PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Desarrollo de un Framework para la identificación del Nivel de Complejidad de Texto para el Entrenamiento de Chatbots basado en *Machine Learning*

Tesis para obtener el título profesional de Ingeniero Informático

AUTOR:

Hans Matos Rios

ASESOR:

Dr. César Armando Beltrán Castañón

Lima, Enero, 2022

Resumen

La generación de diálogo implica diseñar un programa para generar una conversación natural, esto requiere desarrollar algoritmos que puedan conversar con un ser humano y otro programa de manera coherente y fluida. Desarrollar una conversación también depende del contexto y conocimiento del interlocutor, lo cual nos lleva a definir la existencia de niveles de complejidad conversacional, la cual se refiere a determinar que tan fácil o difícil de leer es un texto. En este aspecto, especialmente para el idioma español, no existe una herramienta que tenga un módulo propio que le permita clasificar textos en español por su complejidad textual.

En el presente trabajo de fin de carrera se realiza el desarrollo de un módulo en el lenguaje de programación Python, el cual funciona como un *Framework* para identificar la complejidad textual de textos en español usando técnicas de *Machine Learning*. Para ello, en primer lugar, se implementaron 48 métricas de análisis de complejidad textual basadas en Coh-Metrix usando el lenguaje de programación Python. Dichas métricas convierten textos en español en datos numéricos con los cuales se entrenaron distintos modelos de *Machine Learning*, con el motivo de obtener el mejor modelo a utilizar con el *Framework* desarrollado, siendo este capaz de utilizar un modelo personalizado provisto por el usuario. Para ello, se necesitó obtener un corpus de 183 textos en español para realizar dicho entrenamiento, el cual fue obtenido al descargar textos educativos de nivel primaria y secundaria. Por último, se entrenó un *chatbot* con los textos obtenidos para el corpus, cuyas respuestas generadas fueron analizados con el *Framework* previamente desarrollado, identificando que el nivel de complejidad de dichas respuestas correspondía al nivel de los textos con los cuales el *chatbot* fue entrenado.

En conclusión, en el presente proyecto de investigación se desarrolla un módulo de Python que funciona como un Framework, el cual es capaz de identificar la complejidad textual de textos en español, ya sea con el mejor modelo de *Machine Learning* utilizado en el presente proyecto o utilizando uno provisto por el usuario de la herramienta.

Agradecimientos

Agradezco a mis compañeros con quieres compartí buenos momentos a lo largo de los semestres; a mis profesores, quienes me compartieron sus valiosos conocimientos durante todas las clases; a mi familia, quienes me motivaron a esforzarme, seguir adelante y no rendirme durante todo este tiempo; y a mi asesor, el Doctor César Armando Beltrán Castañón, por dedicar su tiempo a guiarme en la elaboración de la presente tesis.

Tema FCI

FACULTAD DE
CIENCIAS E INGENIERÍA
ESPECIALIDAD DE
INGENIERÍA INFORMÁTICA

PUCP

textual. A todo ello se debe adicionar la inexistencia de un corpus numeroso clasificado por complejidad textual para el Español, lo cual limita realizar aplicaciones en diferentes dominios, como el desarrollo de chatbots conversacionales.

El desarrollo de un chatbot implica su entrenamiento previo, para ello se requiere de textos referentes a un dominio o tópico, pero estos dominios también poseen complejidad, de la cual depende la respuesta que otorgará el chatbot. Dicha situación genera que las respuestas del chatbot podrían no estar acordes al interés del usuario, lo que lo lleva a replantear sus preguntas y así se genera confusión.

Para el presente proyecto de fin de carrera, se propone desarrollar un nuevo framework para el análisis e identificación de nivel de complejidad de documentos en Español con el fin de crear un corpus de entrenamiento para un chatbot. Para ello se usarán técnicas de machine learning a fin de predecir el nivel de complejidad, considerando como datos de entrada las diferentes métricas orientadas a medir el nivel de complejidad de un texto. Finalmente, este nuevo framework será desarrollado en Python, herramienta más usual en la comunidad de machine learning y procesamiento de lenguaje natural, de manera que esta pueda ser útil para el desarrollo de futuras aplicaciones.

OBJETIVO GENERAL

Desarrollar un framework para identificar el nivel de complejidad de documentos referentes a un tópico aplicando machine learning para generar corpus de entrenamiento para un chatbot.

OBJETIVOS ESPECÍFICOS

Los objetivos específicos son:
OE1. Implementar el módulo para calcular métricas de complejidad textual.
OE2. Constituir un corpus de más de cien textos en español clasificados por su complejidad textual.
OE3. Evaluar el impacto de los corpus generados en las respuestas de un Chatbot.

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERU
Departamento de Ingeniería
Dra. LAYLA HIRSH M.
Coordinadora de Especialidad
Ingeniería Informática

FACULTAD DE
CIENCIAS E INGENIERÍA
ESPECIALIDAD DE
INGENIERÍA INFORMÁTICA

PUCP

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO:	**Desarrollo de un framework para la identificación del nivel de complejidad de texto para el entrenamiento de chatbots basado en machine learning**
ÁREA:	Ciencias de la Computación
ASESOR:	Dr. César BELTRÁN CASTAÑÓN
ALUMNO:	Hans MATOS RIOS
CÓDIGO:	20151475
FECHA:	San Miguel, Noviembre del 2021

DESCRIPCIÓN

La generación de diálogo implica diseñar un programa para generar una conversación natural, esto requiere desarrollar algoritmos que puedan conversar con un ser humano u otro programa de manera coherente y fluida. Poder participar en un diálogo o simplemente chatear con otros es una habilidad humana natural. Ciertas personas son buenas charlando y otras se sienten atraídas por ellas; en otras palabras, las primeras pueden dirigir la conversación hacia temas que son de interés de su interlocutor, seguir un tema de conversación durante un período prolongado y agregar detalles según sea necesario.

Pero el desarrollar una conversación, también depende del contexto y conocimiento del interlocutor, lo cual nos lleva a definir la existencia de niveles de complejidad conversacional, área de investigación también conocida como análisis de discurso o complejidad textual, la cual se refiere a determinar qué tan fácil o difícil de leer es un texto. En este aspecto, especialmente para el idioma español, no existe una herramienta que tenga un módulo propio que le permita clasificar textos en español por su complejidad textual. La clasificación de textos se refiere a la tarea de asignar una categoría a un documento escrito . En cambio, para otros idiomas, sí podemos encontrar algunas herramientas que evalúan la complejidad textual, basado en ciertos criterios medibles, que se refieren a las métricas de complejidad textual, los cuales son el conjunto de modelos y fórmulas que identifican qué tan fácil o difícil de leer es un texto; y además porque existen varios conjuntos de textos preprocesados, denominados corpus, de estos idiomas.

Por ejemplo, una herramienta de análisis de complejidad textual es Coh-Metrix, el cual se usa para el idioma inglés. Esta herramienta ha sido utilizada en otros idiomas de forma limitada, como portugués, francés, ruso, entre otros; pero para el caso del idiomas español, solo una herramienta fue encontrada durante la revisión: Coh-metrix-esp, la cual es una adaptación de la herramienta Coh-metrix original que sólo implementa 45 métricas de complejidad textual de las 108 disponibles en la versión original de Coh-metrix para inglés. Este hecho genera que dicha herramienta para el idioma español no sea tan eficaz como la versión original para inglés. También, esta herramienta llamada Coh-metrix-esp fue implementada, en su momento, usando lenguaje Java, además de depender de una herramienta de terceros llamada Freeling, la cual provee servicios de análisis textual que calculan las métricas de complejidad

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Capítulo 1. Generalidades

1.1 Problemática

1.1.1 Árbol de Problemas

La Figura 1 muestra el diagrama del árbol de problemas en el cual se aprecian tres causas del problema principal y tres efectos.

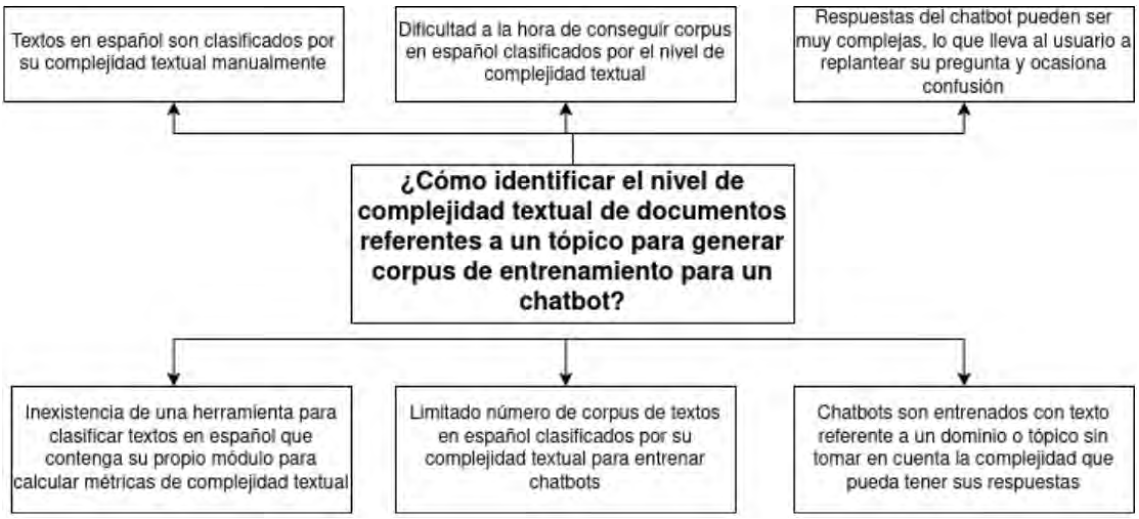

Figura 1. Árbol de Problemas.

1.1.2 Descripción

La generación de diálogo implica diseñar un programa para generar una conversación natural, esto requiere desarrollar algoritmos que puedan conversar con un ser humano u otro programa de manera coherente y fluida (Jurafsky & Martin, 2019). Poder participar en un diálogo o simplemente chatear con otros es una habilidad humana natural. Ciertas personas son buenas charlando y otras se sienten atraídas por ellas; en otras palabras, las primeras pueden dirigir la conversación hacia temas que son de interés de su interlocutor, seguir un tema de conversación durante un período prolongado y agregar detalles según sea necesario (Clair et al., 2018a).

Pero el desarrollar una conversación, también depende del contexto y conocimiento del interlocutor, lo cual nos lleva a definir la existencia de niveles de complejidad conversacional, área de investigación también conocida como análisis de discurso (Johnstone, 2018) o complejidad textual, la cual se refiere a determinar que tan fácil o difícil de leer es un texto (Halliday, 1985; Perfetti et al., 2005). En este aspecto, especialmente para el idioma español, no existe una herramienta que tenga un módulo propio que le permita clasificar textos en español por su complejidad textual

(Quispesaravia et al., 2016a). La clasificación de textos se refiere a la tarea de asignar una categoría a un documento escrito (Jurafsky & Martin, 2019; Yang & Joachims, 2008). En cambio, para otros idiomas, sí podemos encontrar algunas herramientas que evalúan la complejidad textual, basado en ciertos criterios medibles, que se refieren a las métricas de complejidad textual, los cuales son el conjunto de modelos y fórmulas que identifican que tan fácil o difícil de leer es un texto (Balakrishna, 2015; Choi, 2018a; Crossley et al., 2019; Davoodi et al., 2017; Fitzgerald et al., 2015); y además porque existen varios conjuntos de textos preprocesados, denominados corpus, de estos idiomas.

Por ejemplo una herramienta de análisis de complejidad textual es Coh-Metrix, el cual se usa para idioma inglés (Graesser et al., 2004). Esta herramienta ha sido utilizada en otros idiomas de forma limitada, como portugués (Hartmann et al., 2016), francés (Lecorvé et al., 2019), ruso (Guryanov et al., 2017), entre otros; pero para el caso del idiomas español, solo una herramienta fue encontrada durante la revisión: Coh-metrix-esp, la cual es una adaptación de la herramienta Coh-metrix original que sólo implementa 45 métricas de complejidad textual de las 108 disponibles en la versión original de Coh-metrix para inglés (Quispesaravia et al., 2016a). Este hecho genera que dicha herramienta para el idioma español no sea tan eficaz como la versión original para inglés. También, esta herramienta llamada Coh-metrix-esp fue implementada, en su momento, usando lenguaje Java, además de depender de una herramienta de terceros llamada Freeling, la cual provee servicios de análisis textual que calculan las métricas de complejidad textual (Quispesaravia et al., 2016a).

Otra deficiencia es la inexistencia de un corpus clasificados por complejidad textual para el idioma español que posea más de cien textos. Aquellos corpus se refieren a conjuntos de entrenamiento comprendidos por uno o más textos, destinados a la investigación científica (Jurafsky & Martin, 2019). Dicha insuficiencia se puede observar al revisar la literatura del tema, siendo que la mayor disponibilidad de corpus de entrenamientos están para el idioma inglés y chino; sin embargo, para el caso del español, existen corpus muy limitados y cuyos textos abarcan cien cuentos para niños y adultos (Quispesaravia et al., 2016a). Esta poca disponibilidad de corpus en español que posean una buena cantidad de documentos, obliga que al momento de desarrollar nuevas aplicaciones, como el análisis de complejidad de textos en determinados

dominios, uno deba de trabajar manualmente más textos, preprocesarlos y clasificarlos y, con ello, crear un nuevo corpus para realizar los experimentos propios.

Ahora, para presentar el último problema causa, se necesita primero saber que es un *Chatbot*. Estos son programas que imitan la conversación humana mediante algoritmos de aprendizaje de máquina o árboles de decisiones (Jurafsky & Martin, 2019). Además, el aprendizaje de máquina se refiere a la rama de la inteligencia artificial en la cual se busca que una computadora aprenda sin ser explícitamente programada (Russell et al., 1995). También, estos *Chatbots* son entrenados con textos, en este caso en español, los cuales pueden ser una conversación o un documento informativo/narrativo cuyo contenido afecta las respuestas del *Chatbot* según cómo haya sido este entrenado (Beaver, 2018).

Entonces, a la falta de herramientas para clasificar textos en español por su complejidad textual y de corpus en aquel idioma, para interés de la presentación investigación, se identifica que los *Chatbots* son entrenados con textos referente a un dominio o tópico sin tomar en cuenta la complejidad que puedan tener sus respuestas (Beaver, 2018). Dicha situación genera que las respuestas del *Chatbot* sean muy complejas, lo que lleva al usuario a replantear sus preguntas y así se genera confusión; aquella situación puede ser observada en el estudio realizado por Beaver (2018) en donde el *Chatbot* que desarrollaron ocasionaba que los usuarios se confundieran cuando las respuestas que daba dicho *Chatbot* eran muy complejas y, debido a ello, los usuarios tenían que replantear sus preguntas (Beaver, 2018). Además, se conoce que la implementación de módulos que hacen que los chatbots sean más usables generan respuestas positivas en los usuarios, como se observa con el desarrollo de la ventana Convey para *Chatbots*, la cual les mostraba a los usuarios del programa el contexto conversacional entre el *Chatbot* y el usuario (Jain et al., 2018).

1.1.3 Problema seleccionado

Teniendo los problemas causa y efecto se obtiene el siguiente problema principal: ¿Cómo identificar el nivel de complejidad textual de documentos referentes a un tópico para generar corpus de entrenamiento para un C*hatbot*?

1.2 Objetivos

En esta sección se presentarán el objetivo general y los objetivos específicos obtenidos luego de analizar el árbol de problemas que se presentó en la sección anterior.

1.2.1 Objetivo general

Desarrollar un *Framework* para identificar el nivel de complejidad de documentos referentes a un tópico aplicando métricas de análisis textual para generar corpus de entrenamiento para un C*hatbot*.

1.2.2 Objetivos Específicos

Para conseguir nuestro objetivo general, se definen los siguientes objetivos específicos:

O 1. Implementar el módulo para calcular métricas de complejidad textual.

O 2. Constituir un corpus de más de cien textos en español clasificados por su complejidad textual.

O 3. Evaluar el impacto de los corpus generados en las respuestas de un *Chatbot*.

1.2.3 Resultados Esperados

O 1. Luego de implementar el módulo para calcular métricas de complejidad textual, se tendrán los siguientes resultados.

R 1. Un conjunto de métricas de complejidad textual seleccionadas para analizar textos en español.

R 2. Una herramienta de implementación propia que calcula las métricas de complejidad textual para textos en español.

R 3. Un módulo de clasificación de complejidad textual parametrizado al corpus de textos en español.

O 2. Luego de constituir un corpus de más de cien textos en español clasificados por su complejidad textual, se tendrán los siguientes resultados.

R 1. Un conjunto de textos en español seleccionados por su complejidad textual.

R 2. Validación del agrupamiento de los textos en español seleccionados por su complejidad textual.

R 3. Un corpus de textos en español procesados y clasificados por su complejidad textual.

O 3. Luego de evaluar el impacto de los corpus generados en las respuestas de un *Chatbot*, se tendrán los siguientes resultados.

R 1. Selección de un *Chatbot* implementado para prueba.

R 2. Entrenamiento del *Chatbot* usando corpus de textos en español clasificados por su complejidad textual.

R 3. Pruebas de validación de un *Chatbot* con al menos 3 tópicos del corpus clasificado automáticamente.

1.2.4 Mapeo de objetivos, resultado y verificación

Tabla 1: Resultados del Objetivo 1

Objetivo: Implementar el módulo para calcular métricas de complejidad textual		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
Un conjunto de métricas de complejidad textual seleccionadas para analizar textos en español	Reporte de estudio de selección de métricas de complejidad textual.	-Lista de 45 métricas seleccionadas y validado por un experto al 100%.
Una herramienta de implementación propia que calcula las métricas de complejidad textual para textos en español	- Código de la librería de Python implementada en GitHub. - Documentos y manuales de la librería.	-Un conjunto de 45 algoritmos de cálculo de métricas de complejidad textual implementadas al 100%. -Documentación de la librería implementada validado por un experto al 100%.
Un módulo de clasificación de complejidad textual parametrizado al corpus de textos en español	Código del clasificador implementado en Python en GitHub.	- Modelo de aprendizaje de máquina validado por un experto al 100%. - Resultados de las métricas de Accuracy, Precision, Recall y F1 Score mayor al 75%

Tabla 2. Resultados del Objetivo 2

Objetivo: Constituir un corpus de más de cien textos en español clasificados por su complejidad textual.		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
Un conjunto de textos en español seleccionados por su complejidad textual	Archivos de la base de datos en formato SQLite.	-Base de datos de documentos en español con más de 100 registros validado por un experto al 100%.
Validación del agrupamiento de los textos en español seleccionados por su complejidad textual	Reporte de clustering.	- 1 Gráficos de validación y su interpretación de las clases halladas por el modelo de clustering con respecto a las clases halladas manualmente. - 1 Gráfico de Inercia y su interpretación por número de clases para la evaluación del modelo de clustering. - 1 Gráfico de agrupamiento y su interpretación.
Un corpus de textos en español procesados y clasificados por su complejidad textual	Archivos de la base de datos en formato SQLite.	-Base de datos con archivos clasificados por su complejidad textual, cuyas métricas de complejidad textual han sido calculadas, con más de 100 registros validado por un experto al 100%.

Tabla 3. Resultados del Objetivo 3

Objetivo: Evaluar el impacto de los corpus generados en las respuestas de un chatbot		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
Selección de un *Chatbot* implementado para pruebas	Reporte de selección de un *Chatbot* con código fuente de GitHub o GitLab	- Un *Chatbot* seleccionado para las pruebas.
Entrenamiento del *Chatbot* usando corpus de textos en español clasificados por su complejidad textual	Reporte de entrenamiento del *Chatbot* seleccionado.	- 1 Gráfico de curva de error de entrenamiento del *Chatbot*.
Pruebas de validación de un *Chatbot* con al menos 3 tópicos del corpus clasificado automáticamente	Reportes con gráficos estadísticos de las pruebas de validación.	- 2 experimentos estadísticos de validación para evaluar la capacidad de compresión y respuesta del *Chatbot*.

1.3 Métodos y Procedimientos

1.3.1 Resumen

En esta sección se indicarán los métodos, herramientas o procedimientos a usar por cada resultado esperado.

Tabla 4. Herramientas por objetivo

Resultado	**Herramientas/métodos/procedimientos**
Un conjunto de métricas de complejidad textual seleccionadas para analizar textos en español	Jupyter Notebook Google Colab

	Coh-Metrix Git
Una herramienta de implementación propia que calcula las métricas de complejidad textual para textos en español	Python Coh-Metrix Spacy Git
Un módulo de clasificación de complejidad textual parametrizado al corpus de textos en español	Python Scikit-learn Jupyter Notebook Google Colab Git
Un conjunto de textos en español seleccionados por su complejidad textual	Python Jupyter Notebook SQLite
Validación del agrupamiento de los textos en español seleccionados por su complejidad textual	Python Scikit-learn Jupyter Notebook Google Colab Git
Un corpus de textos en español procesados y clasificados por su complejidad textual	Python Jupyter Notebook SQLite
Selección de un chatbot implementado para pruebas	Python Jupyter Notebook Google Colab Git
Entrenamiento del chatbot usando corpus de textos en español clasificados por su complejidad textual	Python Jupyter Notebook Google Colab Git
Pruebas de validación de un chatbot con al menos 3 tópicos del corpus clasificado automáticamente	Python Jupyter Notebook Google Colab Git

1.3.2 Definiciones

A continuación se definirán mejor las herramientas a usar para obtener los resultados esperados en esta investigación.

- Python: Es un lenguaje de programación multiparadigma, ya que soporta la programación orientada a objetos, programación imperativa y programación funcional el cual fue creado en 1991 por Guido van Rossum (Python, s. f.). Este lenguaje de programación se utilizará para desarrollar todo el código de la herramienta de clasificación de textos.

- Jupyter Notebook: Es una aplicación web de código abierto que permite crear y compartir documentos que contienen código en tiempo real, ecuaciones,

visualizaciones y texto narrativo (Project Jupyter, s. f.). Esta aplicación se usará para crear los reportes indicados en la sección de resultados.

- Google Colab: Basado en Jupyter Notebook, es una aplicación web que provee las mismas funcionalidades que Jupyter Notebook, excepto que el poder de cómputo usado es provisto en una máquina virtual de Google, la cual puede poseer aceleración por GPU o TPU de ser indicado por el usuario. (Google Inc., s. f.).

- Coh-Metrix: Es una herramienta computacional que produce índices de las representaciones linguísticas y de discurso de un texto. Implementa alrededor de 108 índices (Graesser et al., 2004). De esta herramienta se obtendrán las métricas de complejidad textual a implementar en la solución.

- Scikit Learn: Es una biblioteca de Python para aprendizaje automático de software libre que incluye varios algoritmos de clasificación, regresión, análisis de grupo y está diseñada para trabajar fácilmente con las otras librerías de Python como NumPy y SciPy. (scikit-learn, s. f.). De esta librería se utilizarán los modelos de aprendizaje no supervisado para realizar el reporte de clustering y los modelos de aprendizaje supervisado para desarrollar el clasificador de textos.

- Git: Es un sistema distribuido de control de versionado libre y de código abierto diseñado para soportar proyectos de cualquier tamaño de manera eficiente y rápida que fue creado por Linus Torvalds en el año 2005 (Git, s. f.). Se utilizará Git junto a GitHub para versionar el código desarrollado.

- SQLite: Es una librería hecha en Lenguaje C que implementa un motor de base de datos rápido, auto contenido, altamente confiable y con características completas (SQLite, s. f.). Este motor de base de datos se utilizará para guardar los textos descargados y el corpus generado.

- Spacy: Es una librería de Python de código abierto para realizar procesamiento de lenguaje natural avanzado, a más alto nivel que NLTK, que fue desarrollado por Matthew Honnibal e Ines Montani (Spacy, s. f.). Esta librería de Python será

utilizada para implementar el cálculo de métricas de complejidad textual, usando su modelo en español para procesar los textos.

Capítulo 2. Marco Conceptual

2.1 Introducción

A continuación se definirán los conceptos necesarios para entender mejor la problemática definida en el capítulo 1.

2.2 Desarrollo del marco

2.2.1 Aprendizaje de máquina

Machine Learning, aprendizaje de máquina o aprendizaje automático es una rama de la inteligencia artificial en la cual se busca que una computadora aprenda sin ser explícitamente programada, mejorando su desempeño a través de la experiencia y el entrenamiento sin ser explícitamente programada. (Russell et al., 1995).

Por ejemplo, en el contexto del problema de la presente investigación, el aprendizaje de máquina se utilizará para entrenar modelos que clasifiquen textos automáticamente según su complejidad textual.

Dicha complejidad textual se refiere a qué tan fácil o difícil de leer es un texto (Esta definición se verá más a detalle en el punto 2.2.3). Además, para que dicho texto pueda ser clasificado, se medirá el contenido usando las métricas de complejidad textual (Descrito a mayor detalle en el punto 2.2.5).

¿Por qué es necesario calcular dichas métricas para poder entrenar el modelo de aprendizaje de máquina? Esto es necesario debido a que dichas métricas son valores numéricos y los modelos de aprendizaje de máquina necesitan entradas numéricas para poder ejecutar sus algoritmos. En la Figura 2 podemos apreciar dos modelos comunes

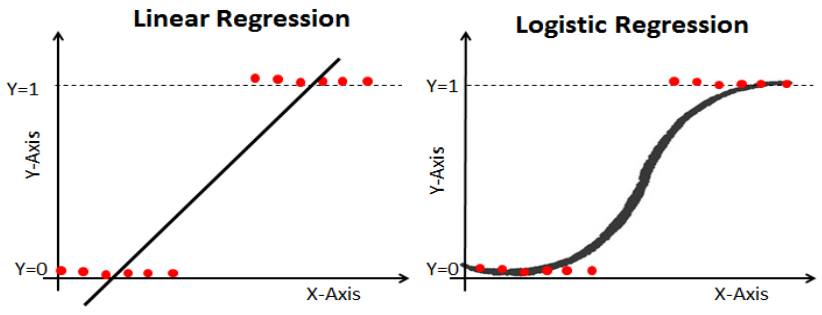

Figura 2. Regresión Lineal y logística. Extraído de https://meraju.com/blog/

de aprendizaje de máquina: Regresión Linear y Regresión Logística. Estos modelos, y los demás, utilizan entradas numéricas para poder encontrar una solución.

Entonces, para poder identificar el nivel de complejidad de un texto y mediante un corpus adecuadamente generado, será factible el entrenamiento de un chatbot mediante uno o varios modelos de *aprendizaje de máquina* y así buscar que el proceso de identificación sea automático.

2.2.2 Chatbot

Los *Chatbots* o robots conversacionales son sistemas que, mediante el uso del lenguaje natural, pueden tener conversaciones extensas con las personas. Dichos sistemas pueden imitar conversaciones no estructuradas, o chats, y pueden ser usados de manera recreativa o para resolver una tarea. (Jurafsky & Martin, 2019).

Estos robots conversacionales pueden ser de dos tipos: Los *Chatbots* basados en reglas, los cuales usan árboles de decisiones para poder encontrar una respuesta adecuada al diálogo recibido; y los basados en corpus, los cuales aprenden las respuestas que deben dar al ser entrenados con datos de entrenamiento, como conversaciones (Jurafsky & Martin, 2019).

Los *Chatbots* son implementados de tres maneras distintas, se tienen aquellos basados en reglas escritas manualmente o árboles de decisiones; están los sistemas de recuperación de información y por último, los modelos de "*Encoder-Decoder*". Los basados en reglas, son aquellos básicos que desarrollan preguntas y respuestas predefinidas; y a los modelos de recuperación de información se orientan a la búsqueda en una base de datos de conocimiento previamente definida, sobre la cual se aplican técnicas clásicas como algoritmos de búsqueda en cadena de texto; finalmente, los modelos Encoder-Decoder usan arquitecturas de redes neuronales para realizar la interpretación de las intenciones y otorgar respuestas.

Uno de los componentes de un *Chatbot* es el que se refiere a la extracción de tópicos, y esto a su vez no lleva al concepto de complejidad de tópico, lo cual nos permite modelar el contexto del diálogo, reconocer la intención del hablante y definir una política o estrategia del diálogo.

Como ejemplo para entender mejor la problemática, se tiene el caso de un *Chatbot* usado en un ambiente educativo. Dicho *Chatbot* podría existir como un ayudante virtual que guíe a los alumnos a través de los distintos cursos. Aquellos alumnos no necesariamente serán de un solo grado sino que pueden pertenecer a todos los grados educativos. ¿Qué pasaría si, por ejemplo, un alumno de primaria hace una pregunta al *Chatbot* sobre un tema en específico y uno de secundaria también? El robot conversacional necesitaría contestar de tal forma que sus respuestas tengan la complejidad correcta y no generen confusión en los alumnos de distintos grados. Esto es importante para que esos estudiantes puedan seguir con sus actividades sin tener que esperar la respuesta del profesor en persona, por ejemplo.

¿Pero cómo haría un *Chatbot* para saber diferencia entre cuán complejo o simple debería responder? Para ello se entrenarían a estos robots conversacionales con textos de distintos niveles de complejidad, siendo que en este punto se hace necesario poder modelar y desarrollar los algoritmos que consigan determinar esta complejidad textual. De esta manera se podrá otorgar apoyo, por ejemplo, para el caso de un *Chatbot* para educación básica regular (primaria y secundaria), pueda establecer niveles de complejidad en sus respuestas de acuerdo al grado del alumno con el cual interactúa el *Chatbot*.

Dados los dos siguientes ejemplos de conversaciones:

Alumno primaria: ¿Qué es un cuadrado? **Chatbot:** Es una figura plana de 4 lados iguales.	**Alumno secundaria:** ¿Qué es un cuadrado? **Chatbot:** Es una figura geométrica bidimensional que posee 4 lados iguales y cuatro ángulos interiores rectos, con diagonales perpendiculares.

Al analizarlos, se puede identificar que la conversación de la derecha es más compleja que la conversación de la izquierda, debido a que contiene un mayor número de palabras y además contiene términos que implican otros conceptos adicionales. La conversación de la derecha podría confundir a un alumno que no posee el conocimiento necesario para poder entenderla, como son los de primaria.

2.2.3 Clasificación de textos

La clasificación de textos es una tarea en la cual se asignan categorías predefinidas a documentos que no las tienen. Esta tarea puede ser realizada manualmente usando criterio experto o automáticamente mediante algoritmos de aprendizaje de máquina (Jurafsky & Martin, 2019; Yang & Joachims, 2008).

Entonces, ¿Para que nos sirve la clasificación de textos? Como el problema a resolver trata sobre como identificar la complejidad textual de distintos textos para poder entrenar un *Chatbot*; para esto, se necesitará asignar una complejidad, la cual vendría a ser una categoría, a un documento. Al asignar una complejidad, la cual es una categoría, se está realizando la tarea de clasificación de textos. Lo que se busca, es realizar dicha tarea de manera automática y no hacerla de manera manual.

2.2.4 Complejidad textual

La complejidad textual se refiere a determinar que tan fácil o difícil de leer es un texto. Evidencia empírica ha demostrado que existe una correlación entre una densidad léxica baja y la facilidad con que un texto es comprendido (Perfetti et al., 2005). Por otra parte, una elevada densidad léxica es un rasgo que diferencia la escritura de la oralidad, pues esta última tiende a un léxico más repetitivo y a una menor complejidad sintáctica (Halliday, 1985). Para el análisis de la complejidad de texto, se puede abordar desde tres perspectivas:

- Cuantitativo:

  Este modelo ser medido por las software y mida la frecuencia y longitud de las palabras, longitud de oraciones, y cohesión textual. Se incluyen aquí las fórmulas de legibilidad de Dale-Chal, el grado de Flesch-Kincaid y el framework de legibilidad Lexile (Hiebert, s. f.).

- Cualitativo:

  Este modelo puede ser medido por el mismo lector. Se mide mediante significado de los textos literarios, o el propósito de los textos informativos, estructura textual, lenguaje, convencionalismo, claridad, y si se requiere conocimiento extra (Hiebert, s. f.).

- Lector y tareas

  Este modelo se enfoca en el lector individualmente y la tarea o propósito de la lectura. Se puede determinar que tan apropiado es un texto, analizando la motivación, conocimiento y experiencia del estudiante; además del criterio experto del profesor (Hiebert, s. f.).

Pero, ¿Por qué es importante conocer sobre la complejidad textual? Debido a que el problema principal trata sobre identificar la complejidad de textos para poder entrenar robots conversacionales, debemos entender qué es la complejidad textual en sí. Entonces, si se quiere poder identificar que tan difícil o fácil de entender un texto, necesitamos entender cuales son los factores que se deben analizar para hallar dicha respuesta. Aquellos factores como el aspecto cuantitativo, cualitativo, el lector y las tareas, ayudarán a entender mejor cómo identificar la complejidad textual de un documento.

2.2.5 Métricas de complejidad textual

Las métricas de complejidad textual son un conjunto de modelos y fórmulas de legibilidad con las cuales se puede determinar que tan difícil o fácil de leer es un texto. Para ello, dichas métricas analizan las distintas características de un texto: descriptivas, morfo-sintácticas y significado de las palabras (Balakrishna, 2015; Choi, 2018a; Crossley et al., 2019; Davoodi et al., 2017; Fitzgerald et al., 2015, 2016; López-Anguita et al., 2018; Ojha et al., 2018; Solovyev et al., 2018; Stevanoski & Gievska, 2019; Trotzek et al., 2017).

Sin embargo, ¿Para qué sirven estas métricas de complejidad textual? Como se pudo apreciar en el punto 2.2.4, los modelos usados para medir la complejidad de un texto incluyen un modelo cuantitativo, el cual puede ser automatizado. Dicho modelo puede ser analizado describiendo el contenido del texto de manera numérica, analizando la morfo-sintaxis del contenido y el significado de las palabras. Una vez encontrados dichos valores numéricos, se pueden enviar aquellos datos a un modelo de aprendizaje automático. Aquello se realiza con la intención de automatizar el proceso de identificación de la complejidad textual.

2.2.6 Corpus

Un corpus es un conjunto de entrenamiento constituido por un solo texto o por varios de ellos, el cual está destinado a la investigación científica. Estos corpus tienen que ser legibles por computadoras. Además, existen dos tipos de corpus: Escritos y hablados (Jurafsky & Martin, 2019).

¿Qué tan importantes son los corpus para nuestro problema de identificación de complejidad textual? El éxito de un modelo de aprendizaje automático, radica en la

cantidad y calidad de información que es utilizada para su entrenamiento. Para el problema en estudio, los corpus son importantes ya que estos contienen los textos que se usarán para poder realizar el experimento de identificación de complejidad textual. Con estos textos, se puede aplicar el conocimiento conceptual de complejidad textual, métricas de complejidad textual y aprendizaje de máquina para poder entrenar un modelo que clasifique automáticamente dichos documentos según el nivel de complejidad textual.

Capítulo 3. Estado del Arte

3.1 Introducción

> Para realizar la revisión del Estado del Arte, se decidió usar la metodología de la revisión sistemática, la cual fue propuesta por Barbara Kitchenham. El uso de esta metodología de investigación permitió que al investigador resumir la evidencia existente con respecto al uso de la tecnología usada para resolver problemas similares al que se investigó en este documento (Keele, 2007).

La búsqueda de documentos fue llevada a cabo usando los motores de búsqueda "Google Scholar", "ACM Digital Library" y "IEEE Explore."

3.2 Objetivos de revisión

> Para realizar esta investigación, se decidió realizar una revisión teórica y del estado del arte de los documentos encontrados.

Esta revisión fue conducida para poder identificar qué tecnologías han sido usadas recientemente para analizar la Complejidad Textual y así poder responder a la pregunta principal de la investigación: ¿Cómo identificar el nivel de complejidad de un texto referente a un tópico para poder entrenar un Chatbot conversacional? Cabe recordar que dichos textos cuya complejidad haya sido identificada serán usados para entrenar chatbots con distinto nivel de complejidad en sus respuestas.

Para este análisis se diseñó la siguiente tabla usando el método PICOC, las cuales están indicadas en la Tabla 5.

Tabla 5. Criterios PICOC

Criterio PICOC	Consideraciones
Population	Identificación de nivel de complejidad de un texto
Intervention	Uso de métricas de complejidad textual
Comparison	No aplica
Outcome	Resultados de las investigaciones, posibles mejoras, determinación de métricas orientadas al análisis de complejidad de textos.
Context	Académico o industrial

3.3 Preguntas de revisión

Para la resolución de la pregunta principal se decidió hacer las siguientes preguntas complementarias:

- P1. ¿Cómo afecta al entrenamiento de un Chatbot el uso de texto complejo?
- P2. ¿Cuáles son las métricas/medidas de análisis de complejidad textual y como se usan para analizar la complejidad de textos?
- P3. ¿Por qué es necesario realizar un análisis de complejidad textual?

3.3.1 Estrategia de búsqueda

Para analizar el estado del arte, se buscaron estudios primarios en tres bases de datos académicas mediante 1 cadena de búsqueda por pregunta: Google Scholar, ACM Digital Library e IEEE Explore.

Sin embargo, no todos los documentos encontrados fueron usados debido a que no todos cumplían con los criterios de inclusión y exclusión, algunos eran documentos repetidos o no guardaban relación con el tema a investigar. Para más detalle, revisar el Anexo B. A continuación, se presenta la estrategia de búsqueda.

3.3.2 Motores de búsqueda a usar

Para encontrar información que ayuden en la presente investigación, se ha decido utilizar algunas bases de datos referenciadas por la Pontificia Universidad Católica del Perú, así como las que fueron sugeridas por el asesor. Estas bases de datos a usar son las siguientes:

- Google Scholar.
- ACM Digital Library.
- IEEE Explore

3.3.3 Cadenas de búsqueda a usar

Para generar las cadenas de búsqueda a usar por cada pregunta, se utilizaron los conectores lógicos como AND y OR, además de agregar sinónimos a las palabras clave

en caso sea posible. Dichas palabras clave fueron usadas en inglés y no en español. Para esto, se realizó una tabla PICOC por cada pregunta,.

A continuación en la Tabla 6, se muestra la tabla PICOC para la pregunta 1, la cual es "¿Cómo afecta al entrenamiento de un Chatbot el uso de texto complejo?".

Tabla 6. Criterios PICOC para la pregunta 1

Criterio PICOC	Consideraciones
Population	Chatbot, Text
Intervention	
Comparison	
Outcome	
Context	Text Complexity, Text Difficulty

Para la pregunta 2, la cual es "¿Cuáles son las métricas/medidas de análisis de complejidad textual y como se usan para analizar la complejidad de textos?", se muestran los criterios PICOC en la Tabla 7.

Tabla 7. Criterios PICOC para la pregunta 2

Criterio PICOC	Consideraciones
Population	Text complexity measures, Text complexity measure, Text complexity metrics
Intervention	Texto Classification, Classification
Comparison	
Outcome	
Context	Machine Learning

Los criterios PICOC de la pregunta 3, la cual es "¿Por qué es necesario realizar un análisis de complejidad textual?", están en la Tabla 8.

Tabla 8. Criterios PICOC para la pregunta 3

Criterio PICOC	Consideraciones
Population	Text complexity
Intervention	Text Classification, Classification
Comparison	
Outcome	
Context	Coh-Metrix

Gracias al análisis hecho mediante el uso de tablas con criterios PICOC, se pudo crear las cadenas de búsqueda a usar por pregunta, las cuales se muestran en la Tabla 9.

Tabla 9. Cadenas de búsqueda.

Pregunta	Cadena de búsqueda
P1	(“Chatbot”) AND (“Training” OR “Train”) AND (“Text Complexity” OR “Text Difficulty”)
P2	("Text Complexity Measure" OR "Text Complexity Metrics" OR "Text Complexity Measures") AND ("Text Classification" OR "Classification") AND (“Machine Learning”)
P3	("Text Complexity" AND "Machine Learning" AND ("Text Classification" OR "Classification") AND "Coh-Metrix" AND ("Reason" OR "Objective")) OR "AzterTest"

3.3.4 Documentos encontrados

En la Tabla 10 se muestran la cantidad de documentos por pregunta, usando las cadenas de búsqueda creadas, detallando la cantidad de documentos totales antes y después de aplicar los criterios de inclusión y exclusión. Las cadenas de búsqueda fueron ejecutadas por última vez el 9 de Mayo del 2020.

Tabla 10. Documentos encontrados por motor de búsqueda

Motor de búsqueda	P1	P2	P3
Google Scholar	13	45	103
ACM Digital Library	2	2	3
IEEE Explore	1	0	0
Total sin criterios	16	47	106
Total con criterios	3	11	24

3.3.5 Criterios de Inclusión/Exclusión

Para esta investigación, no será posible utilizar todos los documentos encontrados, debido a la gran cantidad de los resultados obtenidos. Por ello, y para evitar caer en la imparcialidad, se han formulado los siguientes criterios de inclusión y exclusión.

- Criterios de inclusión.
    - El documento encontrado debe haber usado la herramienta “Coh-metrix”. Este criterio aplica para los documentos encontrados para la pregunta 3.
        - Este criterio se ha considerado debido a que esta herramienta es la más completa para analizar la complejidad textual ya que implementa cerca de 200 medidas de cohesión, lenguaje y legibilidad (Graesser et al., 2004).

- El documento debe haber sido citado al menos 1 vez. Este criterio aplica para los documentos encontrados para todas las preguntas.
    - Este criterio se considera ya que así se podrá saber que tan relevante es en el campo de investigación. Habrá excepciones si es que en el documento se analiza la complejidad textual para textos en español.
- El tema del documento debe guardar relación con el tema a desarrollar en esta investigación. Este criterio aplica para todos los documentos que hayan pasado los filtros de fecha, idioma y cantidad de citaciones.
    - Este criterio se considera debido a que algunos de los documentos encontrados son una revisión del estado del arte. Otros documentos están relacionados a otras áreas de estudio como por ejemplo las ciencias sociales, dejando de lado el tema de aprendizaje de máquina.

- Criterios de exclusión.
    - El documento debe ser del año 2014 o más reciente. Este criterio aplica para los documentos encontrados para todas las preguntas.
        - Como se hará una revisión del estado del arte, la antigüedad del documento no debe ser mayor a los 6 años. Este filtro no tiene efecto sobre los documentos para la revisión teórica.
    - El documento debe estar en español o inglés. Este criterio aplica para los documentos encontrados para todas las preguntas.
        - Se consideró solo utilizar documentos en español o inglés porque el primero es la lengua nativa del investigador y el segundo porque la mayoría de documentos están en inglés.
    - El documento no debe ser una patente. Este criterio aplica para todos los documentos que hayan pasado los filtros de fecha, idioma y cantidad de citaciones.

- Se consideró este criterio debido a que, al realizar la revisión de patentes de Google encontrada, estas no contenían mucha información y no aportaban a la investigación.

3.4 Formulario de extracción de datos

Para realizar la extracción de datos, se plantearon las siguientes preguntas, las cuales están en la Tabla 11 y con las cuales se creó el formulario de extracción.

Tabla 11. Formulario de Extracción

Campo	Descripción	Pregunta General
Título	El título del artículo	Criterio general
Autor	El o los autores del artículo	Criterio general
Año de publicación	Año de publicación del artículo	Criterio general
Idioma	Idioma en el que está el artículo	Criterio general
Modelo de aprendizaje de máquina usado	Modelo tradicional o de red neuronal usado en el artículo	P1
Estructura del modelo de aprendizaje de máquina usado	Arquitectura del modelo de red neuronal, si es que fue usada una red	P1
Medidas de rendimiento usado en los modelos	Puede ser "accuracy", "precision", "recall", "F1 score"	P1
Resultados numéricos de las métricas de medidas de rendimiento	Se necesita para determinar qué tan exitoso fue el experimento	P1
Nombre del conjunto de datos usado	El nombre del dataset de entrenamiento	P1, P2 Y P3
Idioma del conjunto de datos usado	El idioma del dataset con el cual fue realizado el experimento	P1, P2 Y P3
Nombre de los conjuntos de métricas de complejidad textual	Las métricas de complejidad textual son agrupadas por grupos distintos.	P2
Nombre individual de cada métrica de complejidad textual	Las métricas de complejidad textual son agrupadas por grupos distintos	P2
Objetivos	Objetivo del experimento realizado	P3
Resultados	Resultados del experimento realizado	P3

3.5 Resultados de la revisión

Para responder a las preguntas de revisión, se creó una tabla por pregunta donde se listaron las publicaciones obtenidas mediante las cadenas de búsqueda. También, los formularios se recopilaron en tres tablas distintas, nuevamente una por pregunta. Estas

están en la hoja de cálculo como indica el Anexo B, anexo en el cual se pueden encontrar el nombre de las publicaciones totales encontradas y las que quedaron luego de aplicar los criterios de inclusión y exclusión.

3.5.1 Respuesta a la pregunta "¿Cómo afecta al entrenamiento de un Chatbot el uso de texto complejo?"

Para responder esta pregunta se pudo deducir que el nivel de complejidad de una respuesta hecha por un Chatbot puede ocasionar que el usuario reformule su pregunta o no (Beaver, 2018). Por otro lado, un Chatbot suficientemente entrenado puede identificar el nivel de dificultad de las respuestas de un usuario en una conversación en aplicativos de mensajería (Mohammadi & Khasteh, 2018a). Además, un Chatbot que ha sido entrenado con texto de longitud extensa puede predecir de manera más exacta qué diálogos de una conversación son buenos y cuáles malos (Cuáyahuitl et al., 2019a). Entonces, se puede responder que el entrenamiento de un Chatbot con textos de distintos niveles de complejidad puede influenciar en el flujo de la conversación.

3.5.2 Respuesta a la pregunta "¿Cuáles son las métricas/medidas de análisis de complejidad textual y como se usan para analizar la complejidad de textos?"

Luego de analizar los datos encontrados en el formulario para esta pregunta, se pudo recopilar que existen varias formas de medir la complejidad de un texto. Por ejemplo, se tienen medidas que analizan las características descriptivas de un texto: Número de caracteres, oraciones, párrafos y sílabas (Stevanoski & Gievska, 2019).También se analizan las características morfo-sintácticas de los textos, por ejemplo: Cantidad de sustantivos, verbos, adjetivos, pronombres, pronombres personales, negaciones y conectores por oración (Solovyev et al., 2018). Además, se puede analizar el significado de las palabras mediante la edad de adquisición, la abstracción y la rareza de la palabra (Fitzgerald et al., 2015, 2016).Otras métricas comúnmente usadas son las fórmulas clásicas de legibilidad, las cuales mezclan los valores calculados en su mayoría por las medidas de características descriptivas. Estas son las siguientes: Flesch-Kincaid, Flesch, New Dale-Chall, Gunning Fog Index, Fernández-Huerta (Crossley et al., 2019; López-Anguita et al., 2018; Ojha et al., 2018; Trotzek et al., 2017). Existen otras métricas que se usan (Balakrishna, 2015; Choi, 2018a; Davoodi et al., 2017), pero las anteriores son las que se usaron con más frecuencia en los documentos analizados. Entonces, existen

distintas métricas que se usan para analizar distintas características textuales, como se explicó líneas más arriba, para así determinar qué tan complejo es un texto.

3.5.3 Respuesta a la pregunta "¿Por qué es necesario realizar un análisis de complejidad textual?"

Con los documentos recopilados, se pudo concluir que el análisis de la complejidad textual es necesario para resolver distintos problemas del mundo real, como por ejemplo: Detectar fraude financiero (S. Minhas & Hussain, 2016; S. Z. Minhas, 2016); inferir el lenguaje nativo de un autor (Bich, 2017); clasificar modelos estudiantiles (Aguirre et al., 2016); simplificar texto complejo (Balakrishna, 2015); clasificar textos según su complejidad (Bengoetxea et al., 2020; Branco et al., 2014; Fitzgerald et al., 2015; Hartmann et al., 2016; Kleijn, 2018; Quispesaravia et al., 2016a; Sung et al., 2015); evaluar la accesibilidad de los textos (Yaneva, 2016); calificar ensayos automáticamente (Latifi, 2016; Li & Liu, 2016; Patout & Cordy, 2019; Vajjala, 2018); analizar la dificultad de textos auditivos (Aryadoust & Goh, 2014); evaluar el grado de participación de un estudiante en un ambiente educativo virtual (Dascalu et al., 2018); analizar la complejidad de letras de canciones (Choi, 2018a); generar automáticamente corpus textuales clasificados según su complejidad textual (Vajjala & Lučić, 2018; Wagner Filho et al., 2016) y evaluar la legibilidad de los textos (YAO–TING et al., 2015)

3.6 Conclusiones

De lo que se obtuvo luego de realizar la revisión sistemática y responder a las tres preguntas planteadas, se puede concluir que:

Primero, la complejidad textual tiene muchas aplicaciones en el mundo real (pregunta 3), las cuales son las siguientes: Detectar fraude financiero; inferir el lenguaje nativo de un autor; clasificar modelos estudiantiles; clasificar textos según su complejidad; evaluar la accesibilidad y legibilidad de los textos; calificar un ensayo automáticamente; analizar la dificultad de textos auditivos; avaluar el grado de participación de un estudiante en un ambiente educativo virtual; analizar la complejidad de letras de canciones y generar automáticamente corpus textuales clasificados según su complejidad textual.

Segundo, para analizar la complejidad de los textos, se disponen de varias métricas y fórmulas que se pueden usar (pregunta 2), como por ejemplo: Número de caracteres, oraciones, párrafos, sílabas, cantidad de sustantivos, verbos, adjetivos, pronombres, pronombres personales, negaciones, conectores por oración, edad de aquisición, la abstracción, la rareza de la palabra, fórmula de Flesch-Kincaid, fórmula de Flesch, fórmula de New Dale-Chall, Gunning Fog Index, fórmula de Fernández-Huerta.

Tercero, el entrenar un Chatbot con textos de distinta complejidad puede, en efecto, afectar el flujo de la conversación (pregunta 1), por ejemplo: El Chatbot puede ocasionar que el usuario replantee su pregunta, se puede identificar el nivel de complejidad de la respuesta del usuario y predecir que diálogos son buenos y cuáles malos.

Capítulo 4. Implementación de las métricas de complejidad textual

4.1 Introducción

El presente capítulo responde al Objetivo Específico 1 (Implementar el módulo para calcular métricas de complejidad textual). En el cual se busca desarrollar un módulo que comprenda el cálculo de estas métricas.

Este módulo fue creado con la finalidad de adaptar la herramienta coh-metrix (Graesser et al., 2004; McNamara et al., 2014) al idioma español, tal y como fue realizado por Quispesaravia et al. (2016), quienes implementaron las métricas en lenguaje Java, limitando su uso con las actuales herramientas. Por ello, en el presente trabajo, se reinmplementan dichas métricas con herramientas más actualizadas para el desarrollo, como Python para el desarrollo, la librería Spacy para cumplir con las necesidades de Procesamiento de Lenguaje Natural, así como la librería Scikit-Learn para entrenar los modelos de clasificación que identifiquen el nivel de complejidad de textos en español. Este módulo implementado será luego utilizado en los siguientes objetivos para analizar la complejidad textual de los textos encontrados así como permitir al Chatbot identificar el nivel de complejidad que deba usar en sus respuestas. Para ello se presentan tres componentes: Definir el conjunto de métricas de complejidad textual, la implementación de la herramienta que calcula dichas métricas y el módulo de clasificación que utiliza dichas métricas.

4.2 Resultados alcanzados

4.2.1 Un conjunto de métricas de complejidad textual seleccionadas para analizar textos en español

Para implementar la herramienta que calcula las métricas de complejidad textual, primero es necesario saber que métricas se van a implementar. Para ello, las métricas a desarrollar se basan en la herramienta Coh-metrix (Graesser et al., 2004; McNamara et al., 2014) y su adaptación para el idioma en español, Coh-metrix-esp (Quispesaravia et al., 2016a). Entonces, de todos los índices existentes en la herramienta coh-metrix, se implementarán los siguientes, los cuales se usaron en la adaptación para español: índices descriptivos, índices de información de palabras, índices de densidad de patrones sintácticos, índices de complejidad sintáctica, índices de conectivos, índices de diversidad léxica, índices de legibilidad e índices de cohesión referencial.

Cada uno de estos índices agrupa un conjunto de métricas, dando un total de 48, 3 por encima de lo originalmente planeado, las cuales serán implementadas. Los nombres de las métricas a implementar, así como la explicación de cada una de ellas, se encuentran en mayor detalle en el Anexo E, en donde se encuentra el reporte realizado para listar las métricas a usar, así como la validación correspondiente indicado en el indicador objetivamente verificable. A continuación, en la Figura 3, un ejemplo de la información que se puede encontrar según lo indicado en el Anexo E.

Métricas de complejidad

Para el presente trabajo de investigación, las métricas de complejidad textual a usar se basan en aquellas que fueron usadas por la herramienta Coh-Metrix (Graesser et al., 2004; McNamara et al., 2014).

Términos

- Palabras de contenido: Son los sustantivos, verbos, adjetivos y adverbios.
- Incidencia: En este caso, se define como incidencia al porcentaje de una característica por cada mil palabras.

Índices descriptivos (Descriptive)

- DESPC: Cantidad de párrafos totales en el texto. Un parrafo es separado por los caractéres '\n\n'.
- DESSC: Cantidad de oraciones totales en el texto.
- DESWC: Cantidad de palabras totales en el texto. Una palabra es aquel token identificado por Spacy que solo posee letras.
- DESPL: Promedio de oraciones por párrafo.
- DESPLd: Desviación estándar de la cantidad de oraciones por párrafo.
- DESSL: Promedio de palabras por oración.
- DESSLd: Desviación estándar de la cantidad de palabras por oración.
- DESWLsy: Promedio de sílabas por palabra.
- DESWLsyd: Desviación estándar de la cantidad de sílabas por palabra.
- DESWLlt: Promedio de letras por palabra.
- DESWLltd: Desviación estándar de la cantidad de letras por palabra.

Índices de información de palabras (Word information)

- WRDNOUN: Incidencia de sustantivos.
- WRDVERB: Incidencia de verbos.
- WRDADJ: Incidencia de adjetivos.
- WRDADV: Incidencia de adverbios.
- WRDPRO: Incidencia de pronombres.
- WRDPRP1s: Incidencia de pronombres personales en primera persona en singular.
- WRDPRP1p: Incidencia de pronombres personales en primera persona en plural.
- WRDPRP2s: Incidencia de pronombres personales en segunda persona en singular.
- WRDPRP2p: Incidencia de pronombres personales en segunda persona en plural.
- WRDPRP3s: Incidencia de pronombres personales en tercera persona en singular.
- WRDPRP3p: Incidencia dep ronombres personales en tercera persona en plugarl.

Índices de densidad de patrones sintácticos (Syntactic pattern density)

- DRNP: Incidencia de frases nominales.
- DRVP: Incidencia de frases verbales.
- DRNEG: Incidencia de expresiones negativas. Se identifican como expresiones negativas a las siguientes palabras: no, nunca, jamás, tampoco.

Figura 3. Reporte de métricas de complejidad textual (Recorte).

4.2.2 Una herramienta de implementación propia que calcula las métricas de complejidad textual para textos en español

Una vez identificadas las métricas de complejidad textual basadas en Coh-metrix, se procede con la implementación de las mismas; los cuales corresponden a los siguientes índices: los descriptivos, los de legibilidad, los de información de palabras, de densidad de patrones sintácticos, de complejidad sintáctica, de diversidad léxica y de cohesión referencial. En total, se implementarán 48 métricas de las 45 originalmente planeadas.

Para esta implementación, se utilizó el lenguaje de programación Python, la librería de procesamiento de lenguaje natural Spacy y git/Github para el control de versiones.

En la Figura 4 se puede apreciar un texto de prueba para el análisis con la herramienta desarrollada. En el se aprecia tres párrafos los cuales están separados por doble espacio ("\n\n") para poder identificarlos como párrafos. Además, las puntuaciones no han sido omitidas debido a que Spacy las necesita para poder identificar oraciones.

```
Ellos jugaron todo el día. Asimismo, ellas participaron en el juego.

Yo corro con el hermoso gato. A nosotros no nos gusta el gato.
Ella tiene mascotas. Aunque, ella tiene plantas y él no.

Tú jamás dijiste que no, porque ustedes debieron salir temprano al mismo tiempo.
```

Figura 4. Texto de prueba para la herramienta que calcula métricas de complejidad textual en español.

El primer conjunto de métricas, son los índices descriptivos, las cuales implementan 11 de las 48 métricas a implementar.

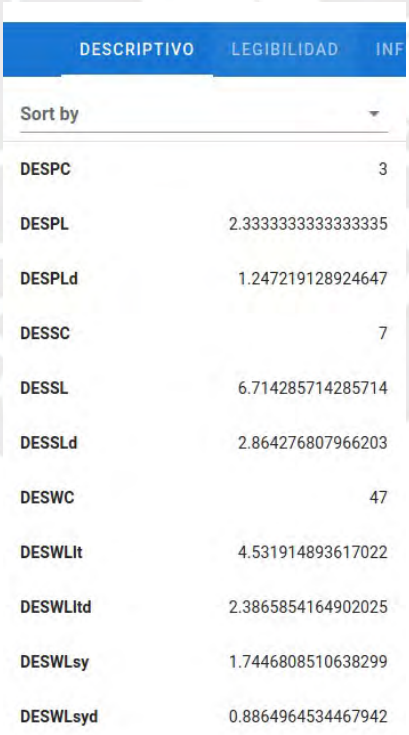

Figura 5. Índices descriptivos.

En la Figura 5 se aprecian los resultados del procesamiento del texto mostrado en la Figura 4. Las oraciones son identificadas mediante el Pipe conocido como 'Sentencizer' de Spacy y las palabras son considerados como aquellos Tokens de Spacy que solo

poseen caracteres alfabéticos. Del texto utilizado, se puede observar que, en efecto, hay 3 párrafos (DESPC), los cuales en promedio tienen 2.333 oraciones (DESPL) y su desviación estándar es 1.247 (DESPLd). En total hay 7 oraciones (DESSC), las cuales tienen en promedio 6.714 palabras (DESSL) y una desviación estándar de 2.864 (DESSLd). Se observa que hay 47 palabras (DESWC), donde en promedio, cada palabra tiene 4.5319 letras (DESWLt) con una desviación estándar de 2.386 (DESWLtd). Además, cada palabra tiene, en promedio 1.744 sílabas (DESWLsy) con una desviación estándar de 0.8864 (DESWLsyd). Estas dos últimas métricas fueron halladas con la ayuda de la librería Pyphen, la cual divide palabras, en distintos idiomas, en sílabas.

El segundo conjunto de métricas corresponden a los índices de legibilidad, las cuales implementan 1 de las 48 métricas a implementar.

Figura 6. Índices de legibilidad.

En la Figura 6 se aprecian los resultados del procesamiento del texto mostrado en la Figura 4. La métrica hallada corresponde al índice de Fernández-Huerta (Crossley et al., 2019; López-Anguita et al., 2018; Ojha et al., 2018; Quispesaravia et al., 2016a; Trotzek et al., 2017) la cuál tiene la siguiente fórmula:

$$206{,}84 - 0{,}6 * mspw - 1{,}02 * mwps$$

Donde mspw es el promedio de sílabas por palabra y mwps es el promedio de palabras por oración.

El tercer conjunto de métricas corresponden a los índices de información de palabras, el cual implementa 11 de las 48 métricas a implementar. Estos índices se obtienen luego de aplicar la siguiente fórmula:

$$(cantidad\ tipo\ palabra / total\ de\ palabras) * 1000$$

En la Figura 7 se aprecian los resultados del procesamiento del texto mostrado en la Figura 4.

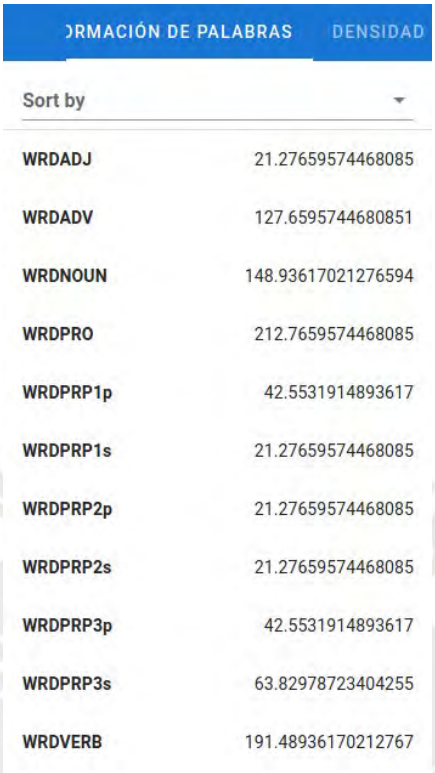

Figura 7. Índices de información de palabra.

En el texto analizado, hay 7 sustantivos (día, juego, gato, gato, mascotas, plantas, tiempo), el cual da un índice de 188.9361 (WRDNOUN). La cantidad de verbos es 9 (jugaron, participaron, corro, gusta, tiene, tiene, dijiste, debieron, salir), por lo que el índice de verbos da 191,489 (WRDVERB). La cantidad de adjetivos es 1 (hermoso), por lo que el índice de adjetivos es 21.276 (WRDADJ). La cantidad de adverbios es 6 (Asimismo, no, no, jamás, no, temprano), por lo que el índice de adverbios es 148.936 (WRDADV). La cantidad de pronombres es 10 (Ellos, ellas, Yo, nosotros, nos, Ella, ella, él, Tú, ustedes), por lo que el índice de pronombres personales es 212.7659 (WRDPRO). De ellos, solo hay 1 pronombre en primera persona en singular (Yo), lo que da un índice de 21.2765 (WRDPRP1s). 2 pronombres en primera persona en plural (Nosotros y nos) darán como resultado 42.553 (WRDPRP1p). 1 pronombre en segunda persona en singular (Tú) dará como resultado 21.2765 (WRDPRP2s). 1 pronombre en segunda persona en plural (ustedes) dará como resultado 21.2765 (WRDPRP2p). 3 pronombres en tercera persona en singular (Ella, ella, él) darán como resultado 63,829

(WRDPRP3s). Por último, 2 pronombres en tercera persona en plural (Ellos, ellas) darán como resultado 42.553 (WRDPRP3p).

El cuarto conjunto de métricas corresponden a los índices de densidad de patrones sintácticos, el cual implementa 3 de las 48 métricas a implementar.

Figura 8. Índices de densidad de patrones sintácticos.

Estos índices se obtienen luego de aplicar la siguiente fórmula:

$$(cantidad\ patrón\ sintáctico / total\ de\ palabras) * 1000$$

En la Figura 8 se aprecian los resultados del procesamiento del texto mostrado en la Figura 4. En el texto analizado, solo hay una frase verbal (Debieron salir), por lo que el índice es 21.2765 (DRVP). La cantidad de expresiones negativas es 4 (no, no, no y jamás) por lo que el índice es 85.1063 (DRNEG). Por último, la cantidad de frases nominales halladas por Spacy es 16 (Ellos, todo el día, ellas, en el juego, Yo, con el hermoso gato, A nosotros, nos, el gato, Ella, mascotas, ella, plantas y él no, Tú, ustedes, en la mañana), por lo que el índice es 340.4255 (DRNP).

El quinto conjunto de métricas corresponden a los índices de complejidad sintáctica, el cual implementa 2 de los 48 métricas a implementar.

Figura 9. Índices de complejidad sintáctica.

En la Figura 9 se aprecian los resultados del procesamiento del texto mostrado en la Figura 4. En el texto analizado, existen 16 frases nominales identificadas, de las cuales, una de ellas posee un modificador (Adjetivo). Esta frase nominal es "Con el hermoso gato". Al calcular el promedio, resulta un valor de 0,0625 (SYNNP). Por último, todas las oraciones tienen en promedio 1.857 palabras antes del verbo principal de la oración (SYNLE).

El sexto conjunto de métricas corresponden a los índices de diversidad léxica, el cual implementa 2 de las 48 métricas a implementar.

Figura 10. Índices de densidad de diversidad léxica.

En la Figura 10 se aprecian los resultados del procesamiento del texto mostrado en la Figura 4. En el texto analizado, la cantidad de palabras únicas es 39. Sabiendo que en total hay 47 palabras, esto, al dividir, da como resultado un radio de 0.8297 (LDTTRa). Por otro lado, la cantidad de palabras únicas de contenido, como sustantivos, verbos, adjetivos y adverbios, es de 19 mientras que la cantidad total de aquellas es de 23, lo que da como resultado, al dividir, un radio de 0.8260 (LDTTRcw).

El séptimo conjunto de métricas corresponden a los índices de cohesión referencial, el cual implementa 12 de las 48 métricas a implementar.

Figura 11. Índices de cohesión referencial.

En la Figura 11 se aprecian los resultados del procesamiento del texto mostrado en la Figura 4. Para obtener estas métricas, se analizaron las oraciones dos formas:

La primera forma es analizando pares de oraciones contiguas, ignorando la separación entre párrafos (Graesser et al., 2004; McNamara et al., 2014). Con ello, en la Tabla 12, se obtienen los siguientes pares de oraciones:

Tabla 12. Pares de oraciones contiguas.

Oración 1	Oración 2
Ellos jugaron todo el día.	Asimismo, ellas participaron en el juego.
Asimismo, ellas participaron en el juego.	Yo corro con el hermoso gato.
Yo corro con el hermoso gato.	A nosotros no nos gusta el gato.
A nosotros no nos gusta el gato.	Ella tiene mascotas.

Ella tiene mascotas.	Aunque, ella tiene plantas y él no.
Aunque, ella tiene plantas y él no.	Tu jamás dijiste que no, porque ustedes debieron salir temprano al mismo tiempo.

Para hallar cohesión referencial entre sustantivos, un par de oraciones tiene que tener exactamente el mismo sustantivo escrito de la misma manera (Dascalu et al., 2018; Graesser et al., 2004) y en el texto de ejemplo, el tercer par de oraciones es el único que cumple con dicha condición y, al haber 6 pares de oraciones, luego de calcular el promedio, da como resultado 0.1666 (CRFNO1). La cohesión referencial entre argumentos se da cuando un par de oraciones posee al menos un pronombre o un sustantivo, en plural o singular, igual en ambas oraciones (Graesser et al., 2004; McNamara et al., 2014). En el ejemplo, el tercer y quinto par poseen estas características y, con ello, al calcular el promedio, se obtiene 0.3333 (CRFAO1). La cohesión referencial entre raíces se da cuando, en un par de oraciones, un sustantivo de una oración coincide con una palabra de contenido de una oración anterior, haciendo comparaciones entre lemmas (Graesser et al., 2004; McNamara et al., 2014). En el ejemplo, el primer y tercer par cumplen con esa condición, dando un promedio de 0.3333 (CRFSO1). La cohesión referencial entre palabras de contenido se da calculando la proporción de palabras de contenido que coinciden en un par de palabras, dividiéndolo por la cantidad total de palabras de contenido en dicho par (Graesser et al., 2004; McNamara et al., 2014). En el ejemplo, el tercer par tiene una proporción de 0.333, el quinto par 0.4 y el sexto par 0.2. Estos tres valores dan como promedio 0.1555 (CRFCWO1) con una desviación estándar de 0.1662 (CRFCWO1d). La cohesión referencial entre anáforas se da cuando un par de oraciones tienen al menos un pronombre igual en ambas oraciones (Graesser et al., 2004; McNamara et al., 2014). En el texto analizado, el quinto par cumple con dicha condición, dando un valor de 0.1666 (CRFANP1).

La segunda forma se obtiene al analizar todas las combinaciones posibles de oraciones en el texto (Graesser et al., 2004; McNamara et al., 2014). Con ello, en la Tabla 13, se obtienen los siguientes pares de oraciones:

Tabla 13. Pares de oraciones todas las oraciones posibles.

Oración 1	**Oración 2**
Ellos jugaron todo el día.	Asimismo, ellas participaron en el juego.
Ellos jugaron todo el día.	Yo corro con el hermoso gato.

Ellos jugaron todo el día.	A nosotros no nos gusta el gato.
Ellos jugaron todo el día.	Ella tiene mascotas.
Ellos jugaron todo el día.	Aunque, ella tiene plantas y él no.
Ellos jugaron todo el día.	Tú jamás dijiste que no, porque ustedes debieron salir temprano al mismo tiempo.
Asimismo, ellas participaron en el juego.	Yo corro con el hermoso gato.
Asimismo, ellas participaron en el juego.	A nosotros no nos gusta el gato.
Asimismo, ellas participaron en el juego.	Ella tiene mascotas.
Asimismo, ellas participaron en el juego.	Aunque, ella tiene plantas y él no.
Asimismo, ellas participaron en el juego.	Tú jamás dijiste que no, porque ustedes debieron salir temprano al mismo tiempo.
Yo corro con el hermoso gato.	A nosotros no nos gusta el gato.
Yo corro con el hermoso gato.	Ella tiene mascotas.
Yo corro con el hermoso gato.	Aunque, ella tiene plantas y él no.
Yo corro con el hermoso gato.	Tú jamás dijiste que no, porque ustedes debieron salir temprano al mismo tiempo.
A nosotros no nos gusta el gato.	Ella tiene mascotas.
A nosotros no nos gusta el gato.	Aunque, ella tiene plantas y él no.
A nosotros no nos gusta el gato.	Tú jamás dijiste que no, porque ustedes debieron salir temprano en la mañana.
Ella tiene mascotas.	Aunque, ella tiene plantas y él no.
Ella tiene mascotas.	Tú jamás dijiste que no, porque ustedes debieron salir temprano al mismo tiempo.
Aunque, ella tiene plantas y él no.	Tú jamás dijiste que no, porque ustedes debieron salir temprano al mismo tiempo.

Para hallar la cohesión referencial entre sustantivos, el par N.º 12 cumple los criterios, dando un promedio de 0.0476 (CRFNOa). Los pares N° 12 y 19 cumplen los criterios de cohesión referencial entre argumentos, lo que da un promedio de 0.0952 (CRFAOa). Los pares de oraciones N° 1 y 12 cumplen con los criterios de cohesión referencial entre raíces, lo que da un promedio de 0.0952 (CRFSOa). Para hallar la cohesión referencial entre palabras de contenido, el par N° 12 tiene una proporción de 0.3333; el par N° 17, 0.3333; el par N.º 18, 0.2; el par N.º 19, 0.4 y el par N.º 21, 0.2. Aquello da como promedio 0.069 (CRFCWOa) y con una desviación estándar de 0.1308 (CRFCWOad). El par N.º 19 cumple con las condiciones de la cohesión referencial de anáforas, dando un promedio de 0.0476 (CRFANPa).

El octavo conjunto de métricas corresponden a los índices de cohesión referencial, el cual implementa 6 de las 48 métricas a implementar.

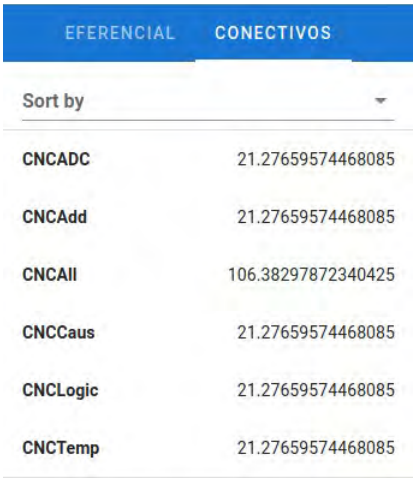

Figura 12. Índices de conectivos

Estos índices se obtienen luego de aplicar la siguiente fórmula:

$$(cantidad\ de\ conectivos / total\ de\ palabras) * 1000$$

En la Figura 12 se aprecian los resultados del procesamiento del texto mostrado en la Figura 4. En el texto analizado, la cantidad de conectivos adversativos es 1 (Aunque), dando un índice de 21.2765 (CNCADC). Además, la cantidad de conectivos de adición es 1 (Asimismo), dando un índice de 21.2765 (CNCAdd) . También, la cantidad de conectivos de causa es 1 (Porque), dando un índice de 21.2765 (CNCCaus). Por otro lado, la cantidad de conectivos lógicos es 1 (Y), dando un índice de 21.2765 (CNCLogic). Asimismo, la cantidad de conectivos temporales es 1 (al mismo tiempo), dando un índice de 21.2765 (CNCTemp). Por último, la cantidad de conectivos totales es 5, la cual es la suma de todos los conectivos anteriores, dando un índice de 106.3829 (CNCAll). Las especificaciones de que palabras o frases son consideradas para cada tipo de conectivos se encuentran en Anexo E, en donde se encuentra el listado de todas los índices implementados con sus respectivas métricas especificados de manera más detallada.

El código fuente de la implementación de esta herramienta se encuentra en el siguiente repositorio de Github: https://github.com/Hans03430/TextComplexityAnalyzerCM. Además, la implementación de cada métrica implementada, así como la documentación de la herramienta y el conjunto de algoritmos que implementan dichas métricas se encuentran en el Anexo F, en donde se encuentran de manera más detallada.

4.2.3 Un módulo de clasificación de complejidad textual parametrizado al corpus de textos en español

Luego de procesar los 183 textos en español obtenidos en el Capítulo 5, se usaron dichos datos procesados para realizar el experimento de aprendizaje supervisado con los datos de los textos procesados en el punto 5.2.3. También, la cantidad de datos disponible se encuentra en Tabla 18 en el punto 5.2.2. Además, los modelos de aprendizaje supervisado, entrenados mediante Grid Search para encontrar los mejores parámetros, son los siguientes:

- Regresión Logística (Logistic Regression).
- Máquina de soporte de Vectores (Support Vector Machine).
- Árboles de decisión (Decision Tree).
- Bosque aleatorio (Random Forest).
- K Vecinos más próximos (K Nearest Neighbors).
- Perceptron.
- Ada Boost.
- Gradient Boosting.
- Voting Classifier (Voting) (Se usaron todos los modelos anteriores, excepto el perceptron ya que no implementa el método 'predict_proba()', que necesita este clasificador).
- Stacking Classifier (Stacking) (Se consideraron los mismos modelos que para el modelo Voting Classifier).

Al realizar dicha experimentación, se obtuvieron los siguientes resultados, en la Tabla 14, al momento de entrenar los modelos usando todo el conjunto de datos disponible.

Tabla 14. Entrenamiento sobre todos los textos.

	Ajustado sobre F1 Score				**Ajustado sobre Accuracy**			
Modelo	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Logistic	0.89	0.933	0.82	0.875	0.88	0.933	0.82	0.875

Regression								
Support Vector Machine	0.892	0.882	0.882	0.882	0.885	0.882	0.882	0.882
Decision Tree	0.89	0.842	0.94	0.889	0.89	0.857	0.71	0.774
Random Forest	0.865	0.875	0.824	0.848	0.853	0.875	0.824	0.848
K Nearest Neighbors	0.92	0.938	0.88	0.909	0.91	0.938	0.88	0.909
Perceptron	0.892	0.933	0.824	0.875	0.881	0.531	1.000	0.694
Ada Boost	0.81	0.857	0.71	0.774	0.79	0.857	0.71	0.774
Gradient Boosting	0.892	0.842	0.941	0.889	0.891	0.882	0.882	0.882
Voting	0.89	0.882	0.88	0.882	0.88	0.882	0.88	0.882
Stacking	0.892	0.882	0.882	0.882	0.885	0.882	0.882	0.882
Promedio	0.884	0.89	0.859	0.87	0.875	0.85	0.847	0.84

En la Tabla 14 se observan los resultados obtenidos para Accuracy, Precision, Recall y F1 Score son, en promedio para todos los modelos usados, mejores cuando los modelos fueron ajustados sobre la métrica F1 Score. Además, para entrenar los modelos, se normalizaron los datos usando el escalador MinMaxScaler de scikit learn, además de una división estratificada del conjunto de entrenamiento y prueba, usando un 20% de los datos para entrenamiento.

El mismo experimento fue reproducido usando solo los textos los textos procesados pero divididos por dominio o categoría.

En la Tabla 15, se aprecian los resultados obtenidos para Accuracy, Precision, Recall y F1 Score para los textos de “Comunicación”, donde se observa que los resultados son mejores que al usar todos los textos descargados y procesados. Ello se debe a que los textos de distintos dominios o categorías no necesariamente tienen el mismo estilo de redacción.

Tabla 15. Entrenamiento sobre los textos de Comunicación.

	Ajustado sobre F1 Score				**Ajustado sobre Accuracy**			
Modelo	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.95	0.889	1	0.941	0.95	0.889	1	0.941
Support Vector Machine	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941

Decision Tree	0.9	0.800	1	0.889	0.9	0.800	1	0.889
Random Forest	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
K Nearest Neighbors	0.95	0.889	1	0.941	0.95	0.889	1	0.941
Perceptron	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
Ada Boost	0.95	1.000	0.88	0.933	1	1.000	1	1.000
Gradient Boosting	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
Voting	0.95	0.889	1	0.941	0.95	0.889	1	0.941
Stacking	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
Promedio	0.948	0.89	0.988	0.94	0.952	0.89	1.000	0.94

En la Tabla 16, se aprecian los resultados obtenidos para los textos de “Historia, Geografía y Economía”.

Tabla 16. Entrenamiento sobre los textos de Historia, Geografía y Economía.

	Ajustado sobre F1 Score				**Ajustado sobre Accuracy**			
Modelo	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
Support Vector Machine	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
Decision Tree	0.905	0.800	1.000	0.889	0.905	0.800	1.000	0.889
Random Forest	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
K Nearest Neighbors	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
Perceptron	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
Ada Boost	0.952	1.000	0.875	0.933	1.000	1.000	1.000	1.000
Gradient Boosting	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
Voting	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
Stacking	0.952	0.889	1.000	0.941	0.952	0.889	1.000	0.941
Promedio	0.948	0.891	0.988	0.935	0.952	0.891	1.000	0.942

Por último, se realizó el entrenamiento de los modelos sobre los textos de Ciencia y Tecnología/CTA. Los resultados están en la Tabla 17.

Tabla 17. Entrenamiento sobre los textos de Ciencia y Tecnología.

	Ajustado sobre F1 Score				Ajustado sobre Accuracy			
Modelo	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.500	0.000	0.000	0.000	0.500	0.000	0.000	0.000
Support Vector Machine	0.333	0.000	0.000	0.000	0.333	0.000	0.000	0.000
Decision Tree	0.500	0.000	0.000	0.000	0.333	0.000	0.000	0.000
Random Forest	0.500	0.000	0.000	0.000	0.500	0.000	0.000	0.000
K Nearest Neighbors	0.500	0.333	0.500	0.400	0.500	0.333	0.500	0.400
Perceptron	0.500	0.000	0.000	0.000	0.500	0.000	0.000	0.000
Ada Boost	0.333	0.000	0.000	0.000	0.333	0.000	0.000	0.000
Gradient Boosting	0.500	0.333	0.500	0.400	0.333	0.000	0.000	0.000
Voting	0.333	0.000	0.000	0.000	0.333	0.000	0.000	0.000
Stacking	0.500	0.000	0.000	0.000	0.333	0.000	0.000	0.000
Promedio	0.450	0.067	0.100	0.080	0.400	0.033	0.050	0.040

Para este entrenamiento, los resultados son los peores de los tres debido a que la cantidad de los datos disponibles de entrenamiento, contando solo los textos procesados de Ciencia y Tecnología es de 29 documentos, cantidad no adecuada para entrenar modelos de clasificación. Sin embargo, con los resultados del entrenamiento usando todos los documentos, mostrados en la tabla Tabla 14 se observa que se superó el umbral mínimo de 75% para Accuracy, Precision, Recall y F1 Score. Además, con los resultados de dicha tabla se eligío el mejor modelo de clasificación a utilizar para extender la funcionalidad de la herramienta implementada en el punto 4.2.2, siendo este el modelo K Nearest Neighbors. Con ello, se otorga a la herramienta implementada la capacidad de predecir la complejidad textual de un texto en español, en este caso el texto de la Figura 4 en el capítulo 4.2.2. Dicha predicción se observa en la Figura 13.

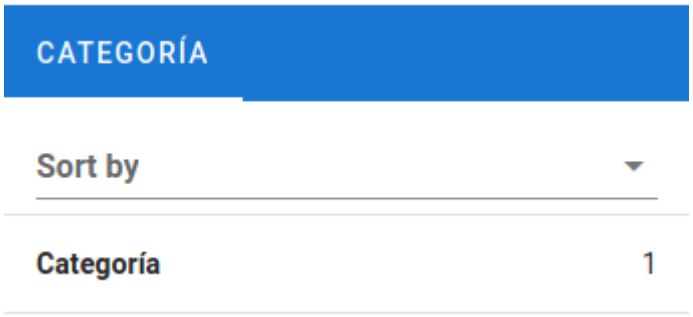

Figura 13. Predicción de la categoría del texto de prueba.

Esa nueva capacidad otorgada a la herramienta, implementada en un método de la clase TextComplexityAnalyzer, permite también usar modelos de clasificación distintos al implementado por defecto, K Nearest Neighbors, lo cual generaliza el funcionamiento de dicha funcionalidad. La implementación de este módulo puede ser encontrado en el Anexo F y los experimentos de aprendizaje supervisado, así como el código fuente del mejor modelo de clasificación y su validación pueden ser encontrados en el Anexo I.

4.3 Discusión

En el punto 4.2.1 se obtuvieron 48 métricas, que si bien no son todas las 108 implementadas por Coh-metrix en su versión web (Graesser et al., 2004; McNamara et al., 2014), son 3 más de las implementadas en la adaptación en lenguaje java a español, Coh-metrix-esp (Quispesaravia et al., 2016b). Dichas 48 métricas, que además de estar de acorde a las métricas de Coh-Metrix, también están de acorde a otras métricas descriptivas y de legibilidad encontradas (Crossley et al., 2019; López-Anguita et al., 2018; Ojha et al., 2018; Solovyev et al., 2018; Trotzek et al., 2017).

Además, la presente implementación de las métricas de complejidad textual utiliza el lenguaje de programación Python que, a diferencia de otras herramientas de análisis de complejidad como AzterTest (Bengoetxea et al., 2020), utiliza multiprocesamiento para agilizar la velocidad de procesamiento de los textos en la mayoría de las métricas, siendo los índices de cohesión referencial los únicos que no pudieron ser paralelizados.

Por último, los resultados obtenidos durante la experimentación con el módulo de clasificación para el mejor modelo usado, si bien no pueden ser comparados con los resultados de los experimentos de Coh-Metrix ni AzterTest debido a que aquellos fueron realizados con textos en inglés, son mejores que los puntajes obtenidos con el mejor modelo empleado para el experimento de Coh-Metrix-esp, donde el mejor modelo de Regresión Logística obtuvo un 0.84 de Precision, 0.82 de Recall y 0.82 de F1 Score, comparado con un 0.93 de Precision, 0.88 de Recall y 0.90 de F1 Score del mejor modelo obtenido en esta investigación, el cual es el K Nearest Neighbors.

Luego de encontrar el mejor modelo de clasificación, se introdujo dicho modelo en la herramienta implementada, otorgando así la capacidad de predecir la categoría (Complejidad textual) de nuevos textos en español, aunque para las clases con las que fue entrenado (1 y 2). Sin embargo, dicho método que calcula la complejidad textual fue

implementado en tal manera que es capaz de aceptar modelos de clasificación entrenados distintos al usado, para ser usado de manera personalizada.

Por último, la herramienta implementada puede ser usado en distintos tipos de texto en español, no solo de tipo académico. También, la herramienta implementada sería capaz de funcionar en otros idiomas además del español de ser implementadas las reglas para analizar textos en aquellos idiomas, aunque dichos idiomas serían aquellos que la librería Spacy puede procesar.

Capítulo 5. Corpus de textos en español

5.1 Introducción

El presente capítulo está dirigido al Objetivo Específico 2 (Constituir un corpus de más de cien textos en español clasificados por su complejidad textual).

El corpus de textos en español es creado para ser utilizado para entrenar el modelo de clasificación del Objetivo Específico 1 y para luego entrenar el Chatbot, en el Objetivo Específico 3.

A continuación, se presentan los tres resultados esperados: El conjunto de textos en español, la validación de la clasificación de dichos textos y los textos procesados.

5.2 Resultados Esperados

5.2.1 Un conjunto de textos en español seleccionados por su complejidad textual

Para recopilar los textos en español, se utilizó la página web conocida como www.perueduca.pe. En dicha página se puede encontrar diverso material educativo en español para alumnos de inicial, primaria y secundaria. Sin embargo, para obtener el presente corpus de textos en español, solo se consideraron los textos para primaria y secundaria. De ellos, se recopilaron los libros, infografías, fascículos y textos del Ministerio de Educación de Comunicación (MED) para las áreas de Ciencia y tecnología/CTA, y Personal Social/Historia, Geografía y Economía. De aquellos textos obtenidos, sólo se utilizaron aquellos dirigidos a los alumnos, ya que los textos para los docentes no están dirigidos para los propios estudiantes de primaria y secundaria.

Los documentos encontrados de tipo fascículos, libros y textos del MED fueron descargados automáticamente usando Python y Jupyter Notebook, empleando la técnica de Web scraping, realizando así el proceso de descarga de archivos PDF. Las infografías, siendo presentaciones en Adobe Flash, se recopilaron de manera manual y se los convirtió en archivos PDF. Durante la descarga de archivos, se pueden encontrar unos pocos que fueron escaneados, lo cual requiere realizar un procesamiento OCR para poder procesarlos. Para ello, se puede usar la librería “ocrmypdf” para realizar la conversión. Una vez recopilados todos los documentos, se los convirtió archivos de tipo TXT, usando la librería “python-tika”, para luego ser guardados en una base de datos SQLite automáticamente usando Python y Jupyter Notebook.

A continuación en la Figura 14 se tiene un archivo PDF descargado de la página web mencionada.

Figura 14. PDF descargado.

Ahora en la Figura 15 se tiene un archivo TXT convertido del archivo PDF.

```
Augusto B. Leguia

286 - Augusto B. Leguía

Biografías

El señor del "Oncenio"

Fue un presidente que acomodó las celebraciones del centenario de la República a sus
particulares intereses y que de su condición
de "maestro de la juventud" pasó a convertirse en un temido dictador. Sin embargo, sus
actos y decisiones han trazado muchos aspectos de la vida nacional.

1863-1932
```

Figura 15. Muestra de PDF convertido a TXT.

Por último en la Figura 16, se muestra un registro de la base de datos en las cuales fueron guardadas el contenido de los archivos descargados.

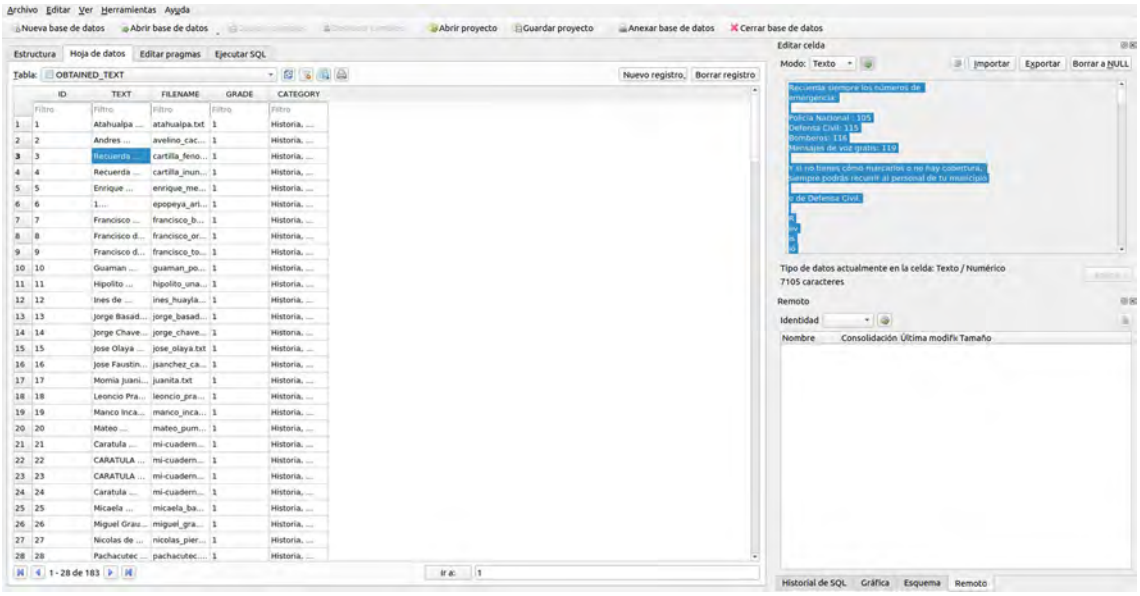

Figura 16. Registro de ejemplo de la base de datos.

En la imagen se puede apreciar un texto seleccionado de toda la base de datos, la cual contiene 183 registros. La columna “Grade” indica el nivel de complejidad textual del texto descargado, donde 1 es “Primaria” y 2 es “Secundaria”. Para mayor información sobre la base de datos, así como su diccionario de datos, revisar el Anexo G.

5.2.2 Validación del agrupamiento de los textos en español seleccionados por su complejidad textual

Luego de recopilar los textos en español y almacenarlos en una base de datos SQLite, se procedió a hacer un análisis exploratorio de datos, en la cual se obtuvieron los siguientes resultados, presentados en la Tabla 18.

Tabla 18. Cantidad de textos.

Dominio	Primaria (Grade = 1)	Secundaria (Grade = 2)	Total
Comunicación	39	62	101
Ciencia y Tecnología	12	17	29
Historia, Geografía y economía	34	19	53
Total	85	98	183

Luego de ello, se realizó, se utilizó la herramienta implementada en el capítulo 4.2.2 para procesar los textos, obteniendo las 48 métricas que fueron usadas como características para el análisis. Con ello, se obtuvo un gráfico de agrupamiento para visualizar la distribución de todos los 183 los documentos descargados en un espacio de dos dimensiones, el cual se observa en Figura 17:

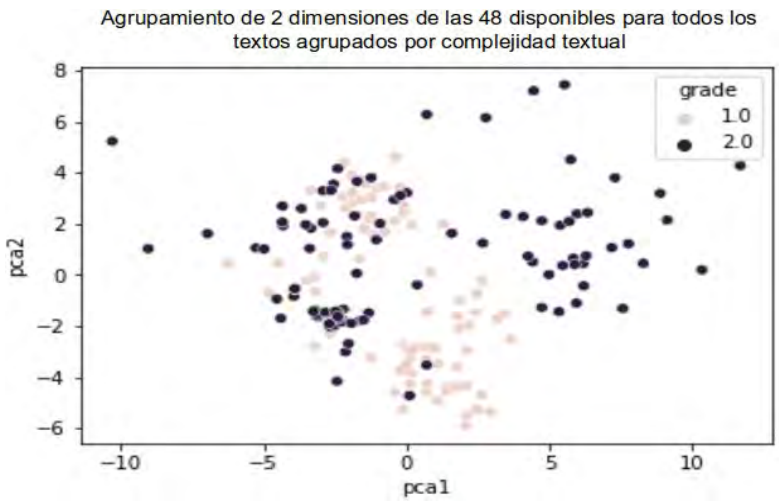

Figura 17. Gráfico de agrupamiento de 2 dimensiones de las 48 para todos los textos agrupados por complejidad textual: Primaria (1) y Secundaria (2).

Para visualizar las 48 características, se realizó un análisis de componentes principales, PCA por sus siglas en inglés, para realizar la reducción a 2 dimensiones, para que sea posible visualizar el conjunto de datos. Debido a la reducción de dimensionalidad, se pierde la información que todas las características poseen, lo cual ocasiona distorsión en la representación gráfica del conjunto de datos. Ello causa que se observe un solapamiento para los dos grados (1=Primaria, 2=Secundaria). Sin embargo, aquello no significa que el conjunto de datos esté mal clasificado, sino que la reducción de dimensiones causa una distorsión en la visualización.

Para demostrar que existe un patrón entre los distintos textos que forman el conjunto de datos, se decidió realizar la técnica de Clustering, usando el algoritmo KMeans. Este algoritmo se encargará de encontrar algún patrón en los datos, aunque no necesariamente el mismo que ya posee dicho conjunto de datos, el cual es el la complejidad textual por nivel Primaria y Secundaria. Para empezar, se creó un gráfico de inercia, o suma de errores cuadrados, por número de clases o clusters, más conocido como gráfico de codo, el cual se observa en la Figura 18.

Dicho gráfico indica la cantidad de clases, o clusters, que el algoritmo KMeans encuentra durante una ejecución iterativa desde 1 hasta 20 clases. En dicho gráfico se identifica que la cantidad adecuada de clases descubiertas es de 2, 3 o 4, coordenadas en las cuales se observa el cambio en la curva, formando la estructura de un “codo”, de ahí el nombre del gráfico. Dicha parte en la curva es usada debido a que, luego de dichos

tres coordenadas encontradas, la suma de los errores cuadrados no varía mucho luego de esa cantidad de clusters, lo que indica que se están considerando más clases de lo necesario.

Como se posee originalmente dos niveles de complejidad textual (1=Primaria, 2=Secundaria), se eligió la misma cantidad de clases para completar el experimento de Clustering, el cual se visualiza en la Figura 19:

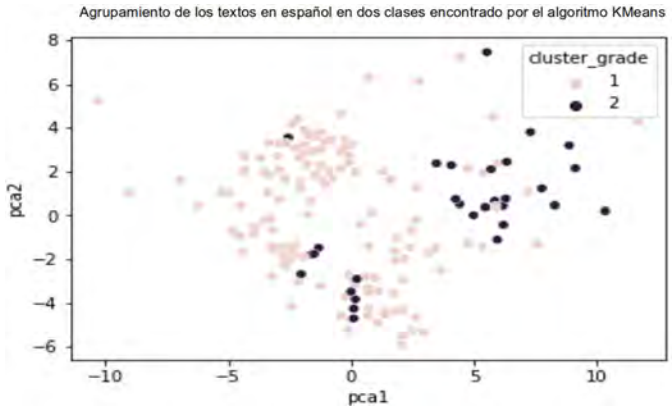

Figura 19. Gráfico en dos dimensiones de agrupamiento de las clases halladas por el algoritmo KMeans donde se demuestra que el conjunto de datos es divisible en dos clases.

Si bien la distribución hallada no es la misma que la de la distribución original con las clases originales (Primaria y secundaria), se observa que hay un patrón que permite dividir el conjunto de datos en dos clases. En este caso, el gráfico de la distribución del conjunto de datos hecho mediante la reducción de dimensionalidad a 2 dimensiones de las 48 no indica que los datos no sean clasificables. Esto se demuestra con el resultado del algoritmo de clustering KMeans, ya que fue capaz de encontrar los patrones para dividir el conjunto de datos en dos clases, aunque dicho conjunto visualizado parezca que no se puede dividir, como se observó en la Figura 17. Con dicha verificación de la divisibilidad, se identifica que es posible aplicar algoritmos de Machine Learning de aprendizaje supervisado (Clasificación).

Aparte del experimento de Clustering presentado, se realizó la misma experimentación pero sobre cada dominio que se tiene para el conjunto de documentos (Ciencia y Tecnología; Comunicación; Historia, Geografía y Economía), el cual se encuentra en el Anexo H, anexo en el que se encuentra todo el experimento realizado para este

resultado. En dichos experimentos extra se encontraron resultados similares a los encontrados durante el el Clustering realizado con todos los documentos.

5.2.3 Un corpus de textos en español procesados y clasificados por su complejidad textual

Una vez guardados los 183 documentos descargados en una base de datos SQLite, expuesto en el punto 5.2.1, así como luego de terminada la herramienta que calcula métricas de complejidad textual para textos en español, en el punto 4.2.2, se procedió a calcular las 48 métricas implementadas en dicha herramienta para todos los documentos de la base de datos. Dichos resultados fueron guardados en la base de datos SQLite, enlazados a los registros correspondientes a cada documento. Cada grupo de métricas, los cuales en la implementación de la herramienta son cada tipo de índices, fue guardado en una tabla que agrupa cada tipo, dando un total de 8 tablas extra en la base de datos. Dichos datos numéricos, así como las categorías asignadas a cada documento (1=Primaria, 2=Secundaria) se usan para entrenar los modelos de aprendizaje de máquina en el punto 4.2.3.

En la Figura 20 se aprecia una tabla de la base de datos SQLite que contiene los índices descriptivos almacenados luego de ser calculados por la herramienta del punto 4.2.2. Estos índices, los cuales demoraron 12 horas, 10 minutos y 32 segundos en ser procesados en su totalidad, son luego utilizados en el punto 4.2.3.

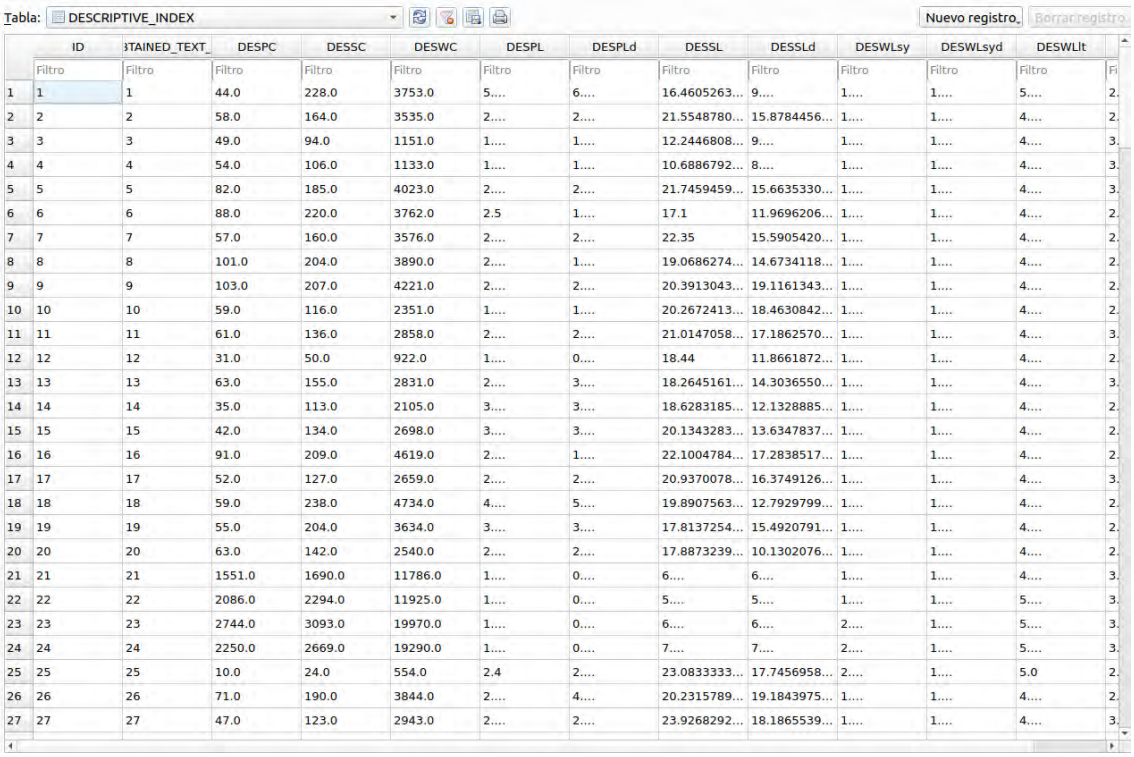

Tabla: DESCRIPTIVE_INDEX — Nuevo registro — Borrar registro

	ID	STAINED_TEXT_	DESPC	DESSC	DESWC	DESPL	DESPLd	DESSL	DESSLd	DESWLsy	DESWLsyd	DESWLlt	
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Fi
1	1	1	44.0	228.0	3753.0	5....	6....	16.4605263...	9....	1....	1....	5....	2.
2	2	2	58.0	164.0	3535.0	2....	2....	21.5548780...	15.8784456...	1....	1....	4....	2.
3	3	3	49.0	94.0	1151.0	1....	1....	12.2446808...	9....	1....	1....	4....	3.
4	4	4	54.0	106.0	1133.0	1....	1....	10.6886792...	8....	1....	1....	4....	3.
5	5	5	82.0	185.0	4023.0	2....	2....	21.7459459...	15.6635330...	1....	1....	4....	3.
6	6	6	88.0	220.0	3762.0	2.5	1....	17.1	11.9696206...	1....	1....	4....	2.
7	7	7	57.0	160.0	3576.0	2....	2....	22.35	15.5905420...	1....	1....	4....	2.
8	8	8	101.0	204.0	3890.0	2....	1....	19.0686274...	14.6734118...	1....	1....	4....	2.
9	9	9	103.0	207.0	4221.0	2....	2....	20.3913043...	19.1161343...	1....	1....	4....	2.
10	10	10	59.0	116.0	2351.0	1....	1....	20.2672413...	18.4630842...	1....	1....	4....	2.
11	11	11	61.0	136.0	2858.0	2....	2....	21.0147058...	17.1862570...	1....	1....	4....	3.
12	12	12	31.0	50.0	922.0	1....	0....	18.44	11.8661872...	1....	1....	4....	2.
13	13	13	63.0	155.0	2831.0	2....	3....	18.2645161...	14.3036550...	1....	1....	4....	3.
14	14	14	35.0	113.0	2105.0	3....	3....	18.6283185...	12.1328885...	1....	1....	4....	2.
15	15	15	42.0	134.0	2698.0	3....	3....	20.1343283...	13.6347837...	1....	1....	4....	2.
16	16	16	91.0	209.0	4619.0	2....	1....	22.1004784...	17.2838517...	1....	1....	4....	2.
17	17	17	52.0	127.0	2659.0	2....	2....	20.9370078...	16.3749126...	1....	1....	4....	3.
18	18	18	59.0	238.0	4734.0	4....	5....	19.8907563...	12.7929799...	1....	1....	4....	2.
19	19	19	55.0	204.0	3634.0	3....	3....	17.8137254...	15.4920791...	1....	1....	4....	2.
20	20	20	63.0	142.0	2540.0	2....	2....	17.8873239...	10.1302076...	1....	1....	4....	2.
21	21	21	1551.0	1690.0	11786.0	1....	0....	6....	6....	1....	1....	4....	3.
22	22	22	2086.0	2294.0	11925.0	1....	0....	5....	5....	1....	1....	5....	3.
23	23	23	2744.0	3093.0	19970.0	1....	0....	6....	6....	2....	1....	5....	3.
24	24	24	2250.0	2669.0	19290.0	1....	0....	7....	7....	2....	1....	5....	3.
25	25	25	10.0	24.0	554.0	2.4	2....	23.0833333...	17.7456958...	2....	1....	5.0	2.
26	26	26	71.0	190.0	3844.0	2....	4....	20.2315789...	19.1843975...	1....	1....	4....	2.
27	27	27	47.0	123.0	2943.0	2....	2....	23.9268292...	18.1865539...	1....	1....	4....	3.

Figura 20. Tabla DESCRIPTIVE_INDEX que contiene los datos procesados de los textos por la herramienta .

Los nombres de las tablas de la base de datos SQLite donde se guardaron las 48 métricas están en la Tabla 19.

Tabla 19. Tablas de la base de datos.

Tabla	Descripción
DESCRIPTIVE_INDEX	Tabla que contiene los índices descriptivos calculados por la herramienta de cálculo de complejidad textual para textos en español
CONNECTIVE_INDEX	Tabla que contiene los índices conectivos por la herramienta de cálculo de complejidad textual para textos en español
LEXICAL_DIVERSITY_INDEX	Tabla que contiene los índices de diversidad léxica calculados por la herramienta de cálculo de complejidad textual para textos en español
READABILITY_INDEX	Tabla que contiene los índices de legibilidad calculados por la herramienta de cálculo de complejidad textual para textos en español
REFERENTIAL_COHESION_INDEX	Tabla que contiene los índices de cohesión referencial calculados por la herramienta de cálculo de complejidad textual para textos en español
SYNTACTIC_COMPLEXITY_INDEX	Tabla que contiene los índices de complejidad sintáctica calculados por la herramienta de cálculo de complejidad textual para textos en español
SYNTACTIC_PATTERN_INDEX	Tabla que contiene los índices de patrones

	sintácticos calculados por la herramienta de cálculo de complejidad textual para textos en español
WORD_INFORMATION_INDEX	Tabla que contiene los índices de información de palabras por la herramienta de cálculo de complejidad textual para textos en español

La información correspondiente a estas tablas a manera de diccionario de datos, así como a la base de datos misma se encuentran en el Anexo G.

5.3 Discusión

Como se vio en el punto 5.2.1 se descargaron un total de 183 documentos en español seleccionados por su grado de complejidad textual, en este caso, primaria y secundaria (1 y 2), los cuales se usaron para realizar los distintos experimentos. Mientras que otras investigaciones usaron corpus y documentos recopilados en inglés para realizar sus propios experimentos (Aguirre et al., 2016; Aryadoust & Goh, 2014; Balakrishna, 2015; Bengoetxea et al., 2020; Bich, 2017; Choi, 2018b; Crossley et al., 2019; Dascalu et al., 2018; Davoodi et al., 2017; Fitzgerald et al., 2015; Latifi, 2016; Li & Liu, 2016; López-Anguita et al., 2018; S. Minhas & Hussain, 2016; S. Z. Minhas, 2016; Patout & Cordy, 2019; Stevanoski & Gievska, 2019; Trotzek et al., 2017; Vajjala, 2018; Vajjala & Lučić, 2018; Yaneva, 2016), en portugués (Branco et al., 2014; Hartmann et al., 2016; Wagner Filho et al., 2016), en ruso (Solovyev et al., 2018), en holandés (Kleijn, 2018) y chino (Sung et al., 2015; YAO–TING et al., 2015, p. 2). Para aquellas investigaciones que usaron textos en español, dichos documentos también fueron recopilados de manera similar, lo cual requirió descargar documentos previamente clasificados y crear un corpus de entrenamiento con dichos textos (López-Anguita et al., 2018; Quispesaravia et al., 2016b).

Luego, se demostró que los documentos descargados eran divisibles en dos grupos al realizar el experimento de Clustering, utilizando el algoritmo de KMeans, demostrando así que el conjunto de datos obtenido es apto para ser usado en experimentos de aprendizaje supervisado (Clasificación). Para realizar dicho experimento, se procesaron los documentos usando la herramienta del punto 4.2.2. Dichos datos procesados fueron almacenados en la base de datos SQLite para ser usados en el punto 4.2.3, donde se realiza el aprendizaje supervisado con los textos en español procesados.

Si bien los documentos obtenidos se descargaron en su mayoría de manera automática y algunos de manera manual de la página www.perueduca.pe, métodos similares podrían

ser usados para descargar otros documentos de otras fuentes en línea. Además, estos textos descargados podrían usarse para otros tipos de experimentos con textos en español que no necesariamente sean de clasificación de textos y entrenamiento de Chatbots, aunque dependería del tipo de experimento a realizar. Por último, no todos los archivos disponibles en dicha página fueron descargados ya que algunos archivos eran de tipo multimedia, en vez de ser archivos PDF.

Capítulo 6. Entrenamiento del Chatbot

6.1 Introducción

El presente capítulo está dirigido al Objetivo Específico 3 (Evaluar el impacto de los corpus generados en las respuestas de un *Chatbot*).

Para entrenar el *Chatbot* se utilizará el corpus obtenido en el Capítulo 5 Además, se utilizará, para identificar la complejidad textual de las respuestas del *Chatbot*, la herramientas de análisis de complejidad textual implementada en el Capítulo 4.

A continuación se presentan los resultados esperados: Selección de un *Chatbot* implementado para prueba; entrenamiento del *Chatbot* usando corpus de textos en español clasificados por su complejidad textual y Pruebas de validación de un *Chatbot* con al menos 3 tópicos del corpus clasificado automáticamente.

6.2 Resultados Esperados

6.2.1 Selección de un *chatbot* implementado para prueba.

Para realizar los experimentos siguientes, primero se tuvo que seleccionar un *Chatbot* a ser usado. Para ello, se buscó en repositorios, así como en tutoriales de como hacer *Chatbots*, implementaciones disponibles para ser usadas al momento de realizar la experimentación con el *Chatbot*. De lo investigado, se encontraron los siguientes links con candidatos a *Chatbots* en la Tabla 20:

Tabla 20. Candidatos de Chatbots.

Modelo/Tutorial	Link	Descripción
Practical Seq2Seq	https://github.com/suriyadeepan/practical_seq2seq	"Modelo de Sequence to Sequence para experimentar con datasets."
TF Chatbot Seq2Seq Antilm	https://github.com/Marsan-Ma-zz/tf_chatbot_seq2seq_antilm	"Chatbot que usa el modelo Sequence to sequence, implementa módulo de atención y modelos anti lenguages para evitar respuestas genéricas."
Chatterbot	https://github.com/gunthercox/ChatterBot	"Framework para crear Chatbots usando Machine Learning"
Creating a chatbot from scratch using keras and Tensorflow	https://medium.com/predict/creating-a-chatbot-from-scratch-using-keras-and-tensorflow-59e8fc76be79	"Tutorial de cómo crear Chatbots usando modelos LSTM Sequence to sequence."

Generative Chatbots using the Seq2Seq model	https://towardsdatascience.com/generative-chatbots-using-the-seq2seq-model-d411c8738ab5	"Tutorial de cómo crear Chatbots generacionales usando modelos de Sequence to Sequence."
Transformer chatbot tutorial with tensorflow 2	https://blog.tensorflow.org/2019/05/transformer-chatbot-tutorial-with-tensorflow-2.html	"Tutorial de como hacer un Chatbot generacional usando modelos Transformer."
Open Domain ChatBot	https://github.com/AbdelrahmanRadwan/Open-Domain-ChatBot	"Modelo de Chatbot de dominio abierto basado en un modelo de Deep Learning de Google."
Naive Bayes Chatbot	https://github.com/jackfrost1411/naive_bayes_chatbot	"Modelo de dominio cerrado usando Naive Bayes."
Seq2Seq Chatbot	https://github.com/AbrahamSanders/seq2seq-chatbot	"Chatbot generacional que utiliza el modelo Sequence to Sequence".

Los links que redireccionan a tutoriales de cómo construir *Chatbots* se tomaron en recopilaron por si ninguno de los *Chatbots* ya implementados funcionaba de manera adecuada para el experimento. Luego de recopilados dichos candidatos a *Chatbots*, se probó cuáles funcionaban de manera adecuada para el experimento a realizar. Para ello, se probo ejecutar los *Chatbots* recopilados, resultando en el fallo de algunos de ellos debido a librerías desactualizadas, incompatibilidad debido a no mostrar los resultados de entrenamiento correcto y errores de ejecución. Otros modelos funcionaron bien al ser probados, sin embargo carecían de los datos necesarios para realizar los experimentos siguientes. A continuación se listan los modelos de *Chatbots* que funcionaron con sus observaciones:

- https://github.com/gunthercox/ChatterBot: No muestra datos numéricos de entrenamiento para saber qué tan bien está siendo entrenado el modelo pero permite guardar el modelo para probarlo luego.
- https://github.com/AbrahamSanders/seq2seq-chatbot: Muestra los datos numéricos que indican que tan bien va el modelo y permite guardar el modelo para probarlo luego. También permite personalizar parámetros de entrenamiento y características del modelo del *Chatbot*.
- https://github.com/jackfrost1411/naive_bayes_chatbot: No muestra los datos numéricos de entrenamiento para saber qué tan bien está siendo entrenado el modelo.

La experimentación de prueba con los Chatbots se desarrolló en un documento de Jupyter Notebook, el cual se puede ubicar en el siguiente link:

https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/reports/Prueba_de_chatbots.ipynb.

Una vez terminado de probar los *Chatbots*, se eligió el repositorio con el *Chatbot* más adecuado, el cuál es el Modelo Seq2Seq implementado por Abraham Sanders (Sanders, 2018/2020).

La recopilación de *Chatbots* realizada se encuentra en el siguiente reporte realizado en Jupyter Notebook, ubicado en el link: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/reports/Selecci%C3%B3n%20de%20Chatbots.ipynb. La validación de lo realizado está en el archivo "Validación de la selección del chatbot.docx", que también se encuentra en el link: https://docs.google.com/document/d/1RIu9HZ8cZ9ik69BiFZrm71H0BRCwIz3nwjCH9hGHpQ0/edit?usp=sharing.

6.2.2 Entrenamiento del *chatbot* usando corpus de textos en español clasificados por su complejidad textual.

Para realizar un entrenamiento inicial del Chatbot se realizaron 6 entrenamientos de prueba. Para todos los experimentos se limpió el conjunto de datos, eliminando los signos de puntuación; además, se recopilaron los pares de oraciones de los documentos descargados en un archivo CSV, el cual es utilizado por el modelo de *Chatbot* en el punto 6.2.1 para realizar su entrenamiento. La métrica usada para la curva de error es denominada *Perplexity*, el cual mide que tanto se compara un modelo al conjunto de entrenamiento al medir que tan bien el modelo predice una muestra (Jurafsky & Martin, 2019). En la Tabla 21 se observan los parámetros usados durante el entrenamiento para los 6 experimentos. Cabe resaltar que el *Chatbot* fue entrenado en un servidor del grupo IA PUCP y que el análisis fue realizado en una laptop personal.

Tabla 21. Parámetros para las pruebas de entrenamiento del chatbot seleccionado.

Experimento	Complejidad Textual del conjunto de datos	Épocas	Optimizador	Tasa de aprendizaje	Porcentaje del conjunto de validación (%)
1	1	50	Adam	0.01	20
2	1	100	Adam	0.01	20
3	1	100	SGD	0.01	20
4	2	50	Adam	0.01	20
5	2	100	Adam	0.01	20

6	2	100	SGD	0.01	20

Además, la configuración del Chatbot se encuentra en la Tabla 22.

Tabla 22. Estructura del Chatbot,

Característica	Valor
Tipo de unidad de Red Neuronal Recurrente	Long-short Term Memory (LSTM)
Tamaño de Red Neuronal Recurrente	1024
Tipo de Encoder	Bidireccional
Número de capas para el Encoder	4
Número de capas para el Decoder	4
Tamaño de Embedding para el Encoder	128
Tamaño de Embedding para el Decoder	128
Tipo de Atención	Normed Bahdanau
Width para Beam Search	20

Luego de realizar el Experimento 1, se obtuvo en la Figura 21 la curva de pérdida del entrenamiento del *Chatbot* con los parámetros respectivos. Aquí se puede apreciar que está ocurriendo *Overfitting* durante el entrenamiento, debido a que el error para el conjunto de datos de validación es mayor a la del conjunto de datos de entrenamiento. Ello lleva a pensar que con una mayor cantidad de épocas el *Chatbot* puede obtener mejores resultados.

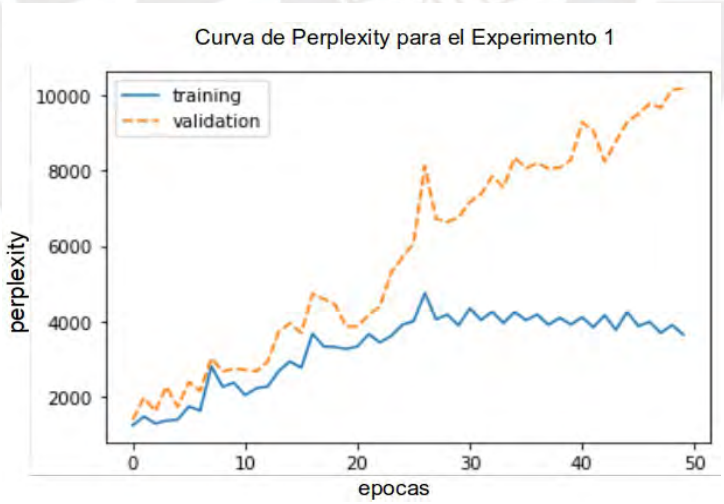

Figura 21. Curva de error del Experimento 1.

Para obtener mejores resultados, se realizó el Experimento 2, esta vez entrenando el Chatbot seleccionado por 100 épocas, con lo que se obtuvo en la Figura 22 la correspondiente curva de pérdida. En aquella gráfica se observa que el error del entrenamiento se estabiliza un poco durante muchas épocas, mientras que el error del conjunto de validación aumenta y disminuye. Ello indica que, para los parámetros

usados durante el entrenamiento, 100 épocas no genera muchos cambios a la hora de entrenar el modelo. Mas bien, se observa que el *Overfitting* se hace peor mientras más épocas se utilizan. Entonces, con lo descubierto, se cambió de optimizador, lo cual lleva al Experimento 3.

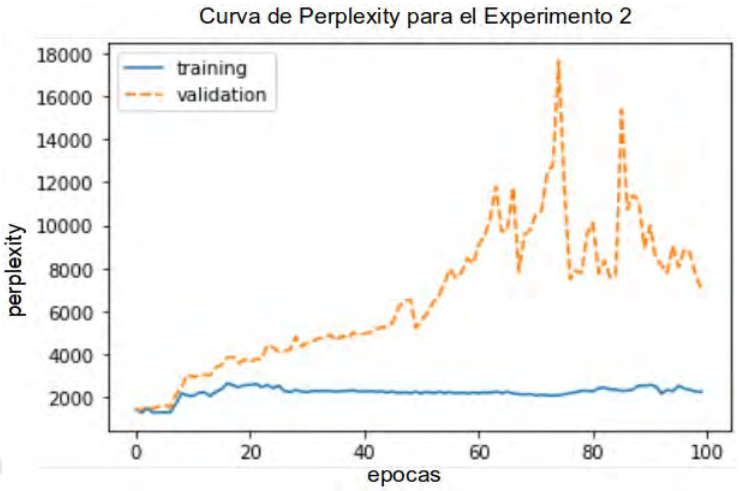

Figura 22. Curva de error del Experimento 2.

Para el Experimento 3 se utilizó, de la misma manera que en el Experimento 2, 100 épocas para entrenar el *Chatbot* seleccionado con el conjunto de datos de complejidad textual 1. En este caso, al usar el optimizador *Stochastic Gradient Descent* (SGD), se obtuvo la curva de pérdida de la Figura 23. En dicha figura se puede observar que el error en la validación y el entrenamiento son similares, lo que significa que no hay *Overfitting* como en los experimentos anteriores. Sin embargo, se observa que el modelo converge de mejor manera que usando el optimizador Adam, ello significa que, para el *Chatbot* encontrado, así como para el conjunto de datos de complejidad textual 1, SGD es el mejor optimizador para entrenar el *Chatbot*.

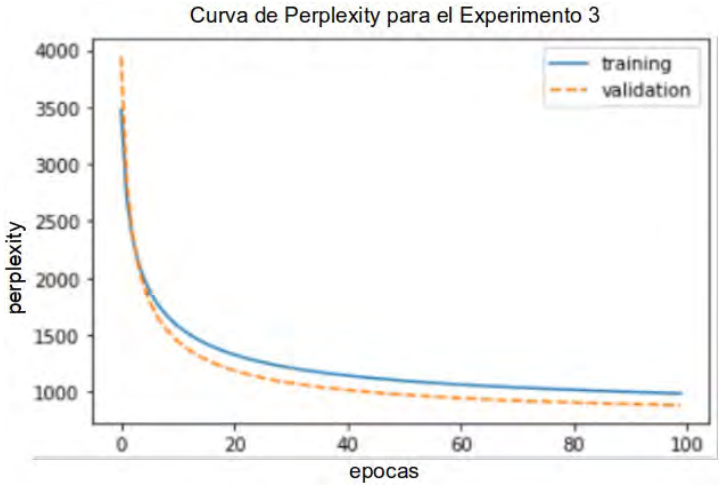

Figura 23. Curva de error del Experimento 3.

Se realizó el Experimento 4 con la finalidad de entrenar el *Chatbot* seleccionado sobre las oraciones del conjunto de datos de complejidad textual 2. Con ello, se obtuvo curva de error mostrada en la Figura 24. En dicha figura se aprecia, como en el Experimento 1, que al entrenar por 50 épocas usando el optimizador Adam ocurre *Overfitting*. Con ello, se realizó otro experimento usando más épocas, para verificar si el *Chatbot* dejaba de hacer *Overfitting*.

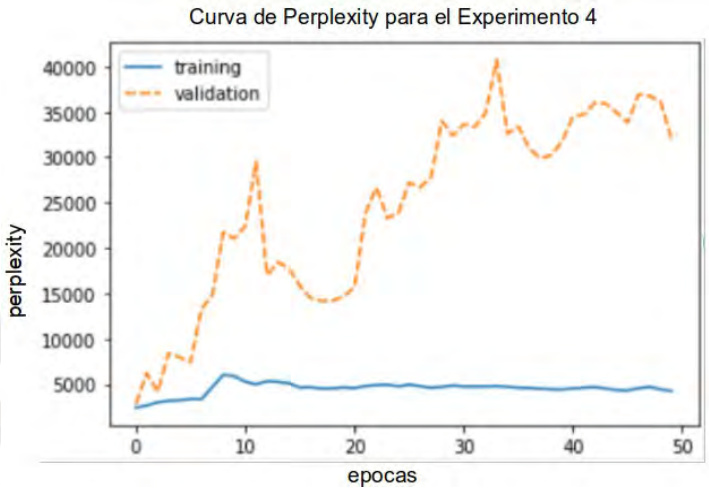

Figura 24. Curva de error del Experimento 4.

En el experimento 5 se entrenó el *Chatbot* por 100 épocas usando el optimizador Adam, obteniendo la curva de error de la Figura 25. En dicha figura se puede apreciar que, de manera similar al Experimento 2, al utilizar el optimizador Adam, el *Chatbot* no llega a tener un buen desempeño, terminando en *Overfitting*. Luego de ello, se realizó un último experimento con el optimizador SGD.

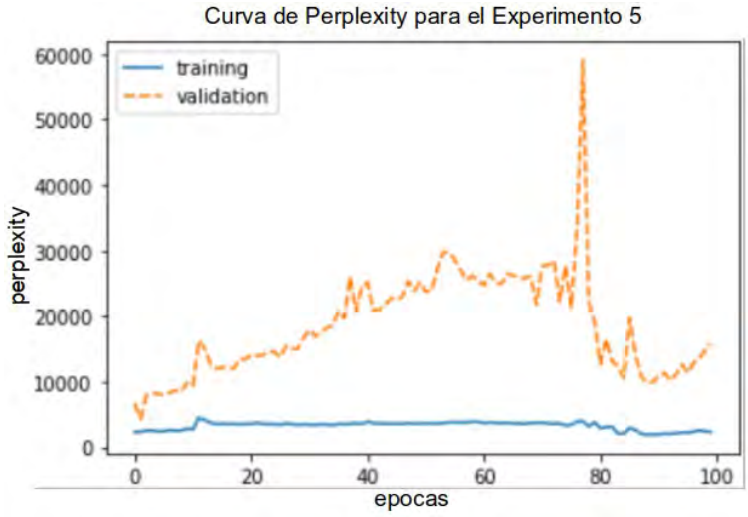

Figura 25. Curva de error del Experimento 5.

El último experimento, el Experimento 6, se realizó entrenando el *Chatbot* para las oraciones de complejidad textual 2 usando el optimizador SGD, obteniendo la curva de pérdida de la Figura 26. En dicha curva se observa que no hay *Overfitting*, debido a que ambas curvas de pérdida para entrenamiento y validación son similares. Con ello, se deduce que para entrenar el *Chatbot* con los textos de complejidad textual 2, el optimizador SGD es mejor que el optimizador ADAM.

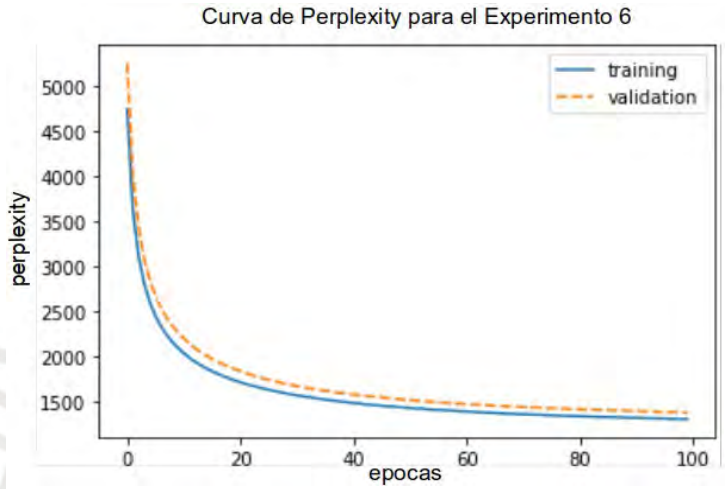

Figura 26. Curva de error del Experimento 6.

Como conclusión de los experimentos realizados, al analizar la Figura 26 y la Figura 23 se puede observar que en la primera figura, la pérdida de validación es mayor que en la segunda figura, con respecto a la pérdida de entrenamiento de ambas. Ello significa que para el *Chatbot* le es más fácil responder a las oraciones de complejidad textual 1 que a las de complejidad textual 2. El reporte con la experimentación realizada se encuentra en el siguiente link: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Entrenamiento%20de%20chatbot.ipynb. La validación de lo realizado se encuentra en el documento "Validación del entrenamiento del Chatbot con el conjunto de datos.docx", que también se encuentra en el link: https://docs.google.com/document/d/1BD4p3faWAFT2mRUznK1KdVOU9b3MZuBYACAS3zv1Zk8/edit?usp=sharing.

6.2.3 Pruebas de validación de un *chatbot* con al menos 3 tópicos del corpus clasificado automáticamente.

En esta parte del experimento con el *Chatbot*, se usó el modelo encontrado en el punto 6.2.1, el cual era un *Chatbot* generativo basado en la arquitectura Sequence 2 Sequence,

diseñado en Tensorflow (Sanders, 2018/2020). Dicho modelo de *Chatbot* fue entrenado con los corpus compuesto en el capítulo 5, separándolos por tópicos y complejidad textual, además de realizar entrenamientos con documentos de todos los tópicos pero separados por complejidad textual. En el punto 6.2.2 se entrenó de manera preliminar el *Chatbot* usando todo el conjunto de entrenamiento, con el motivo de determinar los hiperparámetros más adecuados; sin embargo, al momento de entrenar el modelo con los documentos separados por tópicos (Comunicación; Ciencia y Tecnología; Historia, Geografía y Economía) y con todos los tópicos, usando los hiperparámetros encontrados, no se obtuvieron resultados satisfactorios. Con ello, se utilizaron nuevos hiperparámetros para obtener mejores respuestas del *Chatbot*, aunque la estructura del *Chatbot* sigue siendo la misma que la mostrada en la Tabla 22. Dichos valores se encuentran en la Tabla 23. Cabe resaltar que el *Chatbot* fue entrenado en un servidor del grupo IA PUCP y que el análisis fue realizado en una laptop personal. Además, para cada tópico del corpus constituido en el Capítulo 5, se eliminaron los signos de puntuación y se recopilaron pares de oraciones en un archivo CSV que utiliza el *Chatbot* para su entrenamiento, similar a lo realizado en el punto 6.2.2. Los *Stop Words* fueron eliminados en los documentos de menos tamaño debido a que, gracias a ello, las respuestas dadas por el *Chatbot* eran más entendibles. En el caso de los documentos de mayor tamaño, quitar los *Stop Words* ocasionaba que las respuestas no sean tan entendibles.

Tabla 23. Hiperparámetros para el experimento por tópicos.

Experimento	**Conjunto de datos (Tópico)**	**Complejidad Textual del conjunto de datos**	**Épocas**	**Optimizador**	**Tasa de aprendizaje**	**Porcentaje del conjunto de validación (%)**	**Tasa de decrecimiento de la tasa de aprendizaje**	**¿Se quitaron los Stopwords?**	**Tamaño del conjunto de entrenamiento**	**Nivel de atención**
A	Comunicación	1	100	SGD	1	20	0.975	No	8.2 MB	6
B	Comunicación	2	100	SGD	1	20	0.98	No	22.7 MB	6
C	Ciencia y Tecnología	1	150	SGD	1	20	0.975	Si	1.1 MB	6
D	iencia y Tecnología	2	150	SGD	1	20	0.98	Si	3.7 MB	6
E	Historia, Geografía y	1	150	SGD	1	20	0.975	Si	1.3 MB	6

	Economía									
F	Historia, Geografía y Economía	2	150	SGD	1	20	0.98	Si	472.8 KB	6
G	Todos	1	150	SGD	1	20	0.975	No	11.7 MB	6
H	Todos	2	150	SGD	1	20	0.975	No	28.7 MB	6

El experimento A se hizo con el *Chatbot* entrenado con los documentos de Comunicación de complejidad textual 1, obteniendo la siguiente gráfica de error para el conjunto de datos de entrenamiento y validación, en la Figura 27.

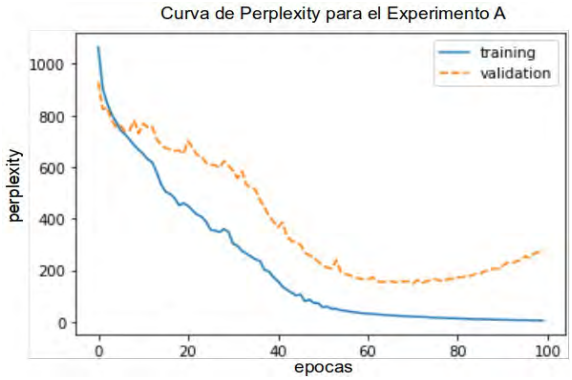

Figura 27. Curva de error para el chatbot en el experimento A.

Para dicho entrenamiento, se obtuvieron los siguientes datos estadísticos, recopilados en la Tabla 24.

Tabla 24. Resumen estadístico para el entrenamiento en el experimento A.

Conjunto de datos	**Media**	**Desviación Estándar**	**Mínimo**	**Máximo**
Entrenamiento	217.391	264.319	5.275	1065.287
Validación	386.721	236.1	148.561	929.83

Al tener en cuenta la Figura 27 y la media, se puede observar que el error, también conocido como *Perplexity*, el cual indica la capacidad de comprensión del *Chatbot*, es menor en el conjunto de entrenamiento, lo que significa que el *Chatbot* es capaz de reconocer oraciones de entrada ya conocidos y, de manera similar, oraciones nuevas, como se observa en la curva de error del conjunto de validación. Sin embargo, se observa que la curva tiende al *Overfitting* debido a que el error en validación empieza a aumentar en las últimas épocas, podría no responder con oraciones que sean entendibles

para entradas nuevas. También, analizando la gráfica, se observa que el *Chatbot* aprende más rápido en el conjunto de entrenamiento que en el de validación.

Una vez entrenado el *Chatbot*, se hizo que generase 20 respuestas a 20 oraciones dadas como entrada, de manera similar al experimento realizado por Clair et al. (2018b), con el objetivo de analizar la capacidad de respuesta del *Chatbot*. Para realizar dicha evaluación, se predijo la complejidad textual de las oraciones que generó el *Chatbot*, agrupado por cantidad de oraciones de manera incremental (Ejemplo: O1, O1-O2, O1-O2-O3,...O1-...O20), usando la herramienta implementada en el capítulo 4, obteniendo el siguiente gráfico de la Figura 28.

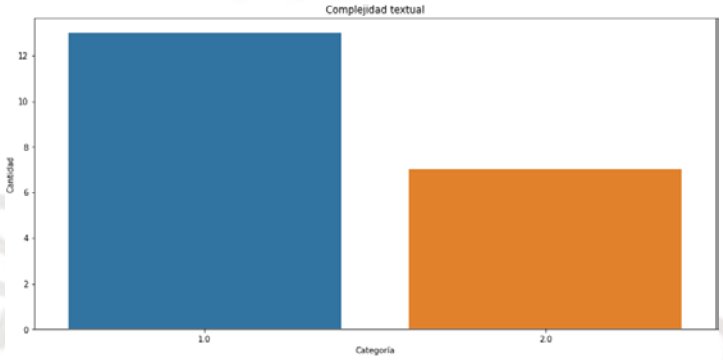

Figura 28. Complejidad textual de las respuestas del chatbot para el Experimento A.

Para evaluar la capacidad de respuesta del *Chatbot* usando la complejidad textual de sus respuestas, se ideó el siguiente experimento estadístico: La complejidad textual que indica la herramienta de análisis de complejidad textual, implementada en el capítulo 4, es 1 o 2, siendo la media de dichos valores 1.5. Entonces, si la media de todas las categorías predichas es menor a 1.5, la complejidad textual de las respuestas del *Chatbot* es 1 (Simple/Primaria), mientras que si es mayor a 1.5 es 2 (Complejo/Secundaria). Si es igual a 1.5, entonces las respuestas son complejas y simples. Además, si se obtiene que la moda es 1, entonces la complejidad textual de las respuestas del *Chatbot* es 1 (Simple/Primaria), mientras que si la moda es 2, la complejidad textual es 2 (Simple/Complejo). Para analizar la media se utilizará una prueba de hipótesis para la media, utilizando el puntaje Z, con un nivel de confianza del 95%.

Para el experimento A, la hipótesis a cola derecha a analizar para la media es la siguiente:

$$\boxed{\begin{array}{l} H_0 \leqslant 1.5(1) \\ H_1 > 1.5(2) \end{array}}$$

Donde la Hipótesis nula requiere que la media de las respuestas dadas por el Chatbot sea menor o igual a 1.5, debido a que la complejidad textual del conjunto de entrenamiento era de 1. Además, la hipótesis alternativa indicaría que las respuestas dadas por el Chatbot tienen una complejidad textual de 2.

Entonces, se obtuvieron los siguientes resultados estadísticos para la gráfica de la Figura 28, ubicados en la Tabla 25.

Tabla 25. Resultados estadísticos de las respuestas generadas en el experimento A.

Z 0.95	Z Score	Moda	Media	Accuracy
1.6448	-1.370	1	1.35	0.65

Siendo el experimento a cola derecha, analizando los puntajes Z, no hay pruebas suficientes para rechazar la hipótesis nula. Siguiendo el experimento estadístico planteado, se observa que la media de 1.35 es menor que 1.5, mientras que la moda es 1, lo que indica que la complejidad textual de las respuestas del *Chatbot* es de 1. Además, como el *Chatbot* fue entrenado con documentos de complejidad textual 1, se concluye, para el experimento A, que la capacidad de respuesta del *Chatbot* ocasiona que sus respuestas sean de complejidad textual 1, lo cual es lo esperado, con un *Accuracy* del 65%.

El experimento B se hizo con el *Chatbot* entrenado con los documentos de Comunicación de complejidad textual 2, obteniendo la siguiente gráfica de error para el conjunto de datos de entrenamiento y validación, en la Figura 29.

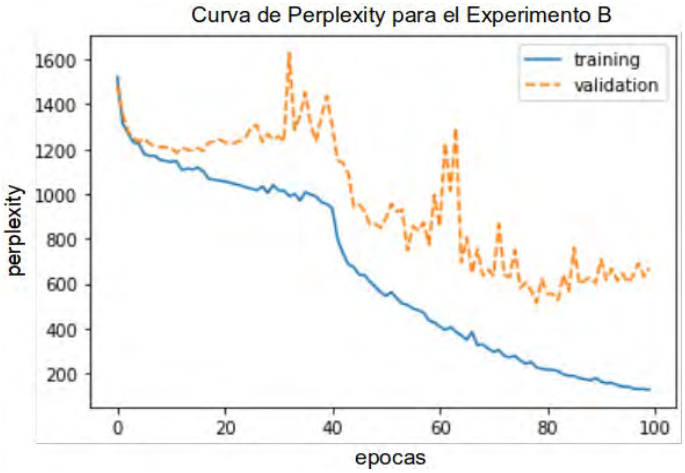

Figura 29. Curva de error para el chatbot en el experimento B.

Para dicho entrenamiento, se obtuvieron los siguientes datos estadísticos, recopilados en la Tabla 26.

Tabla 26. Resumen estadístico para el entrenamiento en el experimento B.

Conjunto de datos	Media	Desviación Estándar	Mínimo	Máximo
Entrenamiento	651.013	397.528	126.090	1520.812
Validación	972.303	294.05	513.885	1629.45

Al tener en cuenta la Figura 29 y la media, se puede observar que el error es menor en el conjunto de entrenamiento, lo que significa que el Chatbot es capaz de reconocer oraciones de entrada ya conocidos y, de menor manera, oraciones nuevas, como se observa en la curva de error del conjunto de validación. Sin embargo, se observa que la curva tiende al *Overfitting* debido a que el error en validación empieza a aumentar en las últimas épocas, podría no responder con oraciones que sean entendibles para entradas nuevas. También, analizando la gráfica, se observa que el *Chatbot* aprende más rápido en el conjunto de entrenamiento que en el de validación.

De mismo modo que en el experimento A, se hizo que el *Chatbot* generase 20 respuestas, para luego agruparlas por oraciones de manera incremental y calcular su complejidad textual. Con ello se obtuvo la Figura 30.

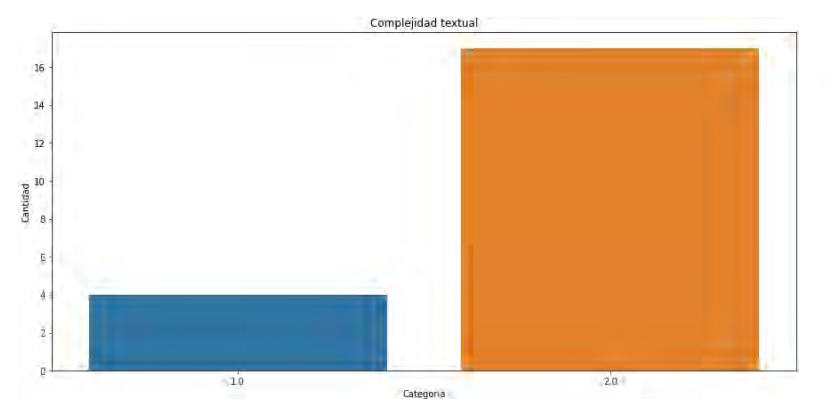

Figura 30. Complejidad textual de las respuestas del chatbot para el Experimento B.

Para el experimento B, la hipótesis a cola izquierda a analizar para la media es la siguiente:

$$\boxed{\begin{array}{l} H_0 \geqslant 1.5(1) \\ H_1 < 1.5(2) \end{array}}$$

Donde la Hipótesis nula requiere que la media de las respuestas dadas por el *Chatbot* sea mayor o igual a 1.5, debido a que la complejidad textual del conjunto de entrenamiento era de 2. Además, la hipótesis alternativa indicaría que las respuestas dadas por el *Chatbot* tienen una complejidad textual de 1. Entonces, se obtuvieron los siguientes resultados estadísticos para la gráfica de la Figura 30, ubicados en la Tabla 27.

Tabla 27. Resultados estadísticos de las respuestas generadas en el experimento B.

Z 0.95	Z Score	Moda	Media	Accuracy
1.6448	5.81	2	1.9	0.9

Siendo el experimento a cola izquierda, analizando los puntajes Z, no hay pruebas suficientes para rechazar la hipótesis nula. Siguiendo el experimento estadístico planteado, se observa que la media de 1.9 es mayor que 1.5, mientras que la moda es 2, lo que indica que la complejidad textual de las respuestas del *Chatbot* es de 2. Además, como el *Chatbot* fue entrenado con documentos de complejidad textual 2, se concluye, para el experimento B, que la capacidad de respuesta del *Chatbot* ocasiona que sus respuestas sean de complejidad textual 2, lo cual es lo esperado, con un Accuracy del 90%.

El experimento C se hizo con el *Chatbot* entrenado con los documentos de Ciencia y Tecnología de complejidad textual 1, obteniendo la siguiente gráfica de error para el conjunto de datos de entrenamiento y validación, en la Figura 31.

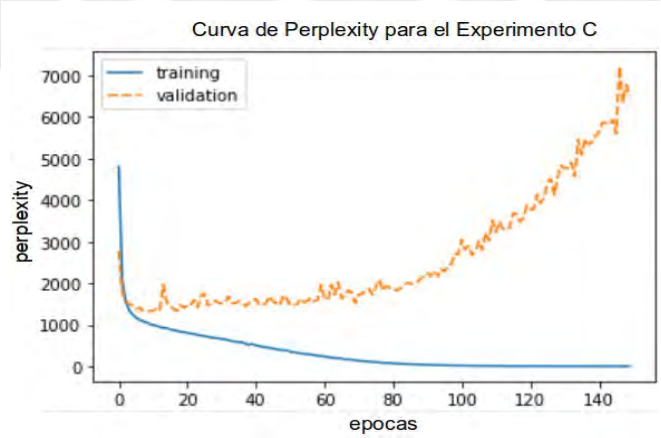

Figura 31. Curva de error para el chatbot en el experimento C.

Para dicho entrenamiento, se obtuvieron los siguientes datos estadísticos, recopilados en la Tabla 28.

Tabla 28. Resumen estadístico para el entrenamiento en el experimento C.

Conjunto de datos	Media	Desviación Estándar	Mínimo	Máximo
Entrenamiento	334.306	536.022	2.181	4803.019
Validación	2614.210	1461.93	1320.80	7208.38

Al tener en cuenta la Figura 31 y la media, se puede observar que el error es menor en el conjunto de entrenamiento, lo que significa que el *Chatbot* es capaz de reconocer oraciones de entrada ya conocidas. Sin embargo, se observa que la curva tiende al *Overfitting* debido a que el error en validación empieza a aumentar, lo que ocasionaría que el *Chatbot* podría no responder con oraciones que sean entendibles para entradas nuevas. También, analizando la gráfica, se observa que el *Chatbot* aprende más rápido en el conjunto de entrenamiento que en el de validación.

De mismo modo que en el experimento B, se hizo que el *Chatbot* generase 20 respuestas, para luego agruparlas por oraciones de manera incremental y calcular su complejidad textual. Con ello se obtuvo la Figura 32.

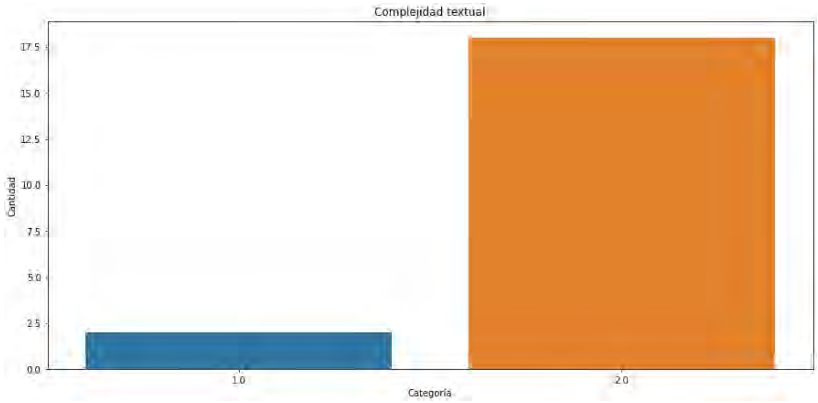

Figura 32. Complejidad textual de las respuestas del chatbot para el Experimento C.

Para el experimento C, la hipótesis a cola derecha a analizar para la media es la siguiente:

$$\boxed{\begin{matrix} H_0 \leqslant 1.5(1) \\ H_1 > 1.5(2) \end{matrix}}$$

Donde la Hipótesis nula requiere que la media de las respuestas dadas por el Chatbot sea menor o igual a 1.5, debido a que la complejidad textual del conjunto de entrenamiento era de 1. Además, la hipótesis alternativa indicaría que las respuestas dadas por el *Chatbot* tienen una complejidad textual de 2. Además, se obtuvieron los

siguientes resultados estadísticos para la gráfica de la Figura 32, ubicados en la Tabla 29.

Tabla 29. Resultados estadísticos de las respuestas generadas en el experimento C.

Z 0.95	Z Score	Moda	Media	Accuracy
1.6448	5.81	2	1.9	0.1

Siendo el experimento a cola derecha, analizando los puntajes Z, hay pruebas suficientes para rechazar la hipótesis nula. Siguiendo el experimento estadístico planteado, se observa que la media de 1.9 es mayor que 1.5, mientras que la moda es 2, lo que indica que la complejidad textual de las respuestas del *Chatbot* es de 2. Además, como el *Chatbot* fue entrenado con documentos de complejidad textual 1, se concluye, para el experimento C, que la capacidad de respuesta del *Chatbot* ocasiona que sus respuestas sean de complejidad textual 2, lo cual no es lo esperado ya que se esperaba que las respuestas dadas por el *Chatbot* fuesen de complejidad textual 1. Sin embargo, para este conjunto de datos, no se obtuvo dicho resultado debido a la poca cantidad de datos de entrenamiento, lo que ocasionó que el *Chatbot* no sea capaz de identificar correctamente los patrones en el texto durante el entrenamiento para generar texto de respuesta, incluso el *Accuracy* es bajo, sólo un 10%.

El experimento D se hizo con el *Chatbot* entrenado con los documentos de Ciencia y Tecnología de complejidad textual 2, obteniendo la siguiente gráfica de error para el conjunto de datos de entrenamiento y validación, en la Figura 33.

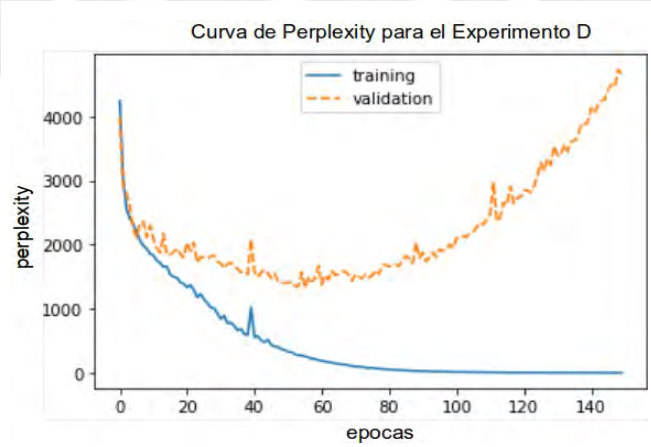

Figura 33. Curva de error para el chatbot en el experimento D.

Para dicho entrenamiento, se obtuvieron los siguientes datos estadísticos, recopilados en la Tabla 30.

Tabla 30. Resumen estadístico para el entrenamiento en el experimento D.

Conjunto de datos	Media	Desviación Estándar	Mínimo	Máximo
Entrenamiento	466.710	736.357	1.525	4242.890
Validación	2244.343	864.41	1326.103	4731.51

Al tener en cuenta la Figura 33 y la media, se puede observar que el error es menor en el conjunto de entrenamiento, lo que significa que el *Chatbot* es capaz de reconocer oraciones de entrada ya conocidos. Sin embargo, se observa que la curva tiende al *Overfitting* debido a que el error en validación empieza a aumentar, lo que ocasionaría que el *Chatbot* podría no responder con oraciones que sean entendibles para entradas nuevas. También, analizando la gráfica, se observa que el *Chatbot* aprende más rápido en el conjunto de entrenamiento que en el de validación.

De mismo modo que en el experimento C, se hizo que el Chatbot generase 20 respuestas, para luego agruparlas por oraciones de manera incremental y calcular su complejidad textual. Con ello se obtuvo la Figura 34.

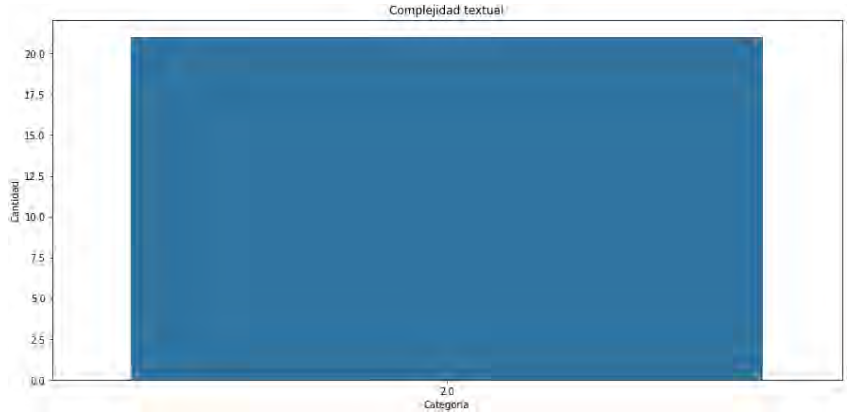

Figura 34. Complejidad textual de las respuestas del chatbot para el Experimento D.

Para el experimento D, la hipótesis a cola izquierda a analizar para la media es la siguiente:

$$\boxed{\begin{array}{l} H_0 \geqslant 1.5(1) \\ H_1 < 1.5(2) \end{array}}$$

Donde la Hipótesis nula requiere que la media de las respuestas dadas por el *Chatbot* sea mayor o igual a 1.5, debido a que la complejidad textual del conjunto de entrenamiento era de 2. Además, la hipótesis alternativa indicaría que las respuestas dadas por el *Chatbot* tienen una complejidad textual de 1. Además, se obtuvieron los

siguientes resultados estadísticos para la gráfica de la Figura 34, ubicados en la Tabla 31.

Tabla 31. Resultados estadísticos de las respuestas generadas en el experimento D.

Z 0.95	Z Score	Moda	Media	Accuracy
1.6448	Inf	2	2	1

Siendo el experimento a cola izquierda, analizando los puntajes Z, no hay pruebas suficientes para rechazar la hipótesis nula. Siguiendo el experimento estadístico planteado, se observa que la media de 2.00 es mayor que 1.5, mientras que la moda es 2, lo que indica que la complejidad textual de las respuestas del *Chatbot* es de 2. Además, como el *Chatbot* fue entrenado con documentos de complejidad textual 2, se concluye, para el experimento D, que la capacidad de respuesta del *Chatbot* ocasiona que sus respuestas sean de complejidad textual 2, lo cual es lo esperado, con un *Accuracy* del 100%.

El experimento E se hizo con el *Chatbot* entrenado con los documentos de Historia, Geografía y Economía de complejidad textual 1, obteniendo la siguiente gráfica de error para el conjunto de datos de entrenamiento y validación, en la Figura 35.

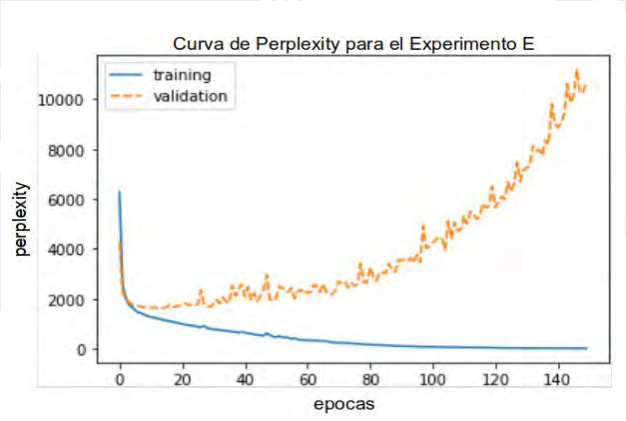

Figura 35. Curva de error para el chatbot en el experimento E.

Para dicho entrenamiento, se obtuvieron los siguientes datos estadísticos, recopilados en la Tabla 32.

Tabla 32. Resumen estadístico para el entrenamiento en el experimento E.

Conjunto de datos	Media	Desviación Estándar	Mínimo	Máximo
Entrenamiento	443.365	680.032	4.903	6279.213

Validación	3917.738	2516.73	1619.706	11203.71

Al tener en cuenta la Figura 35 y la media, se puede observar que el error es menor en el conjunto de entrenamiento, lo que significa que el *Chatbot* es capaz de reconocer oraciones de entrada ya conocidos. Sin embargo, se observa que la curva tiende al *Overfitting* debido a que el error en validación empieza a aumentar, lo que ocasionaría que el *Chatbot* podría no responder con oraciones que sean entendibles para entradas nuevas. También, analizando la gráfica, se observa que el *Chatbot* aprende más rápido en el conjunto de entrenamiento que en el de validación.

De mismo modo que en el experimento D, se hizo que el *Chatbot* generase 20 respuestas, para luego agruparlas por oraciones de manera incremental y calcular su complejidad textual. Con ello se obtuvo la Figura 36.

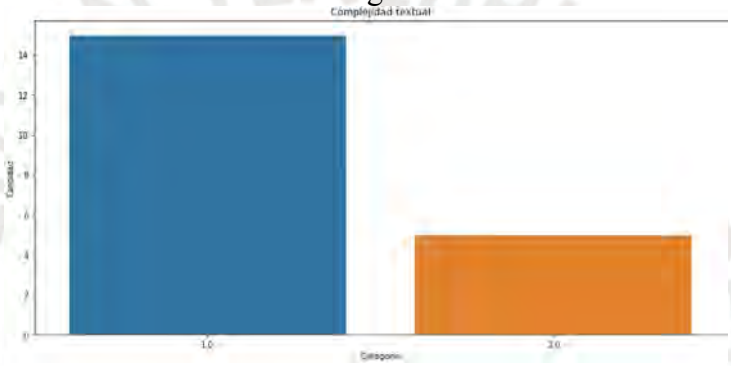

Figura 36. Complejidad textual de las respuestas del chatbot para el Experimento E.

Para el experimento E, la hipótesis a cola derecha a analizar para la media es la siguiente:

$$\boxed{\begin{array}{l} H_0 \leqslant 1.5\,(1) \\ H_1 > 1.5\,(2) \end{array}}$$

Donde la Hipótesis nula requiere que la media de las respuestas dadas por el *Chatbot* sea menor o igual a 1.5, debido a que la complejidad textual del conjunto de entrenamiento era de 1. Además, la hipótesis alternativa indicaría que las respuestas dadas por el *Chatbot* tienen una complejidad textual de 2. Además, se obtuvieron los siguientes resultados estadísticos para la gráfica de la Figura 36, ubicados en la Tabla 33.

Tabla 33. Resultados estadísticos de las respuestas generadas en el experimento E.

Z 0.95	Z Score	Moda	Media	Accuracy
1.6448	-2.516	1	1.25	0.75

Siendo el experimento a cola derecha, analizando los puntajes Z, no hay pruebas suficientes para rechazar la hipótesis nula. Siguiendo el experimento estadístico planteado, se observa que la media de 1.25 es menor que 1.5, mientras que la moda es 1, lo que indica que la complejidad textual de las respuestas del *Chatbot* es de 1. Además, como el *Chatbot* fue entrenado con documentos de complejidad textual 1, se concluye, para el experimento E, que la capacidad de respuesta del *Chatbot* ocasiona que sus respuestas sean de complejidad textual 1, lo cual es es lo esperado, con un *Accuracy* del 75%.

El experimento F se hizo con el *Chatbot* entrenado con los documentos de Historia, Geografía y Economía de complejidad textual 2, obteniendo la siguiente gráfica de error para el conjunto de datos de entrenamiento y validación, en la Figura 37.

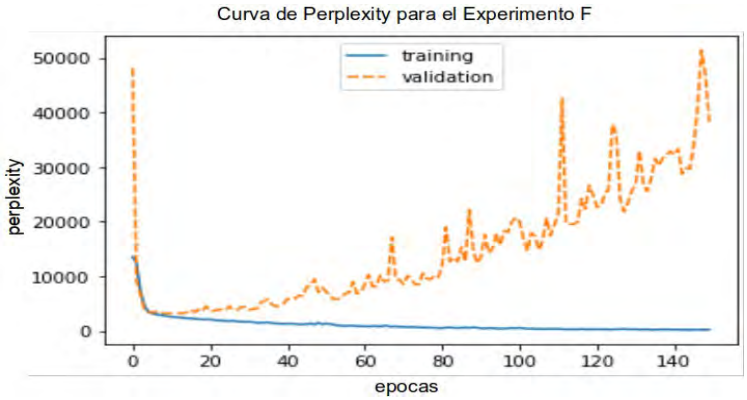

Figura 37. Curva de error para el chatbot en el experimento F.

Para dicho entrenamiento, se obtuvieron los siguientes datos estadísticos, recopilados en la Tabla 34.

Tabla 34. Resumen estadístico para el entrenamiento en el experimento F.

Conjunto de datos	**Media**	**Desviación Estándar**	**Mínimo**	**Máximo**
Entrenamiento	1153.732	1684.745	170.374	13534.537
Validación	14700.003	11076.84	3080.970	51379.78

Al tener en cuenta la Figura 37 y la media, se puede observar que el error es menor en el conjunto de entrenamiento, lo que significa que el *Chatbot* es capaz de reconocer oraciones de entrada ya conocidos. Sin embargo, se observa que la curva tiende al *Overfitting* debido a que el error en validación empieza a aumentar, lo que ocasionaría que el *Chatbot* podría no responder con oraciones que sean entendibles para entradas

nuevas. También, analizando la gráfica, se observa que el *Chatbot* aprende más rápido en el conjunto de entrenamiento que en el de validación.

De mismo modo que en el experimento E, se hizo que el *Chatbot* generase 20 respuestas, para luego agruparlas por oraciones de manera incremental y calcular su complejidad textual. Con ello se obtuvo la Figura 38.

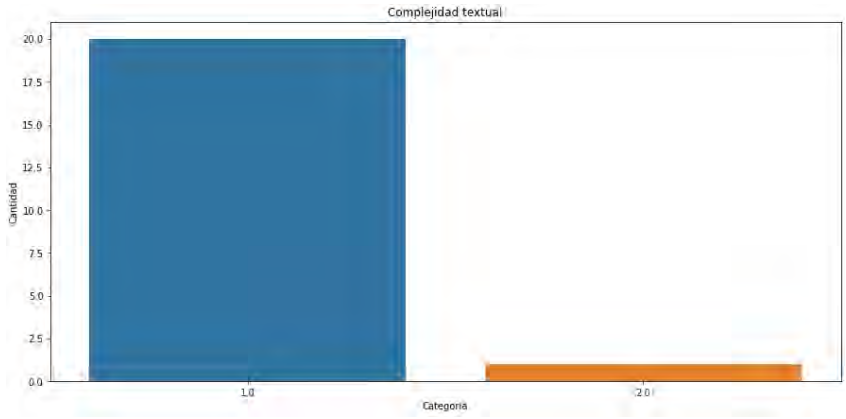

Figura 38. Complejidad textual de las respuestas del chatbot para el Experimento F.

Para el experimento F, la hipótesis a cola izquierda a analizar para la media es la siguiente:

$$\boxed{\begin{array}{l} H_0 \geqslant 1.5(1) \\ H_1 < 1.5(2) \end{array}}$$

Donde la Hipótesis nula requiere que la media de las respuestas dadas por el Chatbot sea mayor o igual a 1.5, debido a que la complejidad textual del conjunto de entrenamiento era de 2. Además, la hipótesis alternativa indicaría que las respuestas dadas por el *Chatbot* tienen una complejidad textual de 1. Además, se obtuvieron los siguientes resultados estadísticos para la gráfica de la Figura 38, ubicados en la Tabla 35.

Tabla 35. Resultados estadísticos de las respuestas generadas en el experimento F.

Z 0.95	Z Score	Moda	Media	Accuracy
1.6448	-8.999	1	1.05	0.05

Siendo el experimento a cola izquierda, analizando los puntajes Z, hay pruebas suficientes para rechazar la hipótesis nula. Siguiendo el experimento estadístico planteado, se observa que la media de 1.05 es menor que 1.5, mientras que la moda es 1, lo que indica que la complejidad textual de las respuestas del *Chatbot* es de 1. Además,

como el *Chatbot* fue entrenado con documentos de complejidad textual 2, se concluye, para el experimento F, que la capacidad de respuesta del *Chatbot* ocasiona que sus respuestas sean de complejidad textual 1, lo cual no es lo esperado ya que se esperaba que las respuestas dadas por el Chatbot fuesen de complejidad textual 2. Sin embargo, para este conjunto de datos, no se obtuvo dicho resultado debido a la poca cantidad de datos de entrenamiento, lo que ocasionó que el *Chatbot* no sea capaz de identificar los patrones en el texto durante el entrenamiento para generar texto de respuesta, incluso el *Accuracy* es bajo, sólo un 5%.

El experimento G se hizo con el *Chatbot* entrenado con los documentos de todos los tópicos de complejidad textual 1, obteniendo la siguiente gráfica de error para el conjunto de datos de entrenamiento y validación, en la Figura 39.

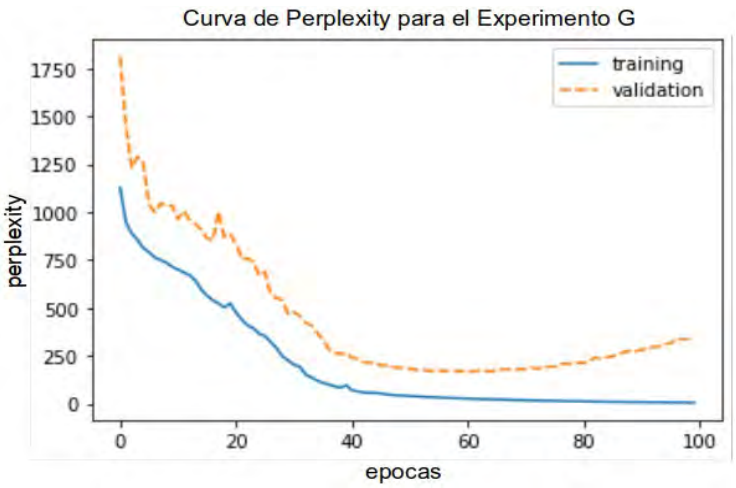

Figura 39. Curva de error para el chatbot en el experimento G.

Para dicho entrenamiento, se obtuvieron los siguientes datos estadísticos, recopilados en la Tabla 36.

Tabla 36. Resumen estadístico para el entrenamiento en el experimento G.

Conjunto de datos	**Media**	**Desviación Estándar**	**Mínimo**	**Máximo**
Entrenamiento	205.325	285.275	5.754	1126.645
Validación	446.688	360.46	165.836	1809.85

Al tener en cuenta la Figura 39 y la media, se puede observar que el error es menor en el conjunto de entrenamiento, lo que significa que el *Chatbot* es capaz de reconocer oraciones de entrada ya conocidos. Sin embargo, se observa que la curva tiende al *Overfitting*, en las últimas épocas, debido a que el error en validación empieza a

aumentar, lo que ocasionaría que el *Chatbot* podría no responder con oraciones que sean entendibles para entradas nuevas. También, analizando la gráfica, se observa que el *Chatbot* aprende más rápido en el conjunto de entrenamiento que en el de validación.

De mismo modo que en el experimento F, se hizo que el *Chatbot* generase 20 respuestas, para luego agruparlas por oraciones de manera incremental y calcular su complejidad textual. Con ello se obtuvo la Figura 40.

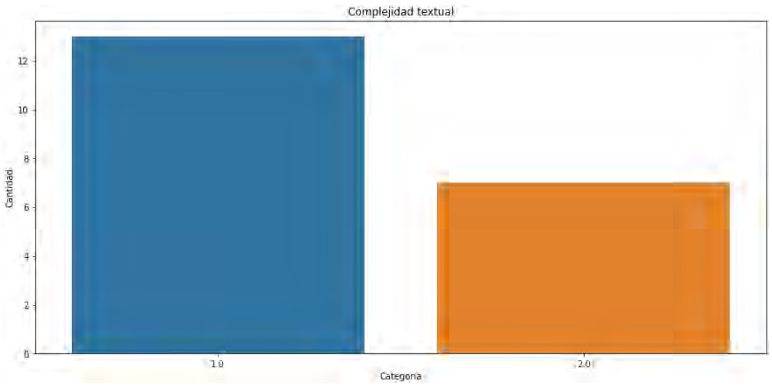

Figura 40. Complejidad textual de las respuestas del chatbot para el Experimento G.

Para el experimento G, la hipótesis a cola derecha a analizar para la media es la siguiente:

$$\boxed{\begin{array}{l} H_0 \leqslant 1.5(1) \\ H_1 > 1.5(2) \end{array}}$$

Donde la Hipótesis nula requiere que la media de las respuestas dadas por el *Chatbot* sea menor o igual a 1.5, debido a que la complejidad textual del conjunto de entrenamiento era de 1. Además, la hipótesis alternativa indicaría que las respuestas dadas por el *Chatbot* tienen una complejidad textual de 2. Además, se obtuvieron los siguientes resultados estadísticos para la gráfica de la Figura 40, ubicados en la Tabla 37.

Tabla 37. Resultados estadísticos de las respuestas generadas en el experimento G.

Z 0.95	Z Score	Moda	Media	Accuracy
1.6448	-1.370	1	1.35	0.65

Siendo el experimento a cola derecha, analizando los puntajes Z, no hay pruebas suficientes para rechazar la hipótesis nula. Siguiendo el experimento estadístico planteado, se observa que la media de 1.35 es menor que 1.5, mientras que la moda es 1,

lo que indica que la complejidad textual de las respuestas del *Chatbot* es de 1. Además, como el Chatbot fue entrenado con documentos de complejidad textual 1, se concluye, para el experimento G, que la capacidad de respuesta del *Chatbot* ocasiona que sus respuestas sean de complejidad textual 1, lo cual es es lo esperado, con un *Accuracy* del 65%.

El experimento H se hizo con el *Chatbot* entrenado con los documentos de todos los tópicos de complejidad textual 2, obteniendo la siguiente gráfica de error para el conjunto de datos de entrenamiento y validación, en la Figura 41.

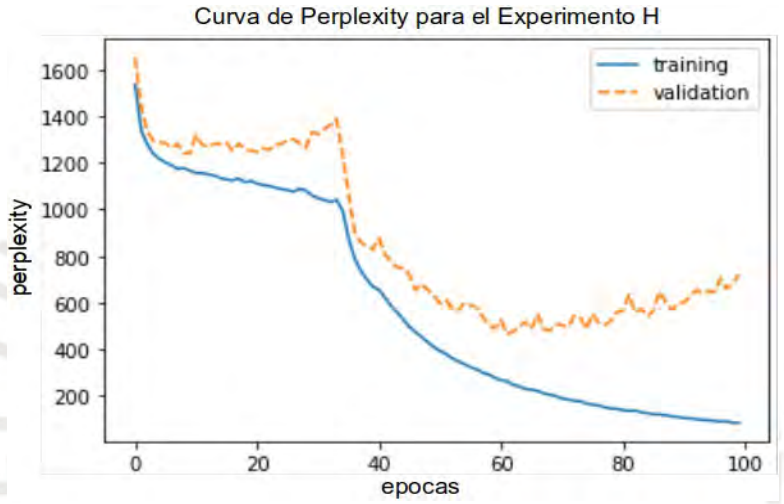

Figura 41. Curva de error para el chatbot en el experimento H.

Para dicho entrenamiento, se obtuvieron los siguientes datos estadísticos, recopilados en la Tabla 38.

Tabla 38. Resumen estadístico para el entrenamiento en el experimento H.

Conjunto de datos	**Media**	**Desviación Estándar**	**Mínimo**	**Máximo**
Entrenamiento	583.815	445.281	84.015	1536.096
Validación	856.294	345.41	465.447	1652.43

Al tener en cuenta la Figura 41 y la media, se puede observar que el error es menor en el conjunto de entrenamiento, lo que significa que el *Chatbot* es capaz de reconocer oraciones de entrada ya conocidos y, de menor manera, oraciones nuevas, como se observa en la curva de error del conjunto de validación. Sin embargo, se observa que la curva tiende al *Overfitting* debido a que el error en validación empieza a aumentar en las últimas épocas, podría no responder con oraciones que sean entendibles para

entradas nuevas. También, analizando la gráfica, se observa que el *Chatbot* aprende más rápido en el conjunto de entrenamiento que en el de validación.

De mismo modo que en el experimento G, se hizo que el *Chatbot* generase 20 respuestas, para luego agruparlas por oraciones de manera incremental y calcular su complejidad textual. Con ello se obtuvo la Figura 42.

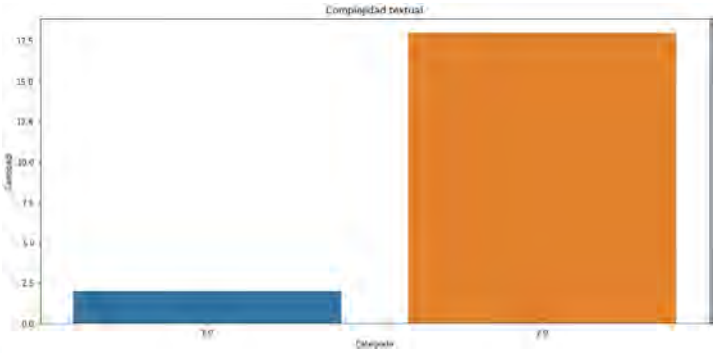

Figura 42. Complejidad textual de las respuestas del chatbot para el Experimento H.

Para el experimento H, la hipótesis a cola izquierda a analizar para la media es la siguiente:

$$\boxed{\begin{array}{l} H_0 \geqslant 1.5(1) \\ H_1 < 1.5(2) \end{array}}$$

Donde la Hipótesis nula requiere que la media de las respuestas dadas por el *Chatbot* sea mayor o igual a 1.5, debido a que la complejidad textual del conjunto de entrenamiento era de 2. Además, la hipótesis alternativa indicaría que las respuestas dadas por el *Chatbot* tienen una complejidad textual de 1. Entonces, se obtuvieron los siguientes resultados estadísticos para la gráfica de la Figura 42, ubicados en la Tabla 39.

Tabla 39. Resultados estadísticos de las respuestas generadas en el experimento H.

Z 0.95	Z Score	Moda	Media	Accuracy
1.6448	5.81	2	1.9	0.9

Siendo el experimento a cola izquierda, analizando los puntajes Z, no hay pruebas suficientes para rechazar la hipótesis nula. Siguiendo el experimento estadístico planteado, se observa que la media de 1.9 es mayor a 1.5, mientras que la moda es 2, lo que indica que la complejidad textual de las respuestas del *Chatbot* es de 2. Además, como el *Chatbot* fue entrenado con documentos de complejidad textual 2, se concluye, para el experimento H, que la capacidad de respuesta del *Chatbot* ocasiona que sus

respuestas sean de complejidad textual 2, lo cual es lo esperado, con un *Accuracy* del 90%.

Gracias a los experimentos realizados con los conjuntos de datos de tamaño mayor a 1.1 MB, se observa que el *Chatbot* aprende a identificar los patrones, conocidos y nuevos, en los textos, siguiendo las curvas de error en el entrenamiento y validación, de manera más eficiente en los experimentos realizados con documentos de complejidad textual 1 que de complejidad textual 2, ya que las curvas de validación y entrenamiento están menos separadas en el primer caso que en el segundo, lo que indicaría que la complejidad textual de los datos de entrada sí afecta el entrenamiento de un *Chatbot*.

El archivo de Jupyter Notebook con el reporte realizado con los experimentos estadísticos se encuentra en el el siguiente link: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Entrenamiento%20de%20Chatbot%20por%20T%C3%B3picos.ipynb. El documento de validación, llamado "Validación de las pruebas de validación para el Chatbot.docx" se puede encontrar también en el siguiente link: https://docs.google.com/document/d/1hf1rWKlMyYTAVAIQlT9WvIzg4quZeA5Hcdi8RpIsLFw/edit?usp=sharing.

6.3 Discusión

Como se vio en el punto 6.2.1, se buscaron distintos repositorios y tutoriales con *Chatbots* ya implementados para usarlos en los experimentos del Objetivo Específico 3. De los 6 repositorios y 3 tutoriales encontrados, se seleccionó un *Chatbot* que utilizaba el modelo de red neuronal *Sequence to Sequence* (Sanders, 2018/2020). Este modelo fue entrenado de manera preliminar en el punto 6.2.2 para encontrar el optimizador y los hiperparámetros que generasen los mejores resultados al momento de realizar el entrenamiento del *Chatbot* con el corpus constituido en el capítulo 5. Estos hiperpárametros encontrados inicialmente sirvieron para realizar experimentos de prueba para el punto 6.2.3; sin embargo, los resultados de dichos entrenamientos de prueba del Chatbot usando el corpus separado por tópicos, demostraron que sería necesario encontrar mejores hiperparámetros, tarea que se realizó, concluyendo con nuevos hiperparámetros relacionados a la tasa de aprendizaje del *Chatbot*, así como la tarea de remover *Stop Words* en los los tópicos que contaban con menor cantidad de datos. Luego de hallados dichos hiperparámetros nuevos, se realizaron 8 entrenamientos

y experimentos estadísticos, 2 para los documentos del tópico de Comunicación, 2 para los de Ciencia y Tecnología, 2 para los de Historia, Geografía y Economía y 2 para todos los documentos. La razón de tener 2 experimentos por cada tópico es que se realizó 1 experimento para cada complejidad textual. Concluidos dichos experimentos, se obtuvieron como resultados qué la complejidad textual de las respuestas dadas por el *Chatbot*, analizados de manera similar al experimento realizado por Clair et al. (2018b), entrenado corresponden a la complejidad textual de los documentos con la que fue entrenado; sin embargo, esto se dio para los entrenamientos que contaban con una cantidad de datos mayores a 1.1 MB de datos, 6 de 8 experimentos, lo que ocasiona que el *Chatbot* no generase respuestas con nivel de complejidad textual correspondiente para 2 de los 8 experimentos, aquellos con pocos datos de entrenamiento. Esto se debe a que, debido a la cantidad de datos, el *Chatbot* no fue capaz de descubrir los patrones correctos en los textos para generar respuestas adecuadas. La complejidad textual de las respuestas fue hallada usando la herramienta de cálculo de complejidad textual desarrollado en el capítulo 4.

De los gráficos de curvas de errores del punto 6.2.3 se deduce que el *Chatbot* entrenado es capaz de reconocer bien los patrones de texto ya conocido, pero es menos capaz de responder bien al recibir texto nuevo, de manera similar al experimento realizado por Cuáyahuitl et al. (2019b). Además, en las gráficas halladas se observa que algunas respuestas no corresponden al nivel de complejidad textual esperado, lo que puede generar confusión a la persona que utilizase el *Chatbot*, ocasionando que deba replantear sus preguntas para obtener una mejor respuesta, tal como lo plantea (Beaver, 2018).

Cabe resaltar que los experimentos realizados con el *Chatbot*, si bien fueron hechos con los textos de materiales educativos encontrados para corpus del capítulo 5, podrían ser realizados con textos de otras áreas, incluso, podrían realizarse en otros idiomas, provisto que dichos documentos en nuevos idiomas estén clasificados por su complejidad textual. En el caso de ser nuevos documentos en español, su complejidad textual podría ser hallado con la herramienta implementada en el capítulo 4.

Por último, el *Chatbot* entrenado no es capaz de identificar la complejidad textual de las preguntas del usuario, como sí sucede en el experimento de Mohammadi & Khasteh (2018b), ya que el objetivo específico de este capítulo era evaluar el impacto de los

corpus de entrenamiento en las respuestas del *Chatbot*, mas no el impacto de la complejidad textual de las preguntas del usuario, además que el *Chatbot* seleccionado en el presente capítulo no disponía de dichas capacidades.

Capítulo 7. Conclusiones y trabajos futuros

En este capitulo se presentan las conclusiones de los resultados alcanzados en el presente documento, de manera tal que se realiza una conclusión por cada capítulo relacionado a cada Objetivo Específico, para luego terminar con una conclusión general que sintetiza todo lo expuesto. Luego de ello, se procede con la sección de trabajos futuros.

7.1 Conclusiones

En el Capítulo 4, el cual trata el Objetivo Específico 1 (Implementar el módulo para calcular métricas de complejidad textual), se recopilaron 48 métricas de análisis de complejidad textual basadas en Coh-Metrix para ser implementadas. Dichas métricas fueron desarrolladas en una librería/módulo de Python con la cual se puede analizar textos en español y calcular dichas métricas. Además, luego de realizar un experimento de aprendizaje supervisado con los textos recopilados y procesados, se entrenó el mejor modelo de clasificación, K Nearest Neighbors para este experimento, el cual obtuvo los siguientes resultados al ser entrenado con todos los documentos: Accuracy = 0.918919; Precision = 0.937500; Recall = 0.882353 y F1 Score = 0.909091. Al ser añadido a la librería implementada dicho modelo de clasificación, otorga la capacidad de predecir la categoría (Complejidad textual) de textos nuevos. Dicha funcionalidad convierte a la librería en un Framework basado en Coh-Metrix para la identificación de complejidad textual para textos en español, el cual puede ser personalizado ya que también permite que nuevos textos puedan ser clasificados usando modelos de aprendizaje de máquina distintos al que es usado por defecto.

Con lo expuesto, se concluye que se logró completar el Objetivo Específico 1, el cuál era implementar el módulo para calcular métricas de complejidad textual. Para ello, se completaron los tres resultados esperados planteados para dicho Objetivo Específico, ya que se recopiló un conjunto de métricas de complejidad textual seleccionadas para analizar textos en español, se desarrolló la herramienta de implementación propia que calcula las métricas de complejidad textual para textos en español y se desarrolló un módulo de clasificación de complejidad textual parametrizado al corpus de textos en español constituido en el capítulo 5.

En el Capítulo 5, el cual trata el Objetivo Específico 2 (Constituir un corpus de más de cien textos en español clasificados por su complejidad textual), se recopiló un conjunto de 183 documentos de textos en español, los cuales fueron constituidos en un Corpus almacenado en una base de datos SQLite que se usó para realizar el experimento de aprendizaje supervisado. Dicho corpus, en primer lugar, fue analizado mediante un experimento de *Clustering*. luego de se procesado por la librería implementada en el Capítulo 5, para verificar que, en efecto, puede ser dividido en dos categorías (1=Primaria, 2=Secundaria). Además, dichos datos procesados para realizar el experimento de *Clustering* fueron almacenados también en la base de datos SQLite para poder ser utilizados en el entrenamiento del modelo de clasificación en el Capítulo 4.

Además, se concluye que se logró completar el Objetivo Específico 2, el cuál era constituir un corpus de más de cien textos en español clasificados por su complejidad textual. Para ello, se completaron los tres resultados esperados planteados para dicho Objetivo Específico, ya que se recopiló un conjunto de textos en español seleccionados por su complejidad textual, se validó el agrupamiento de dichos textos, para finalmente procesar dichos textos encontrados con la herramienta implementada en el capítulo 4.

En el capítulo 6, el cual trata el Objetivo Específico 3 (Evaluar el impacto de los corpus generados en las respuestas de un *Chatbot*), se encontró un *Chatbot* para utilizar en la experimentación de dicho capítulo, luego de investigar en distintos repositorios y tutoriales. Dicho *Chatbot* fue luego entrenado de manera preliminar para encontrar los posibles mejores hiperparámetros y optimizador, de los cuales sólo los hiperparámetros de relacionados a la tasa de aprendizaje fueron modificados en los experimentos finales. De dichos experimentos finales, los cuales fueron 8, 2 de ellos tuvieron resultados no óptimos debido a la poca cantidad de datos de entrenamiento para aquellos 2 experimentos; sin embargo, con los experimentos restantes, se logró concluir que la complejidad textual de los datos de entrenamiento sí influyen en el entrenamiento y nivel de respuesta de los *Chatbots*.

Por último, se concluye que se logró completar el Objetivo Específico 3, el cuál era evaluar el impacto de los corpus generados en las respuestas de un Chatbot. Para ello, se completaron los los resultados esperados planteados para dicho Objetivo Específico, ya que se seleccionó un *Chatbot* implementado para realizar los experimentos, luego se entrenó dicho *Chatbot* de manera preliminar con el corpus de textos en español del

capítulo 5, para luego concluir con las pruebas de validación realizadas con los entrenamientos hechos por tópicos separados y con todos los textos.

Con todo lo expuesto, se concluye que se cumple con el Objetivo Principal planteado al inicio de la Tesis, ya que se logra identificar el nivel de complejidad de documentos referentes a un tópico utilizando la herramienta desarrollado en el capítulo 4, el cuál funciona como *Framework* debido a como fue implementado uno, con ello, resolviendo el problema central descrito al inicio: ¿Cómo identificar el nivel de complejidad textual de documentos referentes a un tópico para generar corpus de entrenamiento para un *Chatbot*?

7.2 Trabajos futuros

Para obtener mejores resultados durante la experimentación con el *Chatbot*, se propone lo siguiente:

- Utilizar un modelo de *Chatbot* para español pre entrenado para obtener mejores respuestas.
- Utilizar un conjunto de datos en español específicamente diseñado para tareas de clasificación complejidad textual.

Referencias

Aguirre, B. V., Uresti, J. A. R., & Du Boulay, B. (2016). An analysis of student model portability. International Journal of Artificial Intelligence in Education, 26(3), 932-974.

Aryadoust, V., & Goh, C. C. M. (2014). CaMLA Working Papers Predicting Listening Item Difficulty with Language Complexity Measures: A Comparative Data Mining Study.

Balakrishna, S. V. (2015). Analyzing Text Complexity and Text Simplification: Connecting Linguistics, Processing and Educational Applications.

Beaver, I. R. (2018). Automatic Conversation Review for Intelligent Virtual Assistants .

Bengoetxea, K., Gonzalez-Dios, I., & Aguirregoitia, A. (2020). AzterTest: Open source linguistic and stylistic analysis tool. Procesamiento de Lenguaje Natural, 64, 61-68. https://doi.org/10.26342/2020-64-7

Bich, S. (2017). « Is There Choice in Non-Native Voice?» Linguistic Feature Engineering and a Variationist Perspective in Automatic Native Language Identification.

Branco, A., Rodrigues, J., Costa, F., Silva, J., & Vaz, R. (2014). Assessing automatic text classification for interactive language learning. International Conference on Information Society (i-Society 2014), 70-78.

Choi, K. (2018a). Computational lyricology: Quantitative approaches to understanding song lyrics and their interpretations. University of Illinois at Urbana-Champaign .

Choi, K. (2018b). Computational lyricology: Quantitative approaches to understanding song lyrics and their interpretations. University of Illinois at Urbana-Champaign.

Clair, J. S., Conley, T., & Kalita, J. (2018a). Impact of Auxiliary Loss Functions on Dialogue Generation Using Mutual Information.

Clair, J. S., Conley, T., & Kalita, J. (2018b). Impact of Auxiliary Loss Functions on Dialogue Generation Using Mutual Information. https://github.com/pender/chatbot-rnn

Crossley, S. A., Skalicky, S., & Dascalu, M. (2019). Moving beyond classic readability formulas: New methods and new models. Journal of Research in Reading, 42(3-4), 541-561.

Cuáyahuitl, H., Lee, D., Ryu, S., Choi, S., Hwang, I., & Kim, J. (2019a). Deep Reinforcement Learning for Chatbots Using Clustered Actions and Human-Likeness Rewards. 2019 International Joint Conference on Neural Networks (IJCNN), 1-8.

Cuáyahuitl, H., Lee, D., Ryu, S., Choi, S., Hwang, I., & Kim, J. (2019b). Deep Reinforcement Learning for Chatbots Using Clustered Actions and Human-Likeness Rewards. 2019 International Joint Conference on Neural Networks (IJCNN), 1-8.

Dascalu, M., McNamara, D. S., Trausan-Matu, S., & Allen, L. K. (2018). Cohesion network analysis of CSCL participation. Behavior Research Methods, 50(2), 604-619.

Davoodi, E., Kosseim, L., & Mongrain, M. (2017). A context-aware approach for the identification of complex words in natural language texts. 2017 IEEE 11th International Conference on Semantic Computing (ICSC), 97-100.

Fitzgerald, J., Elmore, J., Koons, H., Hiebert, E. H., Bowen, K., Sanford-Moore, E. E., & Stenner, A. J. (2015). Important text characteristics for early-grades text complexity. Journal of Educational Psychology, 107(1), 4.

Fitzgerald, J., Elmore, J., Relyea, J. E., Hiebert, E. H., & Stenner, A. J. (2016). Has first-grade core reading program text complexity changed across six decades? Reading Research Quarterly, 51(1), 7-28.

Git. (s. f.). Git. Recuperado 9 de junio de 2020, de https://git-scm.com/

Google Inc. (s. f.). Te damos la bienvenida a Colaboratory—Colaboratory. Recuperado 3 de julio de 2020, de https://colab.research.google.com/notebooks/welcome.ipynb?hl=es

Graesser, A. C., McNamara, D. S., Louwerse, M. M., & Cai, Z. (2004). Coh-Metrix: Analysis of text on cohesion and language. Behavior Research Methods, Instruments, and Computers, 36(2), 193-202. https://doi.org/10.3758/BF03195564

Guryanov, I. O., Yarmakeev, I. E., Kiselnikov, A. S., & Harkova, E. V. (2017). Text Complexity: Periods Of Study In Russian Linguistics. Revista Publicando.

Halliday, M. A. K. (1985). An introduction to functional grammar.

Hartmann, N., Cucatto, L., Brants, D., & Aluísio, S. (2016). Automatic classification of the complexity of nonfiction texts in portuguese for early school years. International Conference on Computational Processing of the Portuguese Language, 12-24.

Hiebert, E. H. (s. f.). What is Text Classification? En Common Core State Standards (pp. 56-63).

Johnstone, B. (2018). Discourse Analysis (3rd ed.).

Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Third Edition draft.

Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report, Ver. 2.3 EBSE Technical Report. EBSE.

Kleijn, S. (2018). Clozing in on readability: How linguistic features affect and predict text comprehension and on-line processing. LOT .

Latifi, S. M. F. (2016). Development and validation of an automated essay scoring framework by integrating deep features of English language .

Lecorvé, G., Ayats, H., Fournier, B., Mekki, J., Chevelu, J., Battistelli, D., Béchet, N., Towards, A., & Fournier, B. (2019). Towards the Automatic Processing of Language Registers: Semi-supervisedly Built Corpus and Classifier for French.

Li, X., & Liu, J. (2016). Automatic Essay Scoring Based on Coh-Metrix Feature Selection for Chinese English Learners. International Symposium on Emerging Technologies for Education, 382-393.

López-Anguita, R., Montejo-Ráez, A., & Díaz-Galiano, M. C. (2018). Complexity Measures and POS N-grams for Author Identification in Several Languages.

McNamara, D. S., Graesser, A. C., McCarthy, P. M., & Cai, Z. (2014). Automated evaluation of text and discourse with Coh-Metrix. Cambridge University Press.

Minhas, S., & Hussain, A. (2016). From spin to swindle: Identifying falsification in financial text. Cognitive computation, 8(4), 729-745.

Minhas, S. Z. (2016). A corpus driven computational intelligence framework for deception detection in financial text .

Mohammadi, H., & Khasteh, S. H. (2018a). A machine learning approach to persian text readability assessment using a crowdsourced dataset. arXiv preprint arXiv:1810.06639.

Mohammadi, H., & Khasteh, S. H. (2018b). A machine learning approach to persian text readability assessment using a crowdsourced dataset. arXiv preprint arXiv:1810.06639.

Ojha, P. K., Ismail, A., & Kuppusamy, K. S. (2018). Perusal of readability with focus on web content understandability. Journal of King Saud University-Computer and Information Sciences.

Patout, P.-A., & Cordy, M. (2019). Towards context-aware automated writing evaluation systems. Proceedings of the 1st ACM SIGSOFT International Workshop on Education through Advanced Software Engineering and Artificial Intelligence , 17-20.

Perfetti, C. A., Landi, N., & Oakhill, J. (2005). The Acquisition of Reading Comprehension Skill. En The science of reading: A handbook (pp. 227-247).

Project Jupyter. (s. f.). Project Jupyter | Home. Recuperado 9 de junio de 2020, de https://jupyter.org/

Python. (s. f.). Welcome to Python.org. Recuperado 9 de junio de 2020, de https://www.python.org/

Quispesaravia, A., Perez, W., Cabezudo, M. S., & Alva-Manchego, F. (2016a). Coh-Metrix-Esp: A complexity analysis tool for documents written in Spanish. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREĆ16) , 4694-4698.

Quispesaravia, A., Perez, W., Cabezudo, M. S., & Alva-Manchego, F. (2016b). Coh-Metrix-Esp: A complexity analysis tool for documents written in Spanish. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREĆ16), 4694-4698.

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., & Edwards, D. D. (1995). Artificial Intelligence A Modern Approach.

Sanders, A. (2020). AbrahamSanders/seq2seq-chatbot [Python]. https://github.com/AbrahamSanders/seq2seq-chatbot (Original work published 2018)

scikit-learn. (s. f.). Scikit-learn. Recuperado 9 de junio de 2020, de https://scikit-learn.org/stable/about.html#citing-scikit-learn

Solovyev, V., Ivanov, V., & Solnyshkina, M. (2018). Assessment of reading difficulty levels in Russian academic texts: Approaches and metrics. Journal of Intelligent & Fuzzy Systems, 34(5), 3049-3058.

Spacy. (s. f.). SpaCy · Industrial-strength Natural Language Processing in Python. Recuperado 6 de julio de 2020, de https://spacy.io/

SQLite. (s. f.). SQLite Home Page. Recuperado 9 de junio de 2020, de https://www.sqlite.org/index.html

Stevanoski, B., & Gievska, S. (2019). Team Ned Leeds at SemEval-2019 Task 4: Exploring Language Indicators of Hyperpartisan Reporting. Proceedings of the 13th International Workshop on Semantic Evaluation , 1026-1031.

Sung, Y.-T., Chen, J.-L., Cha, J.-H., Tseng, H.-C., Chang, T.-H., & Chang, K.-E. (2015). Constructing and validating readability models: The method of integrating

multilevel linguistic features with machine learning. Behavior research methods, 47(2), 340-354.

Trotzek, M., Koitka, S., & Friedrich, C. M. (2017). Linguistic Metadata Augmented Classifiers at the CLEF 2017 Task for Early Detection of Depression. CLEF (Working Notes) .

Vajjala, S. (2018). Automated assessment of non-native learner essays: Investigating the role of linguistic features. International Journal of Artificial Intelligence in Education, 28(1), 79-105.

Vajjala, S., & Lučić, I. (2018). OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications , 297-304.

Wagner Filho, J. A., Wilkens, R., Zilio, L., Idiart, M., & Villavicencio, A. (2016). Crawling by readability level. International Conference on Computational Processing of the Portuguese Language, 306-318.

Yaneva, V. (2016). Assessing text and web accessibility for people with autism spectrum disorder .

Yang, Y., & Joachims, T. (2008). Text categorization. Scholarpedia, 3(5), 4242. https://doi.org/10.4249/scholarpedia.4242

YAO–TING, S. U. N. G., WEI–CHUN, L. I. N., Dyson, S. B., CHANG, K. N., & YU–CHIA, C. H. E. N. (2015). Leveling L2 texts through readability: Combining multilevel linguistic features with the CEFR. The Modern Language Journal, 99(2), 371-391.

Anexos

En esta sección se presentan los anexos referenciados a través de todo el documento de la tesis.

Anexo A :Plan de Proyecto

- Justificación

  Para justificar el desarrollo de esta investigación, se analizarán cuatro niveles:

  - Conveniencia

    Con la presente investigación se busca mejorar las capacidades de respuesta de los *Chatbots*, haciendo que estos generen respuestas que sean del nivel de conversación del usuario. Estos *Chatbots*, los cuales están basados en reglas de decisión o en corpus, estos últimos implementados mediante árboles de decisiones o modelos de aprendizaje de máquina (Jurafsky & Martin, 2019); los cuales son entrenados sin tener en cuenta que tan complejas puedan ser sus respuestas (Beaver, 2018). Para ello se clasificarán automáticamente por su complejidad textual corpus de entrenamiento de textos en español.

  - Implicaciones prácticas

    Dado el caso de tener una plataforma educativa escolar que pueda ser utilizada por profesores, existiría un módulo que los maestros puedan usar en la cual ellos pueden subir documentos escolares de distintos grados. Con ello, la plataforma debería ser capaz de devolver feedback el cual le indique al maestro si el documento es apropiado para el grado correspondiente o no. Para este caso, el modelo de clasificación implementado en la presente investigación podría ser usado para resolver dicho problema.

  - Implicaciones tecnológicas

    La herramienta para calcular las métricas de complejidad textual para el idioma español utilizará tecnología más moderna que la implementada en estudios anteriores (Quispesaravia et al., 2016a). Además, dicha herramienta a implementar no dependerá de herramientas externas que calculan métricas

de complejidad textual, sino que dicho cálculo será implementado desde cero.

También, el modelo de aprendizaje automático que clasifica textos por su complejidad textual será implementado usando tecnología que es usada actualmente en el área de aprendizaje de máquina, en este caso Scikit Learn, en vez de usar Weka como en la investigación conducida por (Bengoetxea et al., 2020; Quispesaravia et al., 2016a).

Además, de darse el caso de obtener resultados positivos en la presente investigación, se tendrían pruebas que demuestran que los *chatbots* generan mejores conversaciones cuando son entrenados con textos apropiados para el público con el que deben interactuar.

- Área de desarrollo

  De manera similar a las implicaciones tecnológicas, de darse el caso de obtener resultados positivos en la presente investigación, se tendrán resultados que aporten al área de Computación Afectiva mediante la aplicación de los conocimientos del área de Procesamiento de Lenguaje Natural. Esto se debe a que se podría verificar que un *chatbot* entrenado con el texto correcto puede generar mejores respuestas para el usuario y con ello ocasionar que la persona se sienta más augusta con los resultados de la conversación con el programa.

- Viabilidad

La viabilidad del proyecto se justifica en tres niveles: La infraestructura, el acceso a la información y el conocimiento. Estos tres niveles se detallarán a continuación:

Desde el punto de vista de infraestructura, los experimentos iniciales se realizarán en la propia laptop del investigador, la cual cuenta con una CPU Intel i7, 16 GB de memoria RAM y una tarjeta gráfica NVIDIA MX150 de 4 GB. También, se usará uso de computación en la nube a través de Google Colab para los experimentos iniciales cuando se requiera mayor poder computacional con respecto a la GPU. De darse el caso que las especificaciones disponibles en el

Google Colab sean insuficientes, se procederá a usar el servidor del grupo de Inteligencia Artificial de la Especialidad de Informática de la PUCP. Este servidor cuenta con las siguientes especificaciones: 1 Deep Learning Workstation con 4 GPUs RTX 2080 Ti. Una Desktop para TensorFlow, Keras, y PyTorch.

Por otro lado, desde el punto de vista de acceso a la información, se posee acceso a los textos a utilizar para el experimento a través de la página del Ministerio de Educación conocida como www.perueduca.com. Ahí se encuentran los documentos necesarios a descargar en formato PDF y estos archivos pueden ser obtenidos mediante la técnica de *Web Scraping*.

Por último, desde el punto de vista de los conocimientos requeridos, el investigador tiene conocimiento sobre los temas de Aprendizaje de máquina, Análisis de Datos, Aprendizaje Profundo y Procesamiento de Lenguaje Natural, los cuales fueron aprendidos en los cursos de la carrera de Informática. Además, con esa base, el investigador tiene la facilidad de entender la información disponible en la web de las técnicas referidas al tema.

- Alcance

  Este proyecto de fin de carrera busca entrenar un *chatbot*, el cual será capaz de dar respuestas complejas o simples, dependiendo del nivel de complejidad textual que posea el usuario.

  Para ello, se creará una librería en Python que calculará las métricas de complejidad textual de textos, con la cual se entrenará un modelo de aprendizaje de máquina capaz de clasificar textos usando dichas métricas calculadas por la librería.

  Los textos a usar serán recopilados mediante de la página www.perueduca.pe, siendo usados a manera de estudio, con los cuales se creará el corpus para entrenar al modelo de clasificación y al *chatbot*. Para este experimento, se usarán los temas como “Comunicación”, “Historia, Geografía y Economía” y “Ciencia y Tecnología”, los cuales serán usados como dominios del *chatbot*.

El alcance del proyecto no incluye el desarrollo de un Sistema de Información que contenga el modelo de clasificación implementado ni la implementación del *chatbot*, debido a que se usará uno ya disponible.

- Restricciones
    - La información a utilizar serán textos escolares peruanos que se encuentran disponibles en formato digital en la página www.perueduca.pe y se usarán los textos de nivel secundaria y primaria. Además, dichos textos estarán en español y se usarán los temas como "Comunicación", "Historia, Geografía y Economía" y "Ciencia y Tecnología", los cuales serán usados como dominios del *chatbot*..
    - Se utilizarán modelos de *chatbot* ya disponibles a fin de probar el efecto del modelo de clasificación de textos en el entrenamiento del *chatbot*.
    - El dominio del chatbot será cada uno de los temas indicados para los documentos extraídos.
    - Se utilizarán modelos tradicionales de aprendizaje de máquina para la clasificación de los textos obtenidos debido a que la base de datos disponible es limitado en cantidad de textos.
- Identificación de los riesgos del proyecto

Para la presente investigación, se definió una escala de severidad del 1 al 5, en la cual 1 es muy bajo, 2 es bajo, 3 es medio, 4 es grave y 5 es muy grave. Con ello en mente, se encontraron los siguientes riesgos para el presente proyecto:

- Disponibilidad limitada de *chatbots*.
    - Descripción: Se refiere a el caso de que los *chatbots* disponibles no tengan la características adecuada para el desarrollo del experimento, lo que ocasionaría que se tenga que implementar un *chatbot* propio y con ello se retrasaría la investigación.
    - Síntomas:

- Cantidad limitada de *chatbots* encontrados en GitHub y GitLab.
- De los *chatbots* encontrados, ninguno o muy pocos de ellos tienen las características adecuadas para realizar el experimento.

- Probabilidad: 10% de probabilidad de que ocurra.
- Impacto: El impacto vendría a ser muy grave, el cual es el número 5.
- Severidad: Conociendo la probabilidad e impacto, la severidad es de 0.5.
- Mitigación: Se buscarían más *chatbots* en otros repositorios además de GitHub y GitLab.
- Contingencia: En caso de concretarse este riesgo, se procedería a implementar un *chatbot* propio.

- Cantidad de datos insuficiente

  - Descripción: Se refiere al caso en el que la cantidad de documentos encontrados en www.perueduca.pe no sea la suficiente como para realizar el entrenamiento del *chatbot* en distintos temas.
  - Síntomas:
    - De los documentos encontrados, solo un tópico tiene documentos en los grados seleccionados.
    - Otros tópicos no tienen documentos en todos los grados seleccionados.
  - Impacto: El impacto vendría a ser leve, el cual es el número 1.
  - Probabilidad: 50% de probabilidad que ocurra.
  - Severidad: Conociendo el impacto y la severidad, la severidad es de 0.5.
  - Mitigación: Buscar mas documentos de otros sitios web.
  - Contingencia: Realizar el entrenamiento del *chatbot* con el tema que posee más documentos.

- Estructura de descomposición del trabajo (EDT)

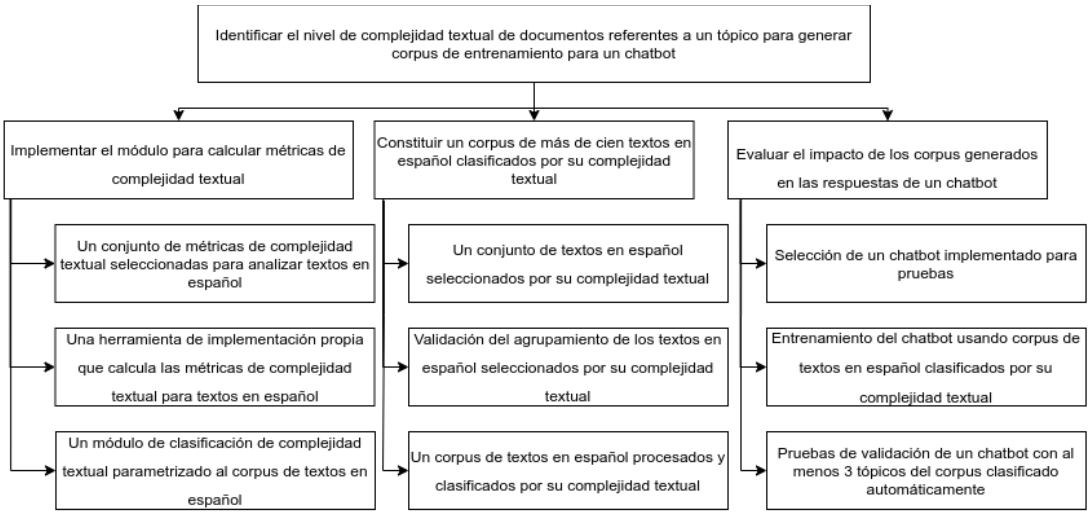

Figura 43. Estructura de descomposición del trabajo

- Descripción de componentes:

A continuación, se describirán los componentes del EDT, describiendo cada paquete, entregable relacionado y criterio de aceptación.

Componente	Descripción
Código del paquete de trabajo	R1.1
Descripción del paquete de trabajo	Un conjunto de métricas de complejidad textual seleccionadas para analizar textos en español
Entregable(s)	Reporte de estudio de selección de métricas de complejidad textual
Criterios de aceptación del entregable(s)	Aceptación del asesor

Componente	Descripción
Código del paquete de trabajo	R1.2
Descripción del paquete de trabajo	Una herramienta de implementación propia que calcula las métricas de complejidad textual para textos en español
Entregable(s)	• Código de la librería implementada en GitHub. • Documentos y manuales de la librería
Criterios de aceptación del entregable(s)	• Implementación aceptada • Aceptación del asesor

Componente	Descripción
Código del paquete de trabajo	R1.3
Descripción del paquete de trabajo	Un módulo de clasificación de complejidad textual parametrizado al corpus de textos en español
Entregable(s)	• Código del modelo implementado en GitHub.
Criterios de aceptación del entregable(s)	• Aceptación del asesor • Métricas de Accuracy, Precision, Recall y F1 score mayores al 75%.

Componente	Descripción
Código del paquete de trabajo	R2.1
Descripción del paquete de trabajo	Un conjunto de textos en español seleccionados por su complejidad textual
Entregable(s)	Archivos de la base de datos en formato SQLite
Criterios de aceptación del entregable(s)	• Base de datos con más de 100 registros

Componente	Descripción
Código del paquete de trabajo	R2.2
Descripción del paquete de trabajo	Validación del agrupamiento de los textos en español seleccionados por su complejidad textual
Entregable(s)	Reporte de clustering
Criterios de aceptación del entregable(s)	• Gráficos de clustering validados

Componente	Descripción
Código del paquete de trabajo	R2.3
Descripción del paquete de trabajo	Un corpus de textos en español procesados y clasificados por su complejidad textual
Entregable(s)	Archivos de la base de datos en formato SQLite
Criterios de aceptación del entregable(s)	• Base de datos con más de 100 registros

Componente	Descripción
Código del paquete de trabajo	R3.1
Descripción del paquete de trabajo	Selección de un chatbot implementado para pruebas
Entregable(s)	Reporte de selección de chatbot con código fuente de GitHub o GitLab
Criterios de aceptación del entregable(s)	• Chatbot seleccionado validado por el asesor

Componente	Descripción
Código del paquete de trabajo	R3.2
Descripción del paquete de trabajo	Entrenamiento del chatbot usando corpus de textos en español clasificados por su complejidad textual
Entregable(s)	Reporte de entrenamiento del chatbot seleccionado
Criterios de aceptación del entregable(s)	• Reporte validado por el asesor

Componente	Descripción
Código del paquete de trabajo	R3.3
Descripción del paquete de trabajo	Pruebas de validación de un chatbot conversacional con al menos 3 tópicos del corpus clasificado automáticamente
Entregable(s)	Reportes con gráficos estadísticos de las pruebas de validación
Criterios de aceptación del entregable(s)	• Reportes validados por el asesor

- Lista de tareas

  Para las tareas a realizar se mostrará una valorización no monetaria.

  Las reuniones con el asesor se darán 1 día por semana como mínimo, considerando un salario de 10000 soles al cual le dedica un 2.5% de su tiempo mensual para realizar las asesorías.

  En cada día se trabajarán 4 horas aproximadamente y se considerará un pago de 980 soles al mes, considerado de 30 días. Este monto se basa en el pago que se realiza a un practicante en una empresa.

  Estas actividades están más detalladas y organizadas en el cronograma del proyecto.

Nombre	Duración	Esfuerzo	Costo
Reunión con el asesor.	17 día	1 día-persona	850 soles
Un conjunto de métricas de complejidad textual seleccionadas para analizar textos en español	1 días	1 días-persona	32 soles
Una herramienta de implementación propia que calcula las métricas de complejidad textual para textos en español	16 días	60 días-persona	512 soles
Un módulo de clasificación de complejidad textual parametrizado al corpus de textos en español	4 días	4 días-persona	128 soles
Un conjunto de textos en español seleccionados por su complejidad textual	4 días	4 días-persona	128 soles
Validación del agrupamiento de los textos en español seleccionados por su complejidad textual	4 días	4 días-persona	128 soles
Un corpus de textos en español procesados y clasificados por su complejidad textual	4 días	4 días-persona	128 soles
Selección de un chatbot implementado para prueba.	4 días	4 días-persona	128 soles
Entrenamiento del chatbot usando corpus de textos en español clasificados por su complejidad textual	4 días	4 días-persona	128 soles
Pruebas de validación de un chatbot conversacional con al menos 3 tópicos del corpus clasificado automáticamente.	12 días	14 días-persona	96 soles

Tabla 40. Lista de tareas.

- Cronograma del proyecto

  El cronograma se encuentra en el Anexo D.

- Lista de recursos

  - Personas involucradas y necesidades de capacitación

    Las personas involucradas en este proyecto son el autor del presente documento y el asesor. El primero se encargará de realizar la investigación e implementación de las herramientas a desarrollar mientras que el segundo se encargará de guiar y asesorar con la metodología.

    En cuanto a la capacitación, el asesor es uno de los investigadores lideres del grupo de Inteligencia Artificial de la PUCP donde se desarrollan proyectos relacionados al Procesamiento de Lenguaje Natural. Por otro lado, el alumno

posee los conocimientos teóricos para poder entender los nuevos conceptos que involucran el trabajo de tesis. Estos fundamentos teóricos fueron adquiridos en los cursos de Aprendizaje de Máquina, Análisis de Datos, Aplicaciones de Ciencias de Computación, Temas avanzados en Computación, Lenguajes de Programación 1 y Algoritmia, aprendidas en la carrera de Ingeniería Informática de la PUCP.

- Materiales requeridos para el proyectos

  No se requerirán materiales físicos para la realización de esta investigación.

- Estándares utilizados en el proyecto

  Para el la implementación del código en Python, se utilizará el la guía de estilo para código en Python PEP-8 (Python, s. f.).

- Equipamiento requeridos

  - Laptop: Se requiere una laptop para realizar la investigación, implementar la herramienta que calcula las métricas de complejidad de textos, el clasificador de textos por su nivel de complejidad textual y los distintos reportes. Servirá como ambiente de desarrollo. Las características de esta laptop fueron descritas en la sección de viabilidad del plan de proyecto.

  - Servidor: En caso de necesitar mayor poder de cómputo que el provisto por la laptop personal, se usará un un servidor del grupo de Inteligencia Artificial de la especialidad de Informática de la PUCP, para realizar el entrenamiento de los modelos. Las características de este servidor fueron descritas en la sección de viabilidad del plan de proyecto.

- Herramientas requeridas

Herramienta	Importancia
Python	Lenguaje con el que se desarrollaran las herramientas.
Jupyter Notebook	Elaboración de reportes gráficos y entrenamiento de modelos de aprendizaje automático.
Google Colab	Entrenamiento de modelos de aprendizaje automático.
Coh-Metrix	Posee la teoría y funcionamiento de las métricas de complejidad textual a usar.
Scikit Learn	Librería de Python que implementa distintos algoritmos de aprendizaje automático.
Git	Realiza control de versionado de código.
SQLite	Implementa bases de datos rápidas, simples y fáciles de usar.
Spacy	Realiza procesamiento de lenguaje natural.

Tabla 41. Herramientas a usar.

- Costeo del Proyecto

  A continuación se presenta el costeo del proyecto por semana. En este costeo no monetario se consideró el mismo sueldo para el estudiante y el asesor que en la sección de lista de tareas.

Ítem	Descripción			Unidad	Cantidad	Valor Unidad (S/.)	Monto Parcial (S/.)	Monto Total (S/.)
0	Costo total del proyecto			---	---	---	---	645
1.	Estudiantes o tesistas			---	---	---	---	245
1.1	Estudiante 1			Horas	20	12,25	245	
2.	Otros participantes (en caso aplique)			---	---	---	---	250
2.1	Asesor			Horas	5	50	250	
3.	Servicios y consultoría (en caso aplique)s			---	---	---	---	---
4.	Materiales e insumos (en caso aplique)s			---	---	---	---	---
5.	Bienes y equipos	Unid1	Cant1 -	Unid2	Cant 2-	-	-	150
5.1	Laptop	Equipo	1	Horas	20	5	100	
5.2	Servidores	Equipo	1	Horas	5	10	50	
6.	Pasajes y viáticos	Unid1	Cant1 -	Unid2	Cant 2-	-	-	---

Tabla 42.Costeo del proyecto.

Anexo B :Estudios primarios y formularios de extracción.

Los estudios primarios y formularios de extracción se encuentran en la hoja de cálculo, creada con Google Sheets/Excel anexa a este documento llamado “Estudios primarios y formularios de extracción.xlsx”. Esta hoja de cálculo tiene las siguientes hojas.

- Estudios primarios P1: Documentos de la pregunta 1.
- Estudios primarios P2. Documentos de la pregunta 2.
- Estudios primarios P3: Documentos de la pregunta 3.
- Formulario de extracción P1: Formulario de extracción de la pregunta 1.
- Formulario de extracción P2: Formulario de extracción de la pregunta 2

- Formulario de extracción P3: Formulario de extracción de la pregunta 3.

La hoja de cálculo también puede ser encontrada con el siguiente link: https://docs.google.com/spreadsheets/d/1eYYqzNcOBDoZ86gQ04w2XWd6vXqiCuGmHh6ZUEXHw8g/edit?usp=sharing.

Anexo C :Metodología de Trabajo.

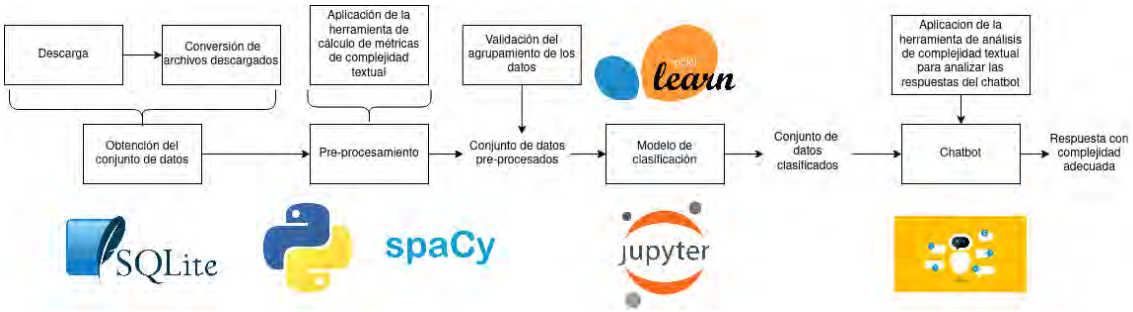

Figura 44. Metodología de trabajo

En el siguiente gráfico se presenta la metodología de trabajo a usar para la realización de los experimentos.

Anexo D :Cronograma del proyecto

Para el cronograma, se creó un un documento de Excel/Google Sheets llamado "Cronograma.xlsx". Este cronograma también puede ser accedido mediante el siguiente link: https://docs.google.com/spreadsheets/d/1BXsKhBBAwRJyf77UkqVobnhcmuEchihB-AFMRink-iI/edit?usp=sharing.

Anexo E :Métricas de complejidad textual a implementar

Para saber qué métricas de complejidad textual basadas en coh-metrix se implementarán, se creó un reporte en Jupyter Notebook (.ipynb) en el cual están detallados las 48 métricas a usar. Dicho reporte se puede encontrar en https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/reports/M%C3%A9tricas.ipynb.

Su aprobación como parte del identificador objetivamente verificable se puede encontrar en el archivo de Word/Google Docs "Validación del reporte de métricas de complejidad textual.docx", el cual puede ser también encontrado en el siguiente link:

https://docs.google.com/document/d/1PjElv9ob4fvYRmFD3PnYWimPQXLkO6fj5cblB_HvNgI/edit?usp=sharing.

Anexo F :Repositorio con la implementación de la herramienta de cálculo de complejidad textual.

La implementación de la herramienta se encuentra en el repositorio de Github: https://github.com/Hans03430/TextComplexityAnalyzerCM.

Dicha herramienta en desarrollo posee documentación, la cual puede ser encontrada en https://hans03430.github.io/TextComplexityAnalyzerCM/.

La validación de la documentación se puede observar en el archivo llamado "Validación de la documentación de la herramienta que calcula las métricas de complejidad textual para textos en español.docx", el cual puede ser encontrado también en el siguiente link: https://docs.google.com/document/d/1zGQrZ9Y7ENjVu6yHFe09oOKt5Q9wfs9QIQdv7Q957FQ/edit?usp=sharing.

El conjunto de 48 algoritmos de cálculo de métricas de complejidad textual puede ser verificado mediante el archivo README.md, así como el código fuente de cada Índice además de la clase Text Complexity Analyzer, las cuales se encuentran en los siguientes links presentados a continuación de manera detallada:

- Text Complexity Analyzer: Clase que invoca a todas las métricas implementadas para analizar textos en español. Además, es capaz de identificar la categoría (Complejidad textual) de un texto a manera de cadena de caracteres.
    - Implementación:
        - https://github.com/Hans03430/TextComplexityAnalyzerCM/blob/master/text_complexity_analyzer_cm/text_complexity_analyzer.py.
    - Documentación:
        - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.text_complexity_analyzer.html.
- Índices Descriptivos:

    - Implementación:
        - https://github.com/Hans03430/TextComplexityAnalyzerCM/blob/master/text_complexity_analyzer_cm/coh_metrix_indices/descriptive_indices.py.
    - Documentación:
        - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.coh_metrix_indices.descriptive_indices.html.
- Índices de legibilidad
    - Implementación:
        - https://github.com/Hans03430/TextComplexityAnalyzerCM/blob/master/text_complexity_analyzer_cm/coh_metrix_indices/readability_indices.py.
    - Documentación:
        - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.coh_metrix_indices.readability_indices.html.
- Índices de diversidad léxica:
    - Implementación:
        - https://github.com/Hans03430/TextComplexityAnalyzerCM/blob/master/text_complexity_analyzer_cm/coh_metrix_indices/lexical_diversity_indices.py.
    - Documentación:
        - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.coh_metrix_indices.lexical_diversity_indices.html.

- Índices de cohesión referencial:
  - Implementación:
    - https://github.com/Hans03430/TextComplexityAnalyzerCM/blob/master/text_complexity_analyzer_cm/coh_metrix_indices/referential_cohesion_indices.py.
  - Documentación:
    - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.coh_metrix_indices.referential_cohesion_indices.html.
- Índices de complejidad sintáctica:
  - Implementación:
    - https://github.com/Hans03430/TextComplexityAnalyzerCM/blob/master/text_complexity_analyzer_cm/coh_metrix_indices/syntactic_complexity_indices.py.
  - Documentación:
    - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.coh_metrix_indices.syntactic_complexity_indices.html.
- Índices de densidad de patrones sintácticos:
  - Implementación:
    - https://github.com/Hans03430/TextComplexityAnalyzerCM/blob/master/text_complexity_analyzer_cm/coh_metrix_indices/syntactic_pattern_density_indices.py.
  - Documentación:

        - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.coh_metrix_indices.syntactic_complexity_indices.html.

- Índices de información de palabras:
    - Implementación:
        - https://github.com/Hans03430/TextComplexityAnalyzerCM/blob/master/text_complexity_analyzer_cm/coh_metrix_indices/word_information_indices.py.
    - Documentación:
        - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.coh_metrix_indices.word_information_indices.html.
- Índices de conectivos:
    - Implementación:
        - https://github.com/Hans03430/TextComplexityAnalyzerCM/blob/master/text_complexity_analyzer_cm/coh_metrix_indices/connective_indices.py.
    - Documentación:
        - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.coh_metrix_indices.connective_indices.html.

Ahora, se indican los links para otros módulos usados en la herramienta:

- Pipes: Estas clases fueron usadas para extender el funcionamiento del Pipeline de Spacy a la hora de implementar las el cálculo de métricas de complejidad textual para español.
    - Implementación:

        - https://github.com/Hans03430/TextComplexityAnalyzerCM/tree/master/text_complexity_analyzer_cm/pipes.
    - Documentación:
        - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.pipes.html.
- Utils: Funciones y clases extra usadas que sirven de apoyo para la herramienta implementada.
    - Implementación:
        - https://github.com/Hans03430/TextComplexityAnalyzerCM/tree/master/text_complexity_analyzer_cm/utils.
    - Documentación:
        - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.utils.html.
- Model: Módulo que contiene los algoritmos de Machine Learning entrenados para la clasificación.
    - Implementación:
        - https://github.com/Hans03430/TextComplexityAnalyzerCM/tree/master/text_complexity_analyzer_cm/model.
    - Documentación:
        - https://hans03430.github.io/TextComplexityAnalyzerCM/text_complexity_analyzer_cm.model.html.

Además, se implementó una pequeña aplicación web para usar dicha herramienta a manera de prueba, cuyo código fuente puede ser encontrado en:

- Front-end: https://github.com/Hans03430/tesis_frontend
- Back-end: https://github.com/Hans03430/tesis_backend

Anexo G :Base de datos con los textos en español.

Para almacenar los documentos PDF descargados y convertidos a archivos TXT, se usó una base de datos SQLite en donde fueron almacenados. En total, se recopilaron 187 textos, los cuales son almacenados en la tabla principal llamada "OBTAINED_TEXT". La base de datos SQLite se encuentra en el siguiente link: https://github.com/Hans03430/Tesis_Chatbot/blob/master/data/processed/db.sqlite. Esta base de datos SQLite puede ser visualizada con el programa "DB Browser for SQLite", disponible para Windows, Mac y Linux.

Además, el diccionario de datos de la base de datos se llama "Data dictionary.xlsx" y se encuentra en los siguientes links:

- https://github.com/Hans03430/Tesis_Chatbot/blob/master/docs/Database/Data%20dictionary.xlsx. Para descargar el diccionario de datos, hacer clic en el botón "Download."
- https://docs.google.com/spreadsheets/d/1fc3BdaUSwKKtTil6VIF1JrQPn_OWIS5N5MPo6ri-vf0/edit?usp=sharing.

En dicho diccionario de datos se encuentra la información detallada de lo que contiene cada tabla de la base de datos SQLite. La validación de la base de datos está en "Validación de la base de datos.docx", el cual también se encuentra en el siguiente link: https://docs.google.com/document/d/1O9BGyiGqHfIA7XMEQtryeV7VKPOPb5RjDkAtgfAbLs4/edit?usp=sharing.

Anexo H :Reporte de Clustering

En el presente anexo se encuentra lo relacionado al reporte de clustering realizado para presentar el resultado 2 del Objetivo Específico 2, en el punto 5.2.2 del Capítulo 5. El experimento se encuentra en el siguiente archivo de Jupyter Notebook, ubicado en el siguiente link: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Clustering.ipynb.

El archivo cuenta con 2 partes principales:

- En la primera parte se realiza el experimento de Clustering con todos los documentos, identificado por el subtítulo “Clustering con todos los datos”.
- En la segunda parte se realiza el experimento de Clustering con todos los documentos, pero separados por dominio, identificados por el subtítulo “Clustering por categoría”. Cada experimento con dominios específicos se identifican por los subtítulos:
    - Comunicación.
    - Ciencia y Tecnología
    - Historia, Geografía y Economía

Por último la validación de dicho reporte se encuentra en el archivo llamado “Validación del reporte de Clustering.docx”. El cual también se encuentra en el siguiente link: https://docs.google.com/document/d/1FGEXnF0SxK4hPPwBW0mHP1UmQ2ZSHP_752PQJDAJ_Gc/edit?usp=sharing.

Anexo I :Módulo de clasificación

En el presente anexo se introducen los reportes donde se realizaron los experimentos de aprendizaje supervisado para el punto 4.2.3. A continuación, se detallan en qué archivos se encuentran los experimentos realizados.

- Clasificación con todos los documentos ajustado en F1 Score: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Classifier.ipynb.
- Clasificación con todos los documentos ajustado en Accuracy: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Classifier-Accuracy.ipynb.
- Clasificación con los documentos de Comunicación ajustado en F1 Score: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Classifier-Comunicacion.ipynb.

- Clasificación con los documentos de Comunicación ajustado en Accuracy: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Classifier-Comunicacion-Accuracy.ipynb.

- Clasificación con los documentos de Historia, Geografía y Economía ajustado en F1 Score: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Classifier-Historia%2C%20Geograf%C3%Ada%20y%20Econom%C3%ADa.ipynb.

- Clasificación con los documentos de Historia, Geografía y Economía ajustado en Accuracy: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Classifier-Historia%2C%20Geograf%C3%Ada%20y%20Econom%C3%Ada-Accuracy.ipynb.

- Clasificación con los documentos de Ciencia y Tecnología ajustado en F1 Score: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Classifier-CTA.ipynb.

- Clasificación con los documentos de Ciencia y Tecnología ajustado en Accuracy: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Classifier-CTA-Accuracy.ipynb

El código fuente del mejor modelo de aprendizaje de máquina de clasificación se encuentra en el siguiente link: https://github.com/Hans03430/Tesis_Chatbot/blob/master/notebook/modeling/Best%20Classifier.ipynb

Por último, la aprobación del experto para el modelo de aprendizaje está en el documento "Validación del experimento de clasificación.docx", el cual también se encuentra en el siguiente link: https://docs.google.com/document/d/19xKGCCTPCxj1XAO627Cvev6zhi_BrZIZ3QLhshAzMYg/edit?usp=sharing.