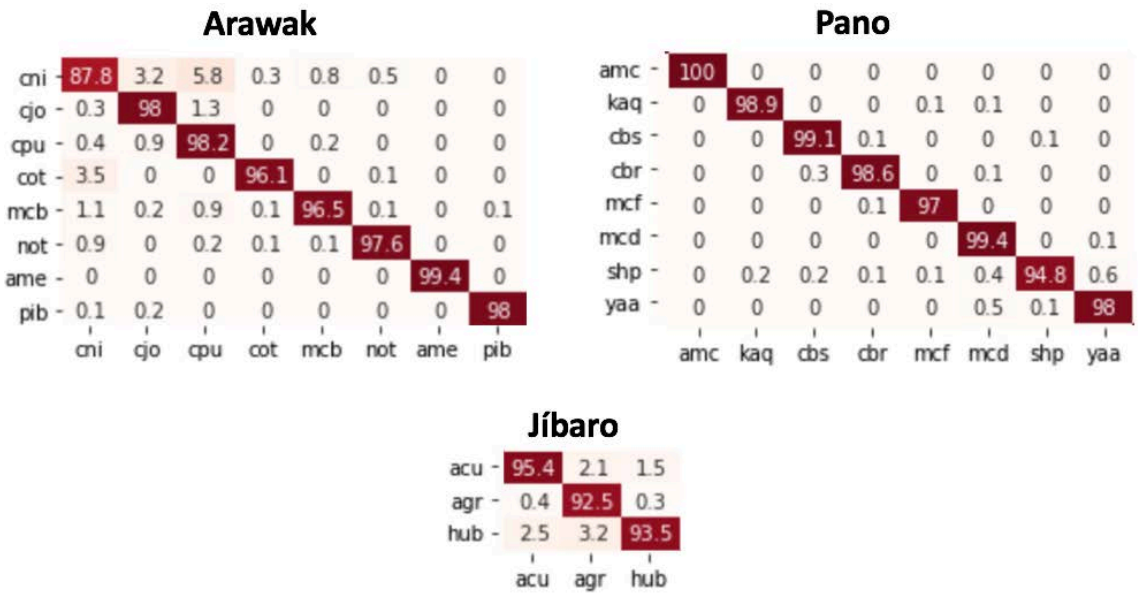


Ilustración 8. Matriz de confusión del modelo tradicional de aprendizaje supervisado en las familias Arawak, Pano y Jíbaro.

8.3.3 Rendimiento del modelo al clasificar palabras

Se probó el rendimiento del modelo al clasificar palabras. En promedio se obtuvo un 76.7% de precisión, este valor es más bajo que el obtenido al clasificar oraciones. En la Ilustración 9 se muestra la matriz de confusión del modelo tradicional de aprendizaje supervisado al clasificar palabras. Al comparar esta matriz con la obtenida en la Ilustración 6, se puede observar que en esta matriz también se obtienen precisiones altas para algunas lenguas. Sin embargo, se demuestra que la confusión entre miembros de una misma familia es más acentuada en este caso. Esto se puede deber a que hay palabras muy parecidas entre los miembros de las familias. De esta manera, se puede observar que hay confusión dentro de las familias Jíbaro, Pano y Quechua y en menor medida entre los dialectos del Asháninka. Por lo tanto, la clasificación de palabras representa un reto aún mayor dada la menor cantidad de información que se puede encontrar en una palabra en comparación a una oración.

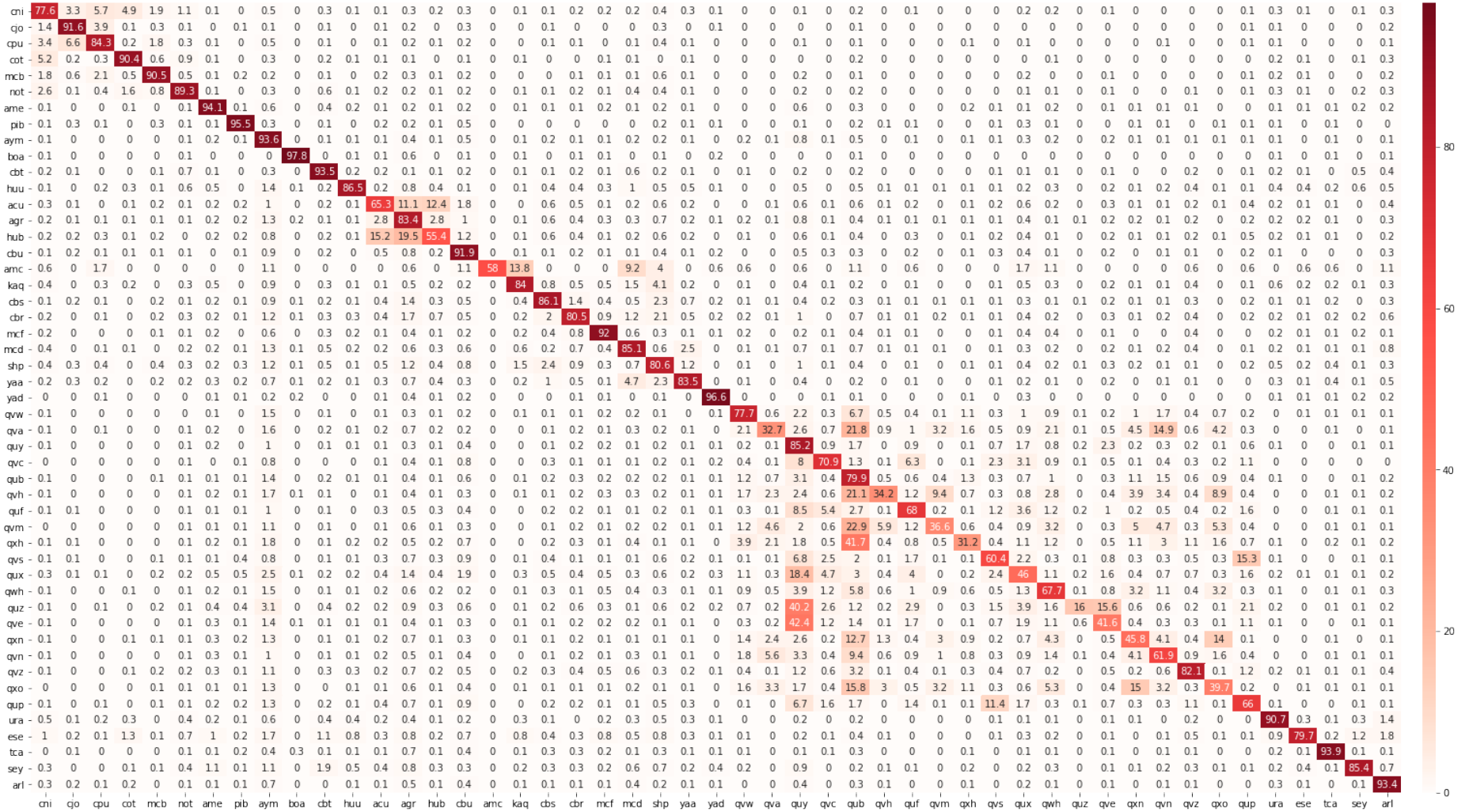


Ilustración 9. Matriz de confusión del modelo tradicional al clasificar palabras.

Capítulo 9. Representación directa en una red neuronal

9.1 Introducción

En este capítulo, se presentará el desarrollo del resultado esperado 6. En primer lugar, al entrenar un modelo algorítmico de aprendizaje profundo, no es necesario extraer características de los datos, pues la propia red neuronal profunda se encargará de reconocer y extraer estas características para la clasificación. Sin embargo, no es posible enviar directamente las oraciones a la red neuronal, sino que se debe enviar una representación numérica de estas oraciones. Por lo tanto, las oraciones serán convertidas a vectores numéricos que la computadora sea capaz de procesar.

9.2 Descripción del resultado

Se obtiene una representación de las oraciones en un vector numérico. Estos vectores estarán listos para su uso en el entrenamiento del modelo algorítmico de aprendizaje profundo.

9.3 Desarrollo del resultado

9.3.1 Construcción de un diccionario de caracteres

Primero, se construyó un diccionario con el fin de poder representar cada caracter como un número. Para esto, se extrajo todos los caracteres diferentes existentes en el conjunto de datos y a cada caracter se le asignó un número. A continuación, en la Ilustración 10 se muestra el diccionario de caracteres obtenido.

```
{' ': 1,    'l': 14,   'y': 27,   'ó': 40,
 "'": 2,    'm': 15,   'z': 28,   'õ': 41,
 'a': 3,    'n': 16,   'à': 29,   'ö': 42,
 'b': 4,    'o': 17,   'á': 30,   'ú': 43,
 'c': 5,    'p': 18,   'ã': 31,   'ü': 44,
 'd': 6,    'q': 19,   'ä': 32,   'ć': 45,
 'e': 7,    'r': 20,   'è': 33,   'ĉ': 46,
 'f': 8,    's': 21,   'é': 34,   'ī': 47,
 'g': 9,    't': 22,   'ë': 35,   'ś': 48,
 'h': 10,   'u': 23,   'ì': 36,   'ũ': 49,
 'i': 11,   'v': 24,   'í': 37,   'ű': 50,
 'j': 12,   'w': 25,   'ï': 38,   'ɨ': 51,
 'k': 13,   'x': 26,   'ñ': 39,   'ē': 52}
```

Ilustración 10. Diccionario de caracteres obtenido.

9.3.2 Conversión de las oraciones a vectores numéricos

Luego, se usa el diccionario desarrollado anteriormente para convertir cada caracter de cada oración en un número, de manera que las oraciones se conviertan en vectores numéricos. Para realizar esto, se desarrolló una función que se encargue de convertir una oración en un vector numérico usando el diccionario anteriormente mencionado. En la Ilustración 11 se muestra un ejemplo de un fragmento de una oración en Asháninka (cni), convertida en un vector numérico.

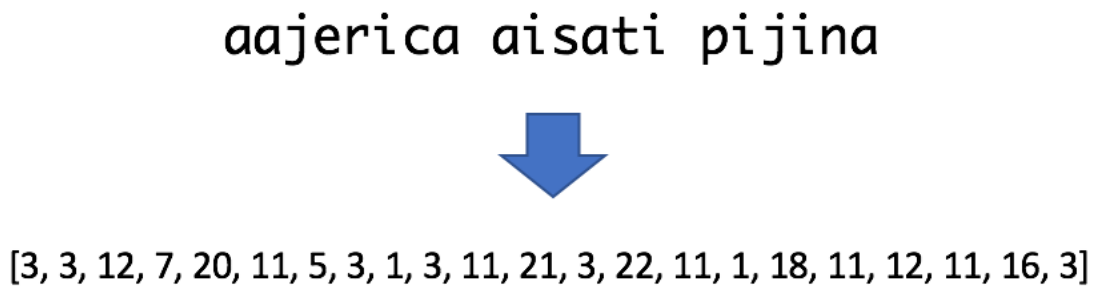


Ilustración 11. Ejemplo de conversión de una oración en un vector numérico usando el diccionario de caracteres.

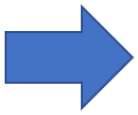
9.3.3 Truncamiento y relleno de secuencias

Los vectores numéricos obtenidos anteriormente pueden ser de distintos tamaños. Sin embargo, para poder ingresar estos vectores a la red neuronal, es necesario que estas secuencias sean de igual tamaño. Por lo tanto, se truncó las oraciones a una longitud máxima igual a 80 caracteres usando la función *pad_sequences* de la librería Keras. Se eligió este valor a partir de la exploración de los datos mostrada en la Ilustración 5, en donde se encontró que la mayoría de oraciones poseían una longitud alrededor de los 70 caracteres. En caso algunas oraciones posean una longitud menor, se rellenan las casillas restantes de “0”. De esta manera, se obtienen vectores de igual longitud para todas las oraciones. Una vez realizado esto ya se tiene las oraciones listas para el entrenamiento.

9.3.4 *One-hot encoding* para el vector objetivo

Por otro lado, para el vector objetivo se utilizó la técnica *one-hot encoding*. De esta manera, este vector se convirtió en una matriz, en la cual cada columna representa a una lengua y para cada fila se coloque un 1 en la columna que represente la lengua a la que pertenece cada oración y un 0 en las demás columnas. En la Ilustración 12 se presenta un ejemplo de convertir un vector con 6 muestras y 5 clases a una matriz utilizando one-hot encoding. Se realizó esto con el fin de que la última capa de la red neuronal asigne una probabilidad para cada clase, es decir, para los datos de entrenamiento se tendrá una probabilidad de 100% en la clase a la que pertenece y 0% en las demás clases.

Y
cni
shp
arl
quz
ese
quz



arl	cni	ese	shp	quz
0	1	0	0	0
0	0	0	1	0
1	0	0	0	0
0	0	0	0	1
0	0	1	0	0
0	0	0	0	1

Ilustración 12. Ejemplo de realizar one-hot encoding sobre un vector objetivo.

Capítulo 10. Modelo algorítmico de aprendizaje profundo

10.1 Introducción

En este capítulo, se presentará el desarrollo del resultado esperado 7. Por lo tanto, una vez se tenga la representación, descrita en el capítulo anterior, de las oraciones y el vector objetivo, se puede entrenar el modelo algorítmico de aprendizaje profundo. De acuerdo a la revisión del estado del arte, las redes neuronales recurrentes con capas del tipo *Long Short Term Memory* (LSTM) fueron el método más usado. Por lo tanto, se usó este tipo de redes neuronales y se ajustaron diversos parámetros para obtener el modelo que posea mejores resultados.

10.2 Descripción del resultado

Se obtiene una red neuronal recurrente con capas LSTM con los mejores parámetros e hiperparámetros encontrados para el dataset.

10.3 Desarrollo del resultado

10.3.1 Arquitectura de la red neuronal

Para construir una red neuronal recurrente se usó la librería Keras sobre un *backend* de la librería Tensorflow. Es decir, Keras funciona como un framework para simplificar el uso de Tensorflow, de esta manera se puede construir una red neuronal y especificar la arquitectura que esta tendrá de manera sencilla.

En la Ilustración 13 se muestra la arquitectura que tendrá la red neuronal. Primero, se tiene una capa *embedding*, esta capa será entrenada para representar cada caracter en un vector, de manera que se obtenga una matriz de tamaño igual al número de caracteres por el tamaño del *embedding*, el cual es un parámetro a ajustar. Luego, hay una capa LSTM, la cual tendrá una cantidad de unidades ajustable y 20% de *dropout*, es decir cada unidad de la capa

tendrá 20% de probabilidad de ser eliminada en cada iteración. Esto se usa para evitar el sobreajuste y reducir la dependencia de ciertas unidades. La capa LSTM en realidad consta de múltiples capas recurrentes; es decir, cada salida depende de las salidas que se obtuvieron anteriormente. Finalmente, se tendrá una capa densa de salida, la cual constará de 49 unidades, es decir la cantidad de lenguas con las que se está trabajando. De esta manera, la última capa devolverá un valor de la probabilidad de que una oración pertenezca a cada lengua. A partir de esta arquitectura luego se ajustarán los parámetros necesarios.

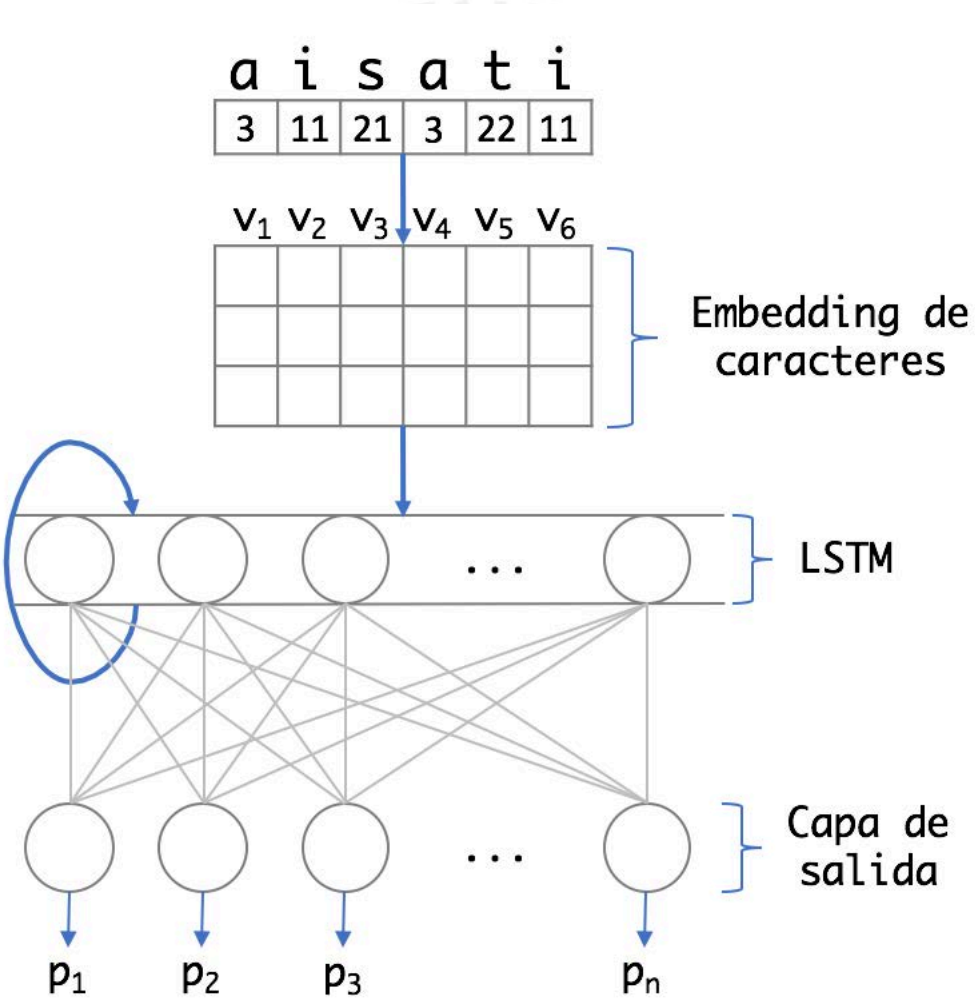


Ilustración 13. Arquitectura de la red neuronal.

10.3.2 Entrenamiento de la red neuronal

Una vez se tiene la arquitectura de la red neuronal, los pesos de las conexiones de la red deben ser ajustados en base a un conjunto de entrenamiento. Para realizar la partición del conjunto de datos total en conjuntos de entrenamiento y prueba se consideró, al igual que para el modelo tradicional, las palabras fuera del vocabulario (OOV) descritas en el capítulo 7.

Por otro lado, para entrenar el modelo también se debe ajustar ciertos parámetros como el tamaño del lote y la cantidad de épocas. El primero se refiere a la cantidad de registros que serán procesados concurrentemente, esta cantidad puede afectar el aprendizaje de la red neuronal. El segundo parámetro se refiere a la cantidad de veces que se dará un pase completo por todos los datos de entrenamiento, entre mayor sea la cantidad, la red neuronal se ajustará más a los datos de entrenamiento, pero podría haber sobre ajuste.

10.3.3 Ajuste de parámetros

Dado que había distintos parámetros que debían ser ajustados, se desarrolló una función a la que se le pasan todos los parámetros a probar, construye distintos modelos, escribe los resultados en un archivo de log y grafica la curva de aprendizaje de cada modelo. Se realizaron dos experimentos para encontrar el mejor modelo. Para realizar estos experimentos no se usó todo el conjunto de datos, sino que se usó una muestra del total de los datos denominada conjunto de desarrollo.

En el primer experimento, los parámetros que fueron probados fueron: el tamaño del *embedding*, la cantidad de unidades en la capa LSTM y el tamaño del lote. Además, se usaron 20 épocas para todos los modelos.

Para elegir la cantidad de épocas a usar, primero se probó un modelo con los parámetros por defecto de la librería Keras y una pequeña porción del conjunto de datos, es

decir, la cantidad de épocas consideradas inicialmente fue 10. Sin embargo, al analizar la curva de aprendizaje se encontró que, si se aumentaba la cantidad de épocas, los resultados podían mejorar. De esta manera, se probó con 15 y luego con 20 épocas, obteniéndose mejores resultados con 20 épocas, pero el proceso empezaba a volverse más lento. Por lo tanto, a pesar que la cantidad de épocas es otro parámetro a ser ajustado, se vio conveniente dejar el ajuste de este parámetro para después y usar 20 épocas para el primer experimento.

De esta manera, en la Tabla *10* se muestran los resultados obtenidos en el primer experimento, de estos resultados se seleccionaron las 4 mejores combinaciones, las cuales están marcadas en negrita.

Tabla 10. Resultados del primer experimento de ajuste de parámetros de la red neuronal recurrente.

Tamaño del *embedding*	**Unidades de la capa LSTM**	**Tamaño del lote**	**Precisión en los datos de entrenamiento**	**Precisión en los datos de validación**
32	128	32	0.9719	0.9302
32	**128**	**64**	**0.9723**	**0.9309**
32	128	128	0.9705	0.9296
32	128	256	0.9680	0.9247
32	128	512	0.9650	0.9226
32	256	32	0.9812	0.9404
32	**256**	**64**	**0.9833**	**0.9435**
32	256	128	0.9844	0.9419
32	256	256	0.9848	0.9416
32	256	512	0.9831	0.9389
64	128	32	0.9750	0.9336
64	128	64	0.9761	0.9341
64	**128**	**128**	**0.9753**	**0.9343**
64	128	256	0.9739	0.9318
64	128	512	0.9707	0.9260
64	256	32	0.9816	0.9396
64	256	64	0.9846	0.9423
64	256	128	0.9866	0.9425
64	**256**	**256**	**0.9870**	**0.9444**
64	256	512	0.9864	0.9427

Luego, se realizó un segundo experimento, en el cual la cantidad de épocas fue aumentada hasta 40. Este valor fue elegido considerando un tiempo prudente que se podría esperar para entrenar un solo modelo con los recursos computacionales disponibles, de esta manera, entrenar cada modelo demoraría alrededor de 1 día y medio. A continuación, se probó las 4 mejores combinaciones mencionadas anteriormente con 40 épocas. Con esta cantidad de épocas, algunos de los modelos llegaron a realizar un sobre ajuste a los datos de entrenamiento, con lo cual la precisión en los datos de prueba comenzó a disminuir. Por lo tanto, en la Tabla *11* se muestra los resultados obtenidos en este experimento, indicando en que época se obtuvo el mejor resultado y la precisión en los datos de entrenamiento y en los datos de validación en esta época. Se resalta el mejor resultado en negrita y los parámetros de este resultado serán usados en el modelo algorítmico de aprendizaje profundo.

Tabla 11. Resultados del segundo experimento de ajuste de parámetros de la red neuronal recurrente.

Tamaño del embedding	Unidades de la capa LSTM	Tamaño del lote	Número de épocas	Precisión en los datos de entrenamiento	Precisión en los datos de validación
32	128	64	38	0.9779	0.9427
32	**256**	**64**	**40**	**0.9877**	**0.9523**
64	128	128	40	0.9813	0.9427
64	256	256	33	0.9896	0.9519

10.3.4 Modelo algorítmico entrenado

Finalmente, se entrena el modelo algorítmico de aprendizaje profundo sobre el conjunto de entrenamiento con la arquitectura mencionada anteriormente y usando los parámetros del mejor resultado de los experimentos. En la Ilustración 14 se muestra la curva de aprendizaje obtenida al entrenar el modelo de aprendizaje profundo con los mejores parámetros seleccionados. La curva azul representa las precisiones obtenidas en el conjunto de entrenamiento y la curva roja representa las precisiones obtenidas en el conjunto de validación a lo largo de las 40 épocas. Se puede observar que el máximo valor se consigue en

la última época, lo cual podría significar que, si se entrenara el modelo con mayor cantidad de épocas, se podría obtener mejores resultados. Sin embargo, dadas las limitaciones de los recursos computacionales que se poseen, en este proyecto no se incrementó más la cantidad de épocas.

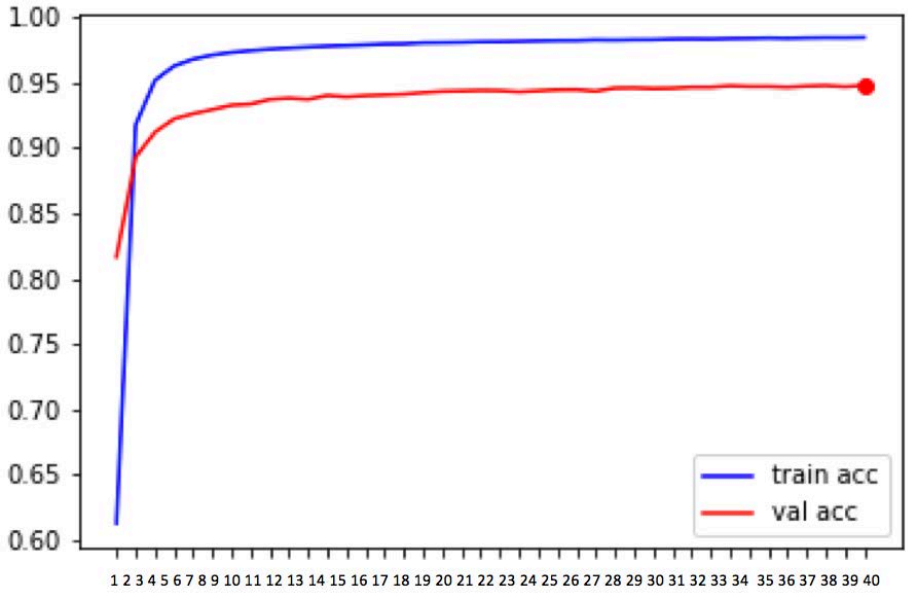


Ilustración 14. Curva de aprendizaje del modelo de aprendizaje profundo.

Capítulo 11. Validación del modelo algorítmico de aprendizaje profundo

11.1 Introducción

En este capítulo, se presentará el desarrollo del resultado esperado 8. De esta manera, una vez que se tiene el modelo de aprendizaje profundo entrenado, este puede ser usado para clasificar el conjunto de prueba y validar los resultados obtenidos. Por tanto, en este capítulo se visualizará los resultados del uso del modelo algorítmico de aprendizaje profundo sobre el conjunto de prueba.

11.2 Descripción del resultado

Se obtiene el valor de diversas métricas que demuestran el rendimiento del modelo algorítmico de aprendizaje profundo sobre cada lengua. Además, se muestra la matriz de confusión obtenida al clasificar las oraciones del conjunto de prueba.

11.3 Desarrollo del resultado

11.3.1 Reporte de clasificación de oraciones

Al igual que con el modelo tradicional, para obtener este reporte de clasificación se usó la función *classification_report* de la librería *scikit-learn*. De esta manera, en el Anexo 6 se muestra el reporte del rendimiento del modelo algorítmico de aprendizaje profundo sobre el conjunto de datos de prueba. Las métricas usadas son precisión, *recall* y *f1-score*. Las muestras son la cantidad de oraciones de cada lengua sobre las que se está probando el modelo algorítmico.

Por otro lado, a diferencia del modelo algorítmico tradicional, se observa que el Quechua de Yauyos (qux) es la lengua que obtuvo los peores resultados. En esta lengua se obtiene un 65.3% de precisión y un 90.2% de recall. Lo cual quiere decir que, de todas las

oraciones que fueron clasificadas como Quechua de Yauyos, solo un 65.3% sí pertenecían a esta lengua. Sin embargo, el recall obtenido es aceptable, ya que un 90.2% de todas las oraciones que si pertenecían al Quechua del Yauyos fueron clasificadas como tal.

En este caso, el Quechua de Cuzco (quz), que es la lengua que obtuvo los peores resultados con el modelo algorítmico tradicional (81.8% de precisión y 55% de recall), obtuvo 70.7% de precisión y 40.5% de recall con el modelo algorítmico de aprendizaje profundo, lo cual demuestra que este último modelo también obtuvo resultados negativos en la clasificación de esta lengua.

Además, Amahuaca (amc), que es la lengua de la que se recolectó menos datos (solo 117 oraciones recolectadas), obtuvo 100% en precisión y 94.3% en recall al probarse el modelo sobre 35 oraciones de prueba para esta lengua.

En la Tabla 12 se presenta un resumen de los resultados, observándose que en promedio se obtiene un 94.8% de precisión, 94.6% de recall y 94.6% de f1-score.

Tabla 12. Resultados promedio del rendimiento del modelo de aprendizaje profundo al clasificar oraciones.

Precision	Recall	F1-score	Muestras
94.8	94.6	94.6	262041

11.3.2 Matriz de confusión de la clasificación de oraciones

De igual manera que con el modelo tradicional, se usó la función *confusion_matrix* de la librería scikit-learn para obtener la matriz de confusión. La matriz de confusión generada por esta función muestra cuántos elementos de cada clase fueron clasificados en otra clase. Luego, como se quería mostrar porcentajes, se usó la función *normalize_confusion_matrix* implementada anteriormente que convierte estas cantidades en porcentajes. Luego usando la librería seaborn se generó un gráfico de la matriz de confusión.

De esta manera, en la Ilustración 15 se muestra la matriz de confusión del modelo algorítmico de aprendizaje profundo. En este gráfico, un color más oscuro quiere decir que el valor está más cerca al 100%, mientras que un color más claro indica que está más cerca al 0%.

Al igual que en el modelo tradicional, en esta matriz de confusión los colores más oscuros están en la diagonal principal, lo cual significa que la mayoría de oraciones ha sido clasificada correctamente. Se observa además que hay una región en la que los colores están más dispersos. Esta región pertenece a la familia Quechua.

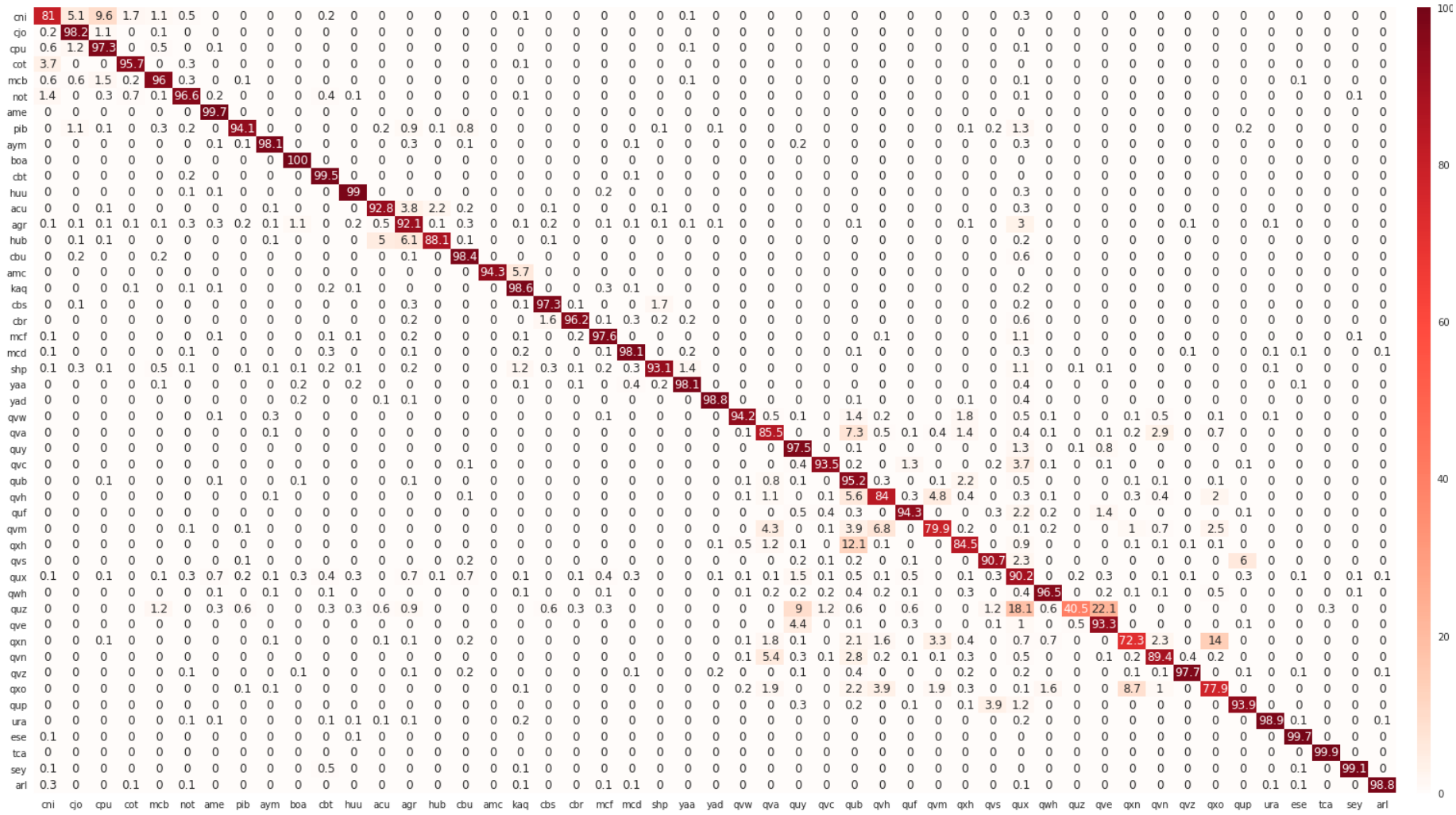


Ilustración 15. Matriz de confusión del modelo de aprendizaje profundo en todas las lenguas.

En la Ilustración 16 se muestra la región mencionada anteriormente, es decir la matriz de confusión de la familia Quechua. Dado que los miembros de esta familia son dialectos de la lengua Quechua, tienen raíces en común y al igual que con el modelo tradicional, era de esperarse encontrar cierto nivel de confusión entre estas lenguas.

Se puede observar que el Quechua del Cuzco (quz) es la lengua que se ha visto más afectada, pues solo un 40.5% de sus oraciones de prueba fueron clasificados correctamente. Además, un 22.1% de las veces fue clasificado incorrectamente como Quechua del Este de Apurímac (qve), un 18.1% como Quechua de Yauyos (qux) y un 9% como Quechua de Ayacucho (quy).

Otro nivel considerable de confusión se ha dado entre el Quechua del Norte de Conchucos (qxn) y el Quechua del Sur de Conchucos (qxo), donde 14% de las oraciones de la primera lengua fueron clasificados como la segunda lengua y un 8.7% de las oraciones de la segunda lengua fueron clasificados como la primera. De igual manera ocurre con el Quechua de Panao (qxh) y el Quechua de Huallaga (qub), dado que 12.1% de las oraciones de la primera lengua fueron clasificados como la segunda lengua; sin embargo, el caso contrario tiene un nivel de confusión bajo (2.2%).

Ilustración 16. Matriz de confusión del modelo de aprendizaje profundo en la familia Quechua.

Por otro lado, en la Ilustración 17 se muestra la matriz de confusión de las familias Arawak, Pano y Jíbaro, las cuales son otras familias aparte de la familia Quechua que poseen más de una lengua como miembro.

En la familia Arawak, se puede observar que la lengua más afecta es el Asháninka (cni), la cual es clasificada incorrectamente un 9.6% de las veces como Ashéninka del Pichis (cpu) y un 5.1% como Ashéninka del Pajonal (cjo).

En la familia Pano, el mayor nivel de confusión se ha dado en la lengua Amahuaca (amc), la cual ha sido clasificada como Capanahua (kaq) un 5.7% de las veces. Por otro lado, otra lengua que se ha visto afectada es la lengua Shipibo-konibo (shp). Esta lengua ha sido clasificada correctamente un 93.1% de las veces. Sin embargo, ha sido confundida en pequeñas proporciones con lenguas de su familia y sumando estos porcentajes no se llega al 100%, por lo tanto, al parecer ha sido confundida también con lenguas que no pertenecen a su familia.

Finalmente, la familia Jíbaro solo tiene tres miembros, de los cuales el que se ha visto más afectado es la lengua Wampis (hub), la cual ha sido clasificada correctamente un 88.1% de las veces, un 6.1% como Awajún (agr) y un 5% como Achuar (acu). Por otro lado, la lengua Awajún (agr) ha sido clasificada correctamente un 92.1% de las veces. Sin embargo, al igual que el Shipibo-konibo, los porcentajes en los que ha sido confundido con lenguas de su familia son pequeños, por lo que se puede concluir que también ha sido confundido con lenguas fuera de su familia.

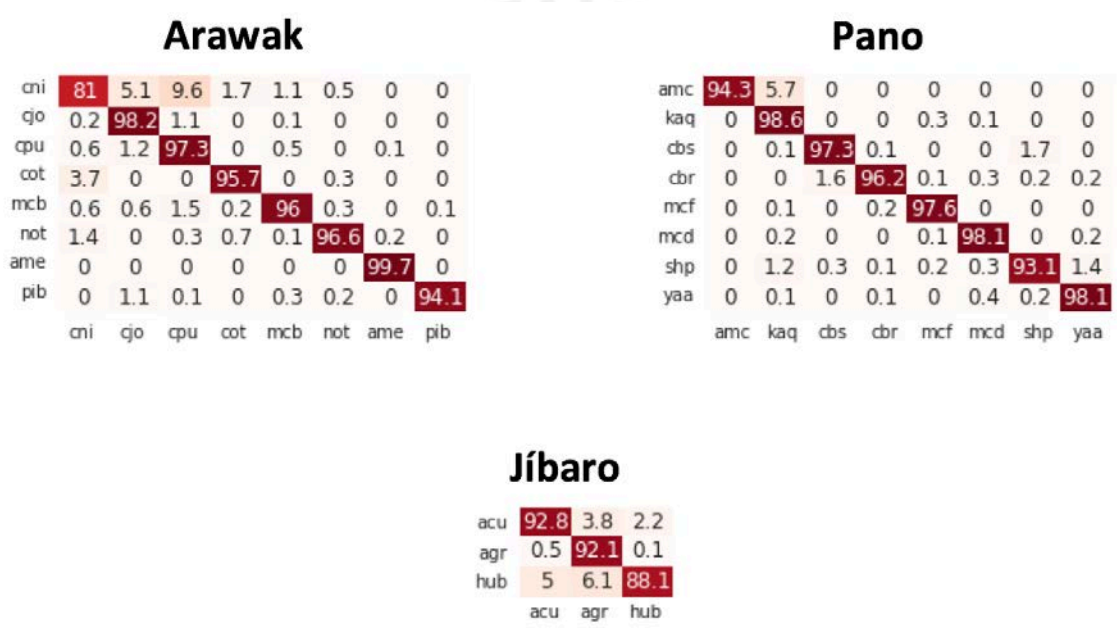


Ilustración 17. Matriz de confusión del modelo de aprendizaje profundo en las familias Arawak, Pano y Jíbaro.

Capítulo 12. Experimentación numérica

12.1 Introducción

En este capítulo, se presentará el desarrollo del resultado esperado 9. Por lo tanto, una vez se tiene el modelo algorítmico tradicional de aprendizaje supervisado y el modelo algorítmico de aprendizaje profundo, lo que sigue es compararlos y elegir el mejor modelo para que este sea usado en la aplicación web. Por lo tanto, se llevó a cabo una experimentación numérica con 30 muestras de resultados de clasificación al probar ambos modelos sobre diversas particiones del conjunto de datos. De esta manera, se podrá asegurar que las diferencias entre ambos algoritmos son estadísticamente significativas. Finalmente, se seleccionará el mejor modelo a partir del resultado de la comparación.

12.2 Descripción del resultado

Se colectan 30 muestras de la precisión obtenida por cada modelo algorítmico sobre 30 particiones aleatorias de los datos mediante la técnica *bootstrap sampling*. Finalmente, se aplican pruebas estadísticas de acuerdo al tipo de variables que se tiene para obtener el mejor modelo.

12.3 Desarrollo del resultado

12.3.1 Recolección de las muestras

Para recolectar varias muestras de la precisión obtenida por cada modelo algorítmico, se necesita tener distintos conjuntos de datos. Debido a esto, se utilizó la técnica *bootstrap sampling*, mediante la cual se obtuvo de manera aleatoria distintos conjuntos de datos de tamaño igual al 25% del total de datos. De esta manera, se obtuvieron 30 conjuntos de datos, los cuales fueron divididos en conjuntos de entrenamiento y prueba. Luego, para cada muestra, se entrenó el modelo algorítmico tradicional y el modelo algorítmico de aprendizaje

profundo con el conjunto de entrenamiento correspondiente, y ambos modelos se probaron sobre el conjunto de prueba. Dado que para esta experimentación numérica la métrica a considerar es la precisión, se obtuvo la precisión de ambos algoritmos en cada conjunto de datos. Los valores de estas precisiones obtenidas se muestran en el Anexo 7. De esta manera, las variables a comparar serán:

• X: Precisiones obtenidas por el modelo algorítmico tradicional de aprendizaje supervisado.

• Y: Precisiones obtenidas por el modelo algorítmico de aprendizaje profundo.

12.3.2 Prueba de Kolmogorov

Para realizar todas las pruebas estadísticas se usó Excel. La primera prueba a realizar es la prueba de Kolmogorov, mediante la cual se busca asegurar que ambas variables tengan una distribución normal. Por lo tanto, para la variable X se tiene las siguientes hipótesis:

• H0: X sigue una distribución normal.

• H1: X no sigue una distribución normal.

Luego, se ordenaron los valores de la variable X y se aplicó la prueba de Kolmogorov con un nivel de significancia del 5%. En la Tabla 13 se muestran los resultados de la prueba obtenidos para la variable X. Dado que la máxima diferencia (0.103) es menor que el valor crítico (0.242), se acepta la hipótesis nula (H0) y se rechaza la hipótesis alternativa (H1). Por lo tanto, la variable X sigue una distribución normal.

Tabla 13. Prueba de Kolmogorov en el modelo algorítmico tradicional de aprendizaje supervisado.

	X
Media	0.94995224
Varianza	9.75E-08
Observaciones	30
Nivel de significancia	0.05
Máxima Diferencia	0.103

Valor Crítico	0.242

El mismo procedimiento se realizó para la variable Y con las siguientes hipótesis:

- H0: Y sigue una distribución normal.
- H1: Y no sigue una distribución normal.

En la Tabla 14 se muestran los resultados de la prueba obtenidos para la variable Y. Dado que la máxima diferencia (0.107) es menor que el valor crítico (0.242), se acepta la hipótesis nula (H0) y se rechaza la hipótesis alternativa (H1). Por lo tanto, la variable Y sigue una distribución normal.

Tabla 14. Prueba de Kolmogorov en el modelo algorítmico de aprendizaje profundo.

	Y
Media	0.931669
Varianza	4.73E-07
Observaciones	30
Nivel de significancia	0.05
Máxima Diferencia	0.107
Valor Crítico	0.242

12.3.3 Prueba F

La segunda prueba a realizar es la prueba F, la cual consiste en determinar si las varianzas de ambas variables son significativamente homógeneas. En caso esto sea cierto, el siguiente paso sería aplicar una prueba Z; y en el caso contrario, el siguiente paso sería aplicar una prueba t-student para muestras con varianzas diferentes.

Por lo tanto, las hipótesis planteadas son las siguientes:

- H0: Las varianzas son significativamente homogéneas.
- H1: Las varianzas son significativamente diferentes.

En la Tabla 15 se muestra los resultados de la prueba F para comparar la varianza de ambas variables. Dado que P (2.88e-05) es menor que el nivel de significancia (0.05), se puede afirmar que rechazar la hipótesis nula (H0) es correcto ya que solo podría ser incorrecto con una probabilidad de 2.88e-05, lo cual no es relevante dado el nivel de significancia planteado. Por lo tanto, se rechaza la hipótesis nula (H0) y se acepta la hipótesis alternativa (H1). Es decir, las varianzas de ambas variables son significativamente diferentes.

Tabla 15. Prueba F para comparar la varianza de ambos modelos.

	X	*Y*
Media	0.949952243	0.931669001
Varianza	9.75202E-08	4.72608E-07
Observaciones	30	30
Grados de libertad	29	29
Nivel de significancia	0.05	
F	0.206344866	
P(F<=f) una cola	2.88263E-05	
Valor crítico para F (una cola)	0.537399965	

12.3.4 Prueba *t-student*

Debido a que las varianzas de ambas variables son significativamente diferentes, la tercera prueba a realizar es la prueba t-student para muestras con varianzas diferentes. Esta prueba, al ser realizada a dos colas consiste en determinar si hay diferencia entre las medias de ambas variables, y al ser realizada a una cola consiste en determinar qué variable tiene mayor media.

Por lo tanto, para la prueba a dos colas se tienen las siguientes hipótesis:

• H0: La media de X es igual a la de Y.

• H1: La media de X es diferente a la de Y.

Por otro lado, para la prueba a una cola se tienen las siguientes hipótesis:

• H0: La media de X es menor a la de Y.

• H1: La media de X es mayor a la de Y.

En la Tabla 16 se muestra los resultados de la prueba t-student para comparar la media de ambas variables. En la prueba a dos colas, dado que P (1.64e-54) es menor que el nivel de significancia (0.05), se puede afirmar que rechazar la hipótesis nula (H0) es correcto. Por lo tanto, se rechaza la hipótesis nula (H0) y se acepta la hipótesis alternativa (H1). Es decir, las medias de ambas variables son diferentes.

Por otro lado, en la prueba a una cola, dado que P (8.2e-55) es menor que el nivel de significancia (0.05), se puede afirmar que rechazar la hipótesis nula (H0) es correcto. Por lo tanto, se rechaza la hipótesis nula (H0) y se acepta la hipótesis alternativa (H1). Es decir, la media de X es mayor a la de Y.

Tabla 16. Prueba t-student para comparar las medias de ambos modelos.

	X	*Y*
Media	0.949952243	0.931669001
Varianza	9.75202E-08	4.72608E-07
Observaciones	30	30
Diferencia hipotética de las medias	0	
Grados de libertad	40	
Nivel de significancia	0.05	
Estadístico t	132.6257165	
P(T<=t) una cola	8.20344E-55	
Valor crítico de t (una cola)	1.683851013	
P(T<=t) dos colas	1.64069E-54	
Valor crítico de t (dos colas)	2.02107539	

12.3.5 Modelo elegido

Finalmente, dados los resultados obtenidos en la experimentación numérica, se puede afirmar que la media del modelo algorítmico tradicional de aprendizaje supervisado es

significativamente mayor a la del modelo algorítmico de aprendizaje supervisado. Por lo tanto, el modelo tradicional será usado en la aplicación web.

Capítulo 13. Aplicación web

13.1 Introducción

En este capítulo, se presentará el desarrollo del resultado esperado 10. Dado que ya se ha seleccionado el mejor modelo algorítmico entrenado para la identificación automática de lenguas originarias peruanas, el último resultado esperado es tener una aplicación web de acceso libre en la que los usuarios puedan cons1ultar a que lengua originaria peruana pertenecen sus propios textos. Por lo tanto, en este capítulo se presentará la aplicación web implementada para este fin.

13.2 Descripción del resultado

Se implementa una aplicación web que permita a los usuarios a identificar a que lengua originaria peruana pertenece algún texto, usando el mejor modelo algorítmico seleccionado a través de la experimentación numérica.

13.3 Desarrollo del resultado

13.3.1 Configuración de la base de datos

La aplicación web fue desarrollada usando el framework Django para Python. Para esta aplicación se hizo uso de una nueva base de datos dentro del proyecto. Por lo tanto, el primer paso fue definir las clases de acuerdo a las tablas que están presentes en el esquema de la base de datos y la información que se tenía tuvo que ser transferida a esta nueva base de datos.

13.3.2 Interfaz de la página de inicio

Luego, se diseñó la página de inicio, la cual se muestra en la Ilustración 18. Esta página de inicio consta de motivos peruanos y un cuadro de texto en el cual los usuarios puedan ingresar el texto que quieren averiguar a qué lengua originaria peruana pertenece. Además, se les presenta dos opciones:

• Mostrar probabilidades: Si se deselecciona esta casilla, solo se mostrará a que lengua pertenece el texto, pero no se mostrará un ranking de lenguas a las que posiblemente pertenezca con sus respectivas probabilidades. En el caso contrario, se les mostrará un ranking de las 5 lenguas más probables a las que pertenece el texto, con sus respectivas probabilidades.

• Oración por oración: Si se deselecciona esta casilla, se mostrará a que lengua posiblemente pertenece todo el texto. Por otro lado, si está seleccionada, la predicción se realizará oración por oración; es decir, se mostrará a que lengua pertenece cada oración del texto.

Ilustración 18. Página de inicio de la aplicación web.

13.3.3 Serialización del modelo algorítmico

Para realizar la tarea de identificación del lenguaje es ineficiente entrenar el modelo cada vez que se va a realizar una predicción. Por lo tanto, el modelo algorítmico fue

entrenado una vez con toda la información disponible y fue serializado en un archivo. De esta manera, cada vez que se requiera identificar a que lengua pertenece un texto, solo bastará con deserializar el objeto y realizar la identificación.

13.3.4 Identificación del lenguaje

Para realizar la identificación del lenguaje, primero se hace una limpieza del texto ingresado por el usuario para luego probarlo sobre el modelo algorítmico entrenado.

De esta manera, en la Ilustración 19 se muestra el resultado de identificar un texto en la aplicación web. En este caso las opciones por defecto estuvieron seleccionadas, por lo tanto, se muestra las probabilidades del top 5 de lenguas a las que posiblemente pertenece cada oración. Para ver los resultados de otra oración basta con hacer clic en la oración correspondiente y los datos de la derecha se actualizarán.

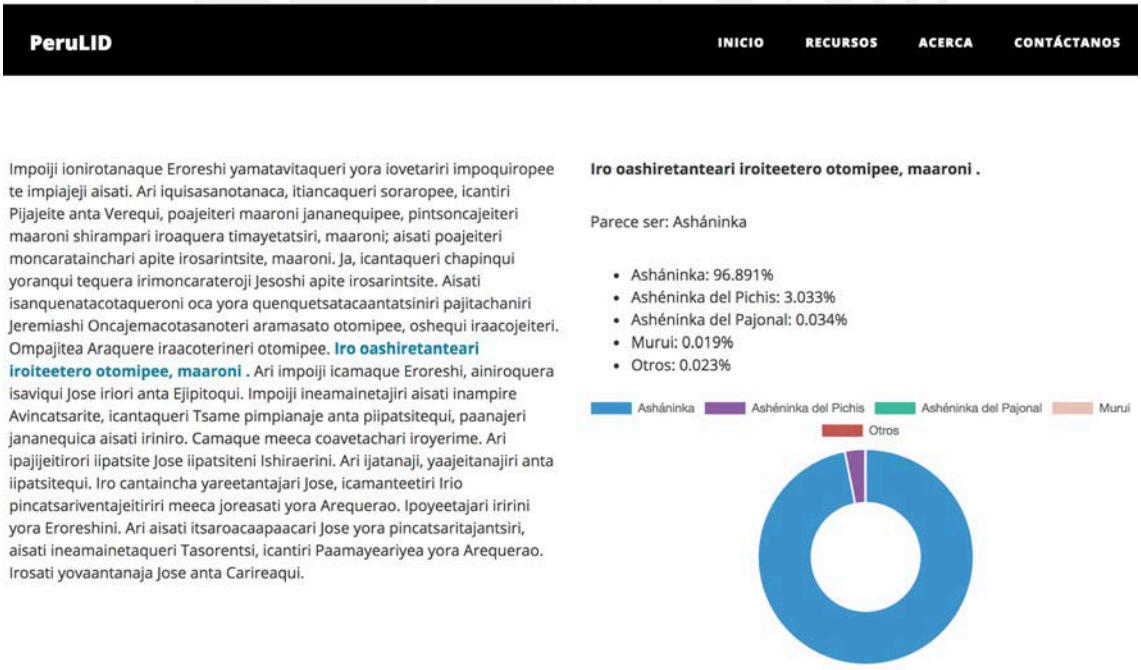


Ilustración 19. Ejemplo de la identificación del lenguaje en la aplicación web.

13.3.5 Elección del umbral de aceptación de la clasificación

Para tener mayor seguridad al clasificar textos, dado que se podría presentar lenguas diferentes a las lenguas originarias peruanas consideradas en este proyecto, se tiene un

criterio de obtener un mínimo de 70% de probabilidad para la lengua originaria peruana más probable. Si este criterio no se cumple entonces se hace uso de la librería *langdetect* de Google para identificar a que lengua pertenece el texto requerido.

Este valor de 70% se eligió tras investigar cual era la probabilidad media obtenida cuando se realizaba una predicción incorrecta con el modelo tradicional. Esto se muestra en la Ilustración 20, donde se puede ver que la mayoría de registros clasificados incorrectamente tienen una probabilidad de entre 50% y 80% con un valor medio de 65% de pertenecer a la lengua incorrecta.

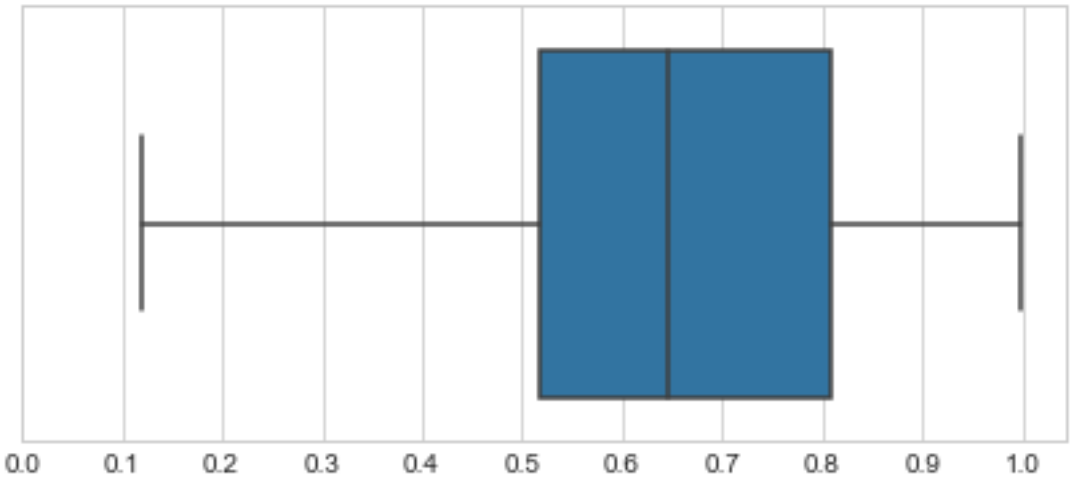


Ilustración 20. Boxplot de la distribución de probabilidades en datos clasificados incorrectamente con el modelo tradicional.

Por ejemplo, en la Ilustración 21 se muestra el resultado de usar el modelo de identificación automática del lenguaje sobre una oración en Español. De esta manera, se puede ver que la lengua originaria peruana más probable era Quechua del Este de Apurímac, sin embargo, la probabilidad de que esta oración pertenezca a esta lengua es de 51% por tanto no cumple el criterio.

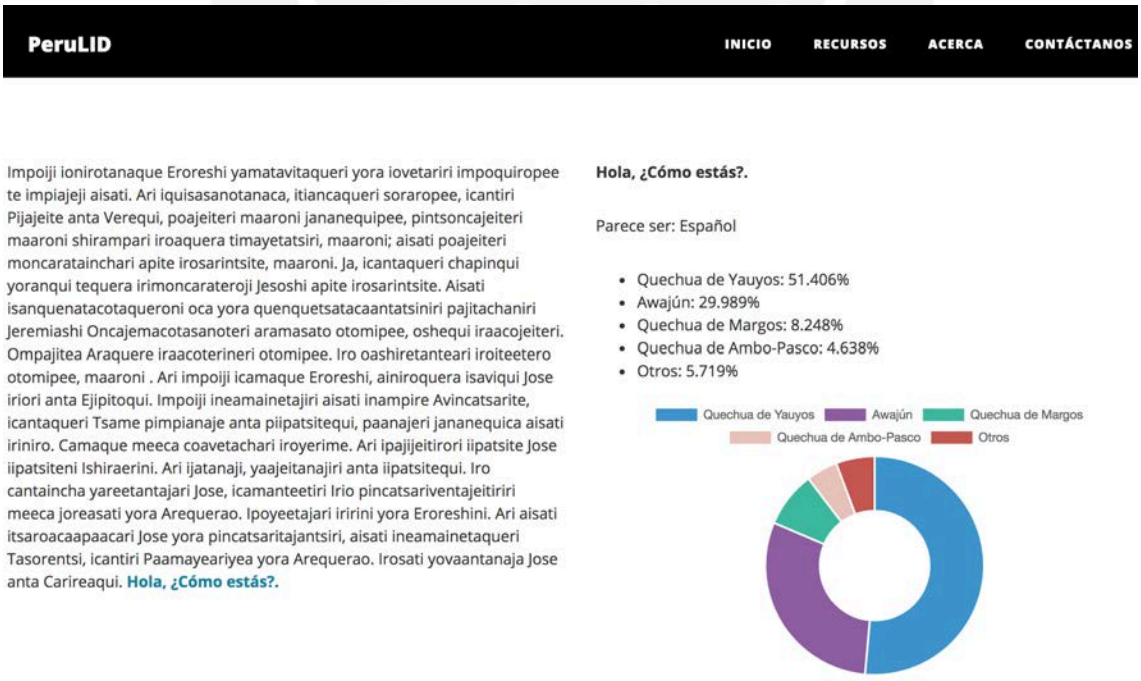


Ilustración 21. Ejemplo de la identificación del lenguaje de una oración en Español en la aplicación web.

Capítulo 14. Conclusiones y trabajos futuros

En esta sección se presentan las conclusiones del presente proyecto al haber completado todos los objetivos específicos. Asimismo, se incluyen recomendaciones con el fin de proponer mejoras a futuro para solucionar el problema planteado.

14.1 Conclusiones

En primer lugar, se construyó un repositorio de oraciones y palabras en lenguas originarias peruanas tras haber realizado una búsqueda de textos en internet. Este repositorio podría ser de mucha ayuda en otros trabajos que necesiten este tipo de información, por lo cual es considerado un resultado sumamente relevante del proyecto.

Luego, tras haber obtenido los resultados de ambos modelos algorítmicos, se concluye que las lenguas originarias peruanas pueden ser diferenciadas con una precisión de 95.9% con el modelo algorítmico de aprendizaje supervisado y con una precisión de 94.8% con el modelo algorítmico de aprendizaje profundo. Sin embargo, se encuentra cierto grado de confusión entre lenguas cercanas, es decir, entre lenguas q ue pertenecen a una misma familia lingüística, como en el caso del Quechua.

Por otro lado, tras haber realizado la experimentación numérica, se puede afirmar que el modelo algorítmico tradicional de aprendizaje supervisado usando el clasificador SVM con kernel linear obtiene una precisión significativamente mayor que el modelo algorítmico de aprendizaje profundo con capas LSTM.

Sin embargo, dadas las limitaciones de recursos computacionales que se tuvo, no fue posible probar más parámetros en la red neuronal, tales como aumentar la cantidad de épocas a usar, ya que esto hacía que entrenar el modelo demore mucho tiempo. Además, dado que la cantidad de épocas planteada (40) ya es considerado exigente para los recursos computacionales con los que se disponía y aun así no se logró obtener mejores resultados que

un método más simple como es SVM, se consideró que aumentar más este parámetro no era justificado

Finalmente, la aplicación web desarrollada podría ser de mucha ayuda para aquellas personas que estén interesadas en el estudio de las lenguas originarias peruanas y deseen saber a qué lengua pertenece un texto u oración dada. Además, podría ser usado como el primer paso en el proceso de un traductor automático, en el que inicialmente se necesita saber a qué lengua pertenece una oración para poder realizar la tarea de traducción. Asimismo, podría ser usada por cualquier persona que simplemente tenga curiosidad en estas lenguas.

14.2 Trabajos futuros

En primer lugar, dado que, al realizar la tarea de identificación del lenguaje, se encontró mayor confusión entre lenguas pertenecientes a la misma familia lingüística, se podría optar por probar un método de aprendizaje jerárquico. De esta manera, primero se identificaría a qué familia lingüística pertenece un registro dado y luego se podría hacer la clasificación del registro solo entre aquellas lenguas pertenecientes a esta familia lingüística.

Por otro lado, se puede afirmar a partir de la curva de aprendizaje obtenida por el modelo de aprendizaje profundo mostrada en la Ilustración *14* que, si el modelo se entrenará por una mayor cantidad de épocas, se podría obtener mejores resultados. Por lo tanto, si se contara con mejores recursos computacionales, se podría entrenar este modelo con una mayor cantidad de épocas y posiblemente mejorar los resultados obtenidos con este método.

Además, en este proyecto se probó una arquitectura general del modelo de aprendizaje profundo, el cual consistió en una red neuronal recurrente con capas LSTM a la cual se ajustó diversos parámetros. Sin embargo, según la revisión del estado del arte, modificaciones como primero pasar los registros por una red neuronal convolucional para luego recién pasar a una red neuronal recurrente han tenido buenos resultados. Se podría probar este enfoque, además

de probar capas tipo GRU o bidireccionales en vez de usar capas LSTM, para tratar de mejorar los resultados obtenidos por el modelo algorítmico de aprendizaje profundo.

Finalmente, el modelo desarrollado funciona para diferenciar 49 lenguas originarias peruanas, las cuales incluyen 19 dialectos del Quechua y 3 dialectos del Asháninka. Debido a esto, el modelo en realidad incluye 29 de las 47 lenguas originarias peruanas existentes. Por lo tanto, aún hay 18 lenguas originarias peruanas que no están incluidas dado que no se encontraron los recursos necesarios. Es por esto que se espera encontrar recursos de más lenguas originarias peruanas y así poder ampliar la cantidad de lenguas incluidas en el identificador automático del lenguaje desarrollado.

Referencias

Bjerva, J. (2016). Byte-based Language Identification with Deep Convolutional Networks. *arXiv preprint arXiv:1609.09004.*

Botha, G. R., & Barnard, E. (2012). Factors that affect the accuracy of text-based language identification. Computer Speech & Language, 26(5), 307-320.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 27

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078.*

Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, *37*(1), 51-89.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan), 1-30.

Fisher, R. A. (1956). Statistical methods and scientific inference.

Forcada, M. (2006). Open source machine translation: an opportunity for minor languages. In Proceedings of the Workshop "Strategies for developing machine translation for minority languages", LREC (Vol. 6, pp. 1-6).

Grothe, L., De Luca, E. W., & Nürnberger, A. (2008, May). A Comparative Study on Language Identification Methods. In LREC.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques.* Elsevier.

Heggarty, P., & Pearce, A. (Eds.). (2011). History and Language in the Andes. Springer.

Jaech, A., Mulcaire, G., Hathi, S., Ostendorf, M., & Smith, N. A. (2016). Hierarchical character-word models for language identification. *arXiv preprint arXiv:1608.03030.*

Jupyter Notebook. (2019). Project Jupyter. EU: Project Jupyter. Recuperado de https://jupyter.org/

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Kocmi, T., & Bojar, O. (2017). LanideNN: Multilingual Language Identification on Character Window. *arXiv preprint arXiv:1701.03338.*

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.

Lui, M., & Baldwin, T. (2012, July). langid. py: An off-the- shelf language identification tool. In Proceedings of the ACL 2012 system demonstrations (pp. 25-30). Association for Computational Linguistics.

Malmasi, S., & Dras, M. (2015, May). Automatic language identification for Persian and Dari texts. In Proceedings of PACLING (pp. 59-64).

Microsoft Excel (2019). Software de hojas de cáculo. EU: Microsoft. Recuperado de https://products.office.com/es/excel

Ministerio de Educación, Perú (Sin fecha), Repositorio Minedu. Lima, Perú: Repositorio Minedu. Recuperado de http://repositorio.minedu.gob.pe/

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

MongoDB (2019). *Document databases*. EU: MongoDB. Recuperado de https://www.mongodb.com/document-databases

Montgomery, D. C. (2017). *Design and analysis of experiments*. John Wiley & Sons.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Pienaar, W., Snyman, D.P., 2010. Spelling checker-based language identification for the eleven official south african languages. In: Proceedings of the Twenty-First Annual Symposium of the Pattern Recognition Association of South Africa, 22–23 November 2010, Stellenbosch, South Africa, 213–216.

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

Quasthoff U., Biemann C., and Richter M. 2006. Corpus portal for search in monolingual corpora. Computational Linguistics, 19(1), 61–74

Real Academia Española. (2016). Diccionario de la lengua española (23.ª ed.). Consultado en http://www.rae.es/rae.html

Resolución Ministerial N°629-2016- MINEDU, Lima, Perú, 14 de diciembre de 2016.

Rios, A. (2015). A Basic Language Technology Toolkit for Quechua (Doctoral dissertation, University of Zurich)

Ross, S. M. (2007). *Introducción a la Estadística*. Reverté.

Selamat, A., & Akosu, N. (2016). Word-length algorithm for language identification of under-resourced languages. Journal of King Saud University-Computer and Information Sciences, 28(4), 457-469.

Simpson, G., & Mayer-Hasselwander, H. (1986). Bootstrap sampling-Applications in gamma-ray astronomy. *Astronomy and Astrophysics*, 162, 340-348.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, *15*(1), 1929-1958.

Wilcox, R. (2005). Kolmogorov–smirnov test. *Encyclopedia of biostatistics*.

Zubiaga, A., San Vicente, I., Gamallo, P., Campos, J. R. P., Loinaz, I. A., Aranberri, N. & Fresno-Fernández, V. (2014, September). Overview of TweetLID: Tweet Language Identification at SEPLN 2014. In *TweetLID@ SEPLN* (pp. 1-11).

a

Anexo 1. Planificación de tareas.

Tarea	Fecha Inicio	Fecha Fin
Búsqueda y recolección de documentos digitales que contengan texto en lenguas originarias peruanas	09/05	09/06
Extracción de texto a partir de los documentos digitales	10/06	16/07
Desarrollar un programa que limpie, busque y extraiga las oraciones en lenguas originarias	17/07	23/07
Implementar un programa que construya vectores de características por cada lengua	24/07	30/07
Realizar pruebas de algunos modelos algorítmicos tradicionales de aprendizaje supervisado	31/07	06/08
Seleccionar el modelo algorítmico tradicional que posea los mejores resultados	07/08	13/08
Validar el modelo algorítmico tradicional para identificar el lenguaje de diversos textos de prueba	14/08	20/08
Analizar y reportar información que permita determinar la eficiencia del modelo algorítmico tradicional en la identificación automática del lenguaje	22/08	24/08
Buscar soluciones a los problemas específicos	25/08	27/08
Implementar un modelo algorítmico de aprendizaje profundo para la identificación automática del lenguaje	29/08	03/09
Validar el modelo algorítmico de aprendizaje profundo para la identificación automática de lenguaje sobre diversos textos de prueba	05/09	10/09
Analizar y reportar información que permita determinar la eficiencia del modelo algorítmico de aprendizaje profundo en la identificación automática del lenguaje	12/09	14/09
Comparar los resultados de ambos métodos y seleccionar el algoritmo a usar en el identificador automático del lenguaje para lenguas originarias peruanas	15/09	17/09
Implementar una aplicación web de acceso libre	19/09	21/09
Integrar el identificador automático del lenguaje en la aplicación web	22/09	24/09
Realizar un proceso de indexación automático de los términos del repositorio	26/09	28/09
Implementar un buscador de palabras en base al repositorio de textos	29/09	01/10

Anexo 2. Fuentes de recolección de textos en lenguas originarias peruanas.

Lengua	Cantidad	Archivo	Fuente
Achuar	3	acuNT.txt	https://www.bible.com/en-GB/bible/4/JHN.1.acunt
		00-OTacu-web.pdf	http://www.scriptureearth.org/data/acu/PDF/00-OTacu-web.pdf
		acuUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=acu
Amahuaca	1	amcUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=amc
Arabela	3	arlNT.txt	https://www.bible.com/en-GB/bible/519/jhn.1.arlnt
		00-POTarl-web.pdf	http://www.scriptureearth.org/data/arl/PDF/00-POTarl-web.pdf
		arlUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=arl
Asháninka	6	Dic_Prelim_Ashaninka.pdf	http://www.lengamer.org/publicaciones/diccionarios/Dic_Prelim_Ashaninka.pdf
		mihas_2011_anaani.pdf	http://biblio.wdfiles.com/local--files/mihas-2011-anaani/mihas_2011_anaani.pdf
		mihas_2014_diccionario.pdf	http://etnolinguistica.wdfiles.com/local--files/biblio%3Amihas-2014-diccionario/mihas_2014_diccionario.pdf
		RELATOS DE NOPOKI 1_ashaninka.pdf	UCSS - RELATOS DE NOPOKI 1.pdf pp. 14-51
		cniNT.txt	https://www.bible.com/en-GB/bible/575/JHN.1.cnint
		cniUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=cni
Ashéninka del Pajonal	2	RELATOS DE NOPOKI 1_asheninka.pdf	UCSS - RELATOS DE NOPOKI 1.pdf pp. 76-132
		cjoNT.txt	https://www.bible.com/en-GB/bible/568/JHN.1.cjont
Ashéninka del Pichis	2	cpuNT.txt	https://www.bible.com/en-GB/bible/238/MAT.1.cpuNT
		cpuUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=cpu
Awajún	7	Awajunti Nugkutai.pdf	http://www.lengamer.org/admin/language_folders/awajun/user_uploaded_files/links/File/Awajunti%20Nugkutai.pdf
		CUENTO DE SUMPA Y CHIACHIA.pdf	http://www.lengamer.org/admin/language_folders/awajun/user_uploaded_files/links/File/CUENTO%20DE%20SUMPA%20Y%20CHIACHIA.pdf
		Diccionario Awajun-Castellano.pdf	https://docs.google.com/file/d/0BxBLALMTzXlkdzJqNnVaSlZuSVU/edit
		Dic_Prelim_Awajun.pdf	http://www.lengamer.org/publicaciones/diccionarios/Dic_Prelim_Awajun.pdf
		Uchi yakiya dapijai.pdf	http://www.lengamer.org/admin/language_folders/awajun/user_uploaded_files/links/File/Uchi%20yakiya%20dapijai.pdf
		agrNT.txt	https://www.bible.com/en-GB/bible/515/JHN.1.agrnt
		agrUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=agr
Aymara	5	aymNT.txt	https://www.bible.com/en-GB/bible/293/gen.1.aymnt
		kimsaqalqu.pdf	http://intercultural.k12.cl/icore/downloadcore/158121

Lengua	Cantidad	Archivo	Fuente
		MiVozNuestroHistoria-Cuentos.pdf	http://www.illa-a.org/cd/literatura/MiVozNuestroHistoria-Cuentos.pdf
		cuentos_aymaras.pdf	http://biblioteca.serindigena.org/libros_digitales/cuentos_aymara/cuentos_aymaras.pdf
		aymUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=aym
Bora	2	boaNT.txt	https://www.bible.com/en-GB/bible/578/jhn.1.boant
		boaUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=boa
Capanahua	1	kaqNT.txt	https://www.bible.com/en-GB/bible/586/hn.1.kaqnt
Cashinahua	3	RELATOS DE NOPOKI 1_cashunahua.pdf	UCSS - RELATOS DE NOPOKI 1.pdf pp. 186-190
		cbsNT.txt	https://www.bible.com/en-GB/bible/540/JHN.1.cbsnt
		cbsUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=cbs
Ese eja	1	eseNT.txt	https://www.bible.com/en-GB/bible/576/jhn.1.esent
Kakataibo	196	Todos los demás txt	Roberto Zariquiey
		cbrNT.txt	https://www.bible.com/en-GB/bible/541/GEN.1.cbrNT
		cbrUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=cbr
Kakinte	2	cotNT.txt	https://www.bible.com/en-GB/bible/566/JHN.1.cotnt
		cotUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=cot
Kandozi	2	cbuNT.txt	https://www.bible.com/en-GB/bible/213/JHN.1.cbuNT
		cbuUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=cbu
Matsigenka	2	RELATOS MATSIGENKA.pdf	UCSS
		mcbNT.txt	https://www.bible.com/en-GB/bible/633/JHN.1.mcbnt
Matsés	4	Diccionario Matsés (2012) Fleck, Uaqui & Jiménez.pdf	http://repositorio.pucp.edu.pe/index/handle/123456789/49614
		Historia de los Matsés (2014) - Jiménez, Jiménez & Fleck.pdf	http://repositorio.pucp.edu.pe/index/bitstream/handle/123456789/50089/Historia%20de%20los%20Mats%C3%A9s%20%282014%29%20-%20Jim%C3%A9nez%2c%20Jim%C3%A9nez%20%26%20Fleck.pdf?sequence=1&isAllowed=y
		mcfNT.txt	https://www.bible.com/en-GB/bible/631/JHN.1.mcfnt
		mcfUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=mcf
Murui	2	huuNT.txt	https://www.bible.com/en-GB/bible/596/JHN.1.huunt
		huuUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=huu
Nomatsigenga	2	notNT.txt	https://www.bible.com/en-GB/bible/647/JHN.1.notnt
		notUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=not
Quechua Huaylla Wanca	7	awilaOC.pdf	http://www.lengamer.org/admin/language_folders/quechuawanca/user_uploaded_files/links/File/awilaOC.pdf

d

Lengua	Cantidad	Archivo	Fuente
		culturamachu.pdf	http://www.lengamer.org/admin/language_folders/quechuawanca/user_uploaded_files/links/File/culturamachu.pdf
		culturaOC.pdf	http://www.lengamer.org/admin/language_folders/quechuawanca/user_uploaded_files/links/File/culturaOC.pdf
		HB BUENO publicado.pdf	http://www.lengamer.org/admin/language_folders/quechuawanca/user_uploaded_files/links/File/HB%20BUENO%20publicado.pdf
		ninezamachu.pdf	http://www.lengamer.org/admin/language_folders/quechuawanca/user_uploaded_files/links/File/ninezamachu.pdf
		Poesias bien.pdf	http://www.lengamer.org/admin/language_folders/quechuawanca/user_uploaded_files/links/File/Poesias%20%20bien.pdf
		qvwNT.txt	https://www.bible.com/en-GB/bible/267/JHN.1.qvwnt
Quechua de Ambo-Pasco	2	qvaNT.txt	https://www.bible.com/en-GB/bible/1769/MAT.1.qvant
		qvaUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=qeg
Quechua de Ayacucho	2	quyNT.txt	https://www.bible.com/en-GB/bible/556/MAT.1.quynt
		quyUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=quy
Quechua de Cajamarca	2	qvcNT.txt	https://www.bible.com/en-GB/bible/648/JHN.1.qvcnt
		qvcUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=qnt
Quechua de Huallaga	1	qubNT.txt	https://www.bible.com/en-GB/bible/1088/JHN.1.qubnt
Quechua de Huamalíes - Dos de Mayo	2	qvhNT.txt	https://www.bible.com/en-GB/bible/670/JHN.1.qvhnt
		qvhUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=qej
Quechua de Lambayeque	9	adivinanza.pdf	http://lengamer.org/admin/language_folders/quechuadelambayeque/user_uploaded_files/links/File/adivinanza.pdf
		apuntes_de_semantica.pdf	http://lengamer.org/admin/language_folders/quechuadelambayeque/user_uploaded_files/links/File/apuntes_de_semantica.pdf
		killata.pdf	http://lengamer.org/admin/language_folders/quechuadelambayeque/user_uploaded_files/links/File/killata.pdf
		kunya.pdf	http://lengamer.org/admin/language_folders/quechuadelambayeque/user_uploaded_files/links/File/kunya.pdf
		la ONU Q.pdf	http://lengamer.org/admin/language_folders/quechuadelambayeque/user_uploaded_files/links/File/la%20ONU%20Q..pdf
		paramuta_qatichanapaq.pdf	http://lengamer.org/admin/language_folders/quechuadelambayeque/user_uploaded_files/links/File/paramuta_qatichanapaq.pdf
		TEXTO_PARA_I_NIVEL.pdf	http://lengamer.org/admin/language_folders/quechuadelambayeque/user_uploaded_files/links/File/TEXTO_PARA_I_NIVEL.pdf
		traduc. quijote quf.pdf	http://lengamer.org/admin/language_folders/quechuadelambayeque/user_uploaded_files/links/File/traduc.%20quijote%20quf.pdf
		qufNT.txt	https://www.bible.com/en-GB/bible/650/JHN.1.qufnt
Quechua de Margos Yarowilca Lauricocha	2	qvmNT.txt	https://www.bible.com/en-GB/bible/265/JHN.1.qvmnt
		qvmUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=qei
Quechua de Panao	1	qxhNT.txt	https://www.bible.com/en-GB/bible/457/JHN.1.qxhnt

d

e

Lengua	Cantidad	Archivo	Fuente
Quechua de San Martin	1	qvsNT.txt	https://www.bible.com/en-GB/bible/266/JHN.1.qvsnt
Quechua de Yauyos	2	SYQ_Lexicon Yauyos to Spanish.pdf	http://repositorio.pucp.edu.pe/index/bitstream/handle/123456789/49426/SYQ_Lexicon%20Yauyos%20to%20Spanish.pdf?sequence=7&isAllowed=y
		Transcriptions_ Compilation.pdf	http://repositorio.pucp.edu.pe/index/handle/123456789/49435
Quechua del Callejón de Huaylas	5	AprestamientoQuechua.pdf	http://www.lengamer.org/admin/language_folders/quechuahuaylas/user_uploaded_files/links/File/AprestamientoQuechua.pdf
		EnsayotextoQuechua.pdf	http://www.lengamer.org/admin/language_folders/quechuahuaylas/user_uploaded_files/links/File/EnsayotextoQuechua.pdf
		LectoEscritura.pdf	http://www.lengamer.org/admin/language_folders/quechuahuaylas/user_uploaded_files/links/File/LectoEscritura.pdf
		qwhNT.txt	https://www.bible.com/en-GB/bible/734/JHN.1.qwhnt
		qwhUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=qan
Quechua del Cusco	2	cuentos.pdf	http://lengamer.org/admin/language_folders/quechuadecusco/user_uploaded_files/links/File/cuentos.pdf
		teqse_quechua.pdf	http://lengamer.org/admin/language_folders/quechuadecusco/user_uploaded_files/links/File/Qhapaqkuna/teqse_quechua.pdf
		quzUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=quz
Quechua del Este de Apurímac	10	bodajosemarian.pdf	http://www.lengamer.org/admin/language_folders/quechuaesteapurimac/user_uploaded_files/links/File/bodajosemarian.pdf
		costumbres.pdf	http://www.lengamer.org/admin/language_folders/quechuaesteapurimac/user_uploaded_files/links/File/costumbres.pdf
		Cuentoanciana.pdf	http://www.lengamer.org/admin/language_folders/quechuaesteapurimac/user_uploaded_files/links/File/Cuentoanciana.pdf
		cuentosdeabuelo.pdf	http://www.lengamer.org/admin/language_folders/quechuaesteapurimac/user_uploaded_files/links/File/cuentosdeabuelo.pdf
		Hobredesconocido.pdf	http://www.lengamer.org/admin/language_folders/quechuaesteapurimac/user_uploaded_files/links/File/Hobredesconocido.pdf
		hukuchamantaatoqmantawan.pdf	http://www.lengamer.org/admin/language_folders/quechuaesteapurimac/user_uploaded_files/links/File/hukuchamantaatoqmantawan.pdf
		laninez.pdf	http://www.lengamer.org/admin/language_folders/quechuaesteapurimac/user_uploaded_files/links/File/laninez.pdf
		Q'enqomanta willakuy.pdf	http://www.lengamer.org/admin/language_folders/quechuaesteapurimac/user_uploaded_files/links/File/Q'enqomanta%20willakuy.pdf
		Takakuqch'akumantan.pdf	http://www.lengamer.org/admin/language_folders/quechuaesteapurimac/user_uploaded_files/links/File/Takakuqch'akumantan.pdf
		qveNT.txt	https://www.bible.com/en-GB/bible/667/JHN.1.qvent

e

f

Lengua	Cantidad	Archivo	Fuente
Quechua del Norte de Conchucos	1	qxnNT.txt	https://www.bible.com/en-GB/bible/268/JHN.1.qxnnt
Quechua del Norte de Junín	2	qvnNT.txt	https://www.bible.com/en-GB/bible/663/JHN.1.qvnnt
		qvnUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=qju
Quechua del Norte de Pastaza	1	qvzNT.txt	https://www.bible.com/en-GB/bible/718/JHN.1.qvznt
Quechua del Sur de Conchucos	1	qxoNT.txt	https://www.bible.com/en-GB/bible/290/JHN.1.qxont
Quechua del Sur de Pastaza	1	qupNT.txt	https://www.bible.com/en-GB/bible/645/JHN.1.qupnt
Secoya	1	seyNT.txt	https://www.bible.com/en-GB/bible/664/JHN.1.seynt
Sharanahua	2	mcdNT.txt	https://www.bible.com/en-GB/bible/522/JHN.1.mcdnt
		mcdUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=mcd
Shawi	2	cbtNT.txt	https://www.bible.com/en-GB/bible/544/JHN.1.cbtnt
		cbtUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=cbt
Shipibo-konibo	4	educativoNL_Ship.txt	
		legalNL_Ship.txt	
		religiosoNL_Ship.txt	
		RELATOS DE NOPOKI 1_shipibo.pdf	UCSS - RELATOS DE NOPOKI 1.pdf pp. 136-183
Tikuna	1	tcaNT.txt	https://www.bible.com/en-GB/bible/289/JHN.1.tcant
Urarina	2	uraNT.txt	https://www.bible.com/en-GB/bible/692/JHN.1.urant
		uraUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=ura
Wampis	1	hubNT.txt	https://www.bible.com/en-GB/bible/720/JHN.1.hubnt
Yagua	2	yadNT.txt	https://www.bible.com/en-GB/bible/680/JHN.1.yadnt
		yadUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=yad
Yaminahua	1	yaaNT.txt	https://www.bible.com/en-GB/bible/679/JHN.1.yaant
Yanesha	2	ameNT.txt	https://www.bible.com/en-GB/bible/512/JHN.1.ament
		ameUDHR.txt	http://www.ohchr.org/EN/UDHR/Pages/Language.aspx?LangID=ame
Yine	2	RELATOS DE NOPOKI 1_yine.pdf	UCSS - RELATOS DE NOPOKI 1.pdf pp. 54-73
		pibNT.txt	https://www.bible.com/en-GB/bible/652/JHN.1.pibnt

f

g

Anexo 3. Información de la base de datos.

iso-639-3	# archivos	# oraciones	# palabras	abecedario	# tokens
acu	4	24665	22863	35	209591
amc	1	117	564	26	1169
arl	4	30105	27172	34	263706
cni	5	22114	29467	32	127783
cjo	3	20458	26820	31	129564
cpu	2	13226	25013	33	96491
agr	7	25085	42947	34	214540
aym	6	40371	93025	40	451854
boa	3	13845	19670	31	123627
kaq	2	14923	14286	33	125780
cbs	4	8911	17712	34	148483
ese	1	20844	6447	25	218504
cbr	197	13738	15360	47	184336
cot	3	14774	32837	32	130384
cbu	3	41761	39206	34	446037
mcf	4	21955	21957	36	186694
mcb	2	14288	27700	33	136444
huu	1	20227	9923	33	149297
not	2	16920	19518	29	123237
qvw	7	9386	22100	42	73168
qva	2	14191	21078	40	107647
quy	2	38254	69911	35	389233
qvc	1	19314	22669	28	149582
qub	1	53474	58253	40	362412
qvh	2	11429	18796	40	93905
quf	9	15384	22796	36	141420
qvm	2	12150	18528	40	102395
qxh	1	11429	14081	38	66283
qvs	1	17879	18184	28	134258
qux	2	12471	23411	37	64820
qwh	5	12866	26850	40	112653
quz	3	1200	5936	36	13571
qve	10	16559	26226	35	151340
qxn	1	12264	25434	39	118143
qvn	2	13156	24236	39	111908
qvz	1	12869	16478	34	128359
qxo	1	14638	24856	34	121974
qup	1	16639	18725	32	146833

g

h

iso-639-3	# archivos	# oraciones	# palabras	abecedario	# tokens
sey	1	14635	10384	44	149584
mcd	2	25228	21361	34	228443
cbt	2	29584	21623	31	166353
shp	4	12008	19652	35	170258
tca	1	15060	15654	65	228158
ura	2	12953	16185	28	170730
hub	1	13286	20940	34	140165
yad	2	12402	20809	43	129238
yaa	1	19704	15616	37	164712
ame	2	18102	25352	46	199319
pib	2	14706	21610	23	107054

h

Anexo 4. Resultados de probar diferentes parámetros de caracterización sobre métodos tradicionales de clasificación

“n” mínimo	“n” máximo	Cantidad de características	Método	Precisión promedio
2	2	5000	SVM (linear)	**96.76**
2	2	8000	SGD	94.51
2	2	5000	Perceptron	94.89
2	2	8000	Passive Aggresive	96.24
2	2	5000	Naive Bayes	91.61
2	3	10000	SVM (linear)	**98.47**
2	3	10000	SGD	96.76
2	3	8000	Perceptron	97.50
2	3	10000	Passive Aggresive	98.30
2	3	5000	Naive Bayes	95.96
2	4	10000	SVM (linear)	**98.70**
2	4	10000	SGD	97.06
2	4	10000	Perceptron	97.89
2	4	10000	Passive Aggresive	98.61
2	4	8000	Naive Bayes	96.70
2	5	10000	SVM (linear)	**98.67**
2	5	10000	SGD	96.94
2	5	8000	Perceptron	97.73
2	5	10000	Passive Aggresive	98.53
2	5	10000	Naive Bayes	96.60
3	3	10000	SVM (linear)	**98.40**
3	3	10000	SGD	96.85
3	3	10000	Perceptron	97.27
3	3	8000	Passive Aggresive	98.20
3	3	5000	Naive Bayes	96.12
3	4	10000	SVM (linear)	**98.59**
3	4	10000	SGD	96.96
3	4	10000	Perceptron	97.61

j

“n” mínimo	**“n” máximo**	**Cantidad de características**	**Método**	**Precisión promedio**
3	4	10000	Passive Aggresive	98.45
3	4	8000	Naive Bayes	96.65
3	5	10000	SVM (linear)	**98.53**
3	5	10000	SGD	96.73
3	5	10000	Perceptron	97.55
3	5	10000	Passive Aggresive	98.43
3	5	10000	Naive Bayes	96.53
4	4	10000	SVM (linear)	**98.36**
4	4	10000	SGD	96.72
4	4	10000	Perceptron	97.31
4	4	10000	Passive Aggresive	98.21
4	4	10000	Naive Bayes	96.29
4	5	10000	SVM (linear)	**98.12**
4	5	10000	SGD	96.08
4	5	10000	Perceptron	96.91
4	5	10000	Passive Aggresive	97.94
4	5	10000	Naive Bayes	95.92
5	5	10000	SVM (linear)	**97.32**
5	5	10000	SGD	95.23
5	5	10000	Perceptron	95.85
5	5	10000	Passive Aggresive	97.05
5	5	10000	Naive Bayes	95.05

j

k

Anexo 5. Reporte de clasificación de oraciones por cada lengua con el modelo algorítmico tradicional.

Lengua	Precision	Recall	F1-score	Muestras
cni	94.1	87.8	90.9	6617
cjo	95.2	98.0	96.6	6136
cpu	87.8	98.2	92.7	3959
cot	99.2	96.1	97.6	4430
mcb	97.8	96.5	97.2	4277
not	97.9	97.6	97.8	5050
ame	99.6	99.4	99.5	5235
pib	98.5	98.0	98.3	4393
aym	99.6	98.4	99.0	12053
boa	99.6	99.7	99.7	4152
cbt	99.4	99.4	99.4	8870
huu	99.3	99.6	99.4	6059
acu	97.6	95.4	96.5	7317
agr	93.7	92.5	93.1	7454
hub	96.0	93.5	94.7	3955
cbu	99.4	99.3	99.3	12523
amc	100.0	100.0	100.0	35
kaq	99.4	98.9	99.2	4449
cbs	98.9	99.1	99.0	2631
cbr	99.3	98.6	99.0	4019
mcf	99.1	97.0	98.1	6526
mcd	98.5	99.4	98.9	7567
shp	99.5	94.8	97.1	3599
yaa	99.2	98.0	98.6	5810
yad	99.4	99.1	99.2	3553
qvw	96.6	96.8	96.7	2759
qva	85.3	87.7	86.5	4215
quy	94.8	98.5	96.6	11451
qvc	99.0	96.1	97.5	5778

k

l

Lengua	Precision	Recall	F1-score	Muestras
qub	89.3	96.8	92.9	15982
qvh	86.8	83.7	85.2	3404
quf	98.6	94.4	96.5	4605
qvm	87.7	82.3	84.9	3612
qxh	87.7	80.7	84.1	3412
qvs	95.1	93.2	94.1	5328
qux	70.5	91.6	79.7	3506
qwh	97.1	95.7	96.4	3822
quz	81.8	55.0	65.8	321
qve	95.8	94.1	94.9	4951
qxn	90.3	77.3	83.3	3591
qvn	96.0	89.1	92.4	3923
qvz	99.2	98.4	98.8	3851
qxo	84.3	84.9	84.6	4364
qup	94.4	95.0	94.7	4975
ura	99.6	99.5	99.6	3878
ese	99.7	99.7	99.7	6042
tca	99.4	99.7	99.6	4498
sey	99.6	99.5	99.5	4090
arl	99.7	99.6	99.7	9014
avg / total	**95.9**	**95.8**	**95.8**	**262041**

m

Anexo 6. Reporte de clasificación de oraciones por cada lengua con el modelo algorítmico de aprendizaje profundo.

Lengua	Precision	Recall	F1-score	Muestras
cni	93.6	81.0	86.9	6617
cjo	92.0	98.2	95.0	6136
cpu	81.8	97.3	88.9	3959
cot	95.8	95.7	95.7	4430
mcb	95.8	96.0	97.9	4277
not	96.9	96.6	96.8	5050
ame	97.2	99.7	98.4	5235
pib	98.3	94.1	96.2	4393
aym	99.4	98.1	98.8	12053
boa	96.8	100.0	98.3	4152
cbt	98.5	99.5	99.0	8870
huu	98.7	99.0	98.9	6059
acu	96.2	92.8	94.4	7317
agr	90.5	92.1	91.3	7454
hub	95.1	88.1	91.5	3955
cbu	98.6	98.4	98.5	12523
amc	100.0	94.3	97.1	35
kaq	97.0	98.6	97.8	4449
cbs	96.0	97.3	96.6	2631
cbr	99.1	96.2	97.6	4019
mcf	98.5	97.6	98.0	6526
mcd	98.4	98.1	98.3	7567
shp	97.0	93.1	95.0	3599
yaa	98.2	98.1	98.1	5810
yad	99.0	98.8	98.9	3553
qvw	97.5	94.2	95.8	2759
qva	82.6	85.5	84.0	4215
quy	96.1	97.5	96.8	11451
qvc	99.0	93.5	96.2	5778

m

n

Lengua	Precision	Recall	F1-score	Muestras
qub	90.9	95.2	93.0	15982
qvh	83.1	84.0	83.6	3404
quf	96.6	94.3	95.4	4605
qvm	87.8	79.9	83.7	3612
qxh	83.3	84.5	83.9	3412
qvs	94.8	90.7	92.7	5328
qux	65.3	90.2	75.7	3506
qwh	96.2	96.5	96.3	3822
quz	70.7	40.5	51.5	321
qve	94.5	93.3	93.9	4951
qxn	84.6	72.3	77.9	3591
qvn	91.1	89.4	90.3	3923
qvz	98.7	97.7	98.2	3851
qxo	82.2	77.9	80.0	4364
qup	92.5	93.9	93.2	4975
ura	99.1	98.9	99.0	3878
ese	99.1	99.7	99.4	6042
tca	99.8	99.9	99.9	4498
sey	99.3	99.1	99.2	4090
arl	99.6	98.8	99.2	9014
avg / total	**94.8**	**94.6**	**94.6**	**262041**

n

o

Anexo 7. Precisiones obtenidas por muestra por cada modelo para la experimentación numérica.

Muestra	Tradicional	RNN
1	0.950045455276	0.932316707371
2	0.950068964729	0.930563045312
3	0.949668417896	0.931808201465
4	0.950371085938	0.930583128502
5	0.949759674560	0.931362198755
6	0.950785409845	0.931680515869
7	0.950337566275	0.932105459539
8	0.949870479746	0.931054167018
9	0.949948312575	0.931337833226
10	0.949678281548	0.931019000166
11	0.949940461276	0.930751065330
12	0.949617185265	0.931593847441
13	0.950488272156	0.932764683878
14	0.949573627345	0.931205893979
15	0.950278495651	0.932686919187
16	0.949484257061	0.932043166158
17	0.949740284453	0.932052375724
18	0.949741060902	0.931706731247
19	0.950130333051	0.931984630397
20	0.950181881748	0.932824356458
21	0.950124355168	0.932149645693
22	0.949678055986	0.931827617514
23	0.949854043644	0.932234207214
24	0.949805742745	0.931947426023
25	0.949864274598	0.929904203843
26	0.949990521221	0.931878176614
27	0.949532605515	0.930916043130
28	0.949697405485	0.932139308953
29	0.950351751605	0.931763685833
30	0.949959030666	0.931865786346