

TECNOLÓGICO NACIONAL DE MEXICO

Instituto Tecnológico de Culiacán



TITULACIÓN INTEGRAL

OPCION:

TESIS PROFESIONAL

NOMBRE DEL PROYECTO:

Corpus en español para el reconocimiento de personalidad

ALUMNO(S)

Gerardo Angel Beltrán Ruiz

CARRERA :

Ingeniería en sistemas computacionales

Num(s). De Control.

18170281

Culiacán, Sinaloa (2024)

AUTORIZACIÓN DE IMPRESIÓN



Instituto Tecnológico de Culiacán

DIVISION DE ESTUDIOS PROFESIONALES
ASUNTO: Autorización de Impresión

Culiacán, Sinaloa, 18/10/2024 8:33

C. GERARDO ANGEL BELTRAN RUIZ
PASANTE(S) DE LA CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES
PRESENTE.

De acuerdo con fallo emitido por la Comisión Revisora Integrada por:

Cédula Profesional

Presidente:	Doctorado en Ciencias de la Computación	
	María Lucía Barrón Estrada	8630269
Secretario:	Maestría en Administración de Negocios, Area Finanzas	
	Víctor Manuel Bátiz Beltrán	6715966
Vocal:	Maestría en Ciencias en Ciencias Computacionales	
	Arcelia Judith Bustillos Martínez	13118410

Y considerando que reúne los requisitos establecidos en la Opción: **TITULACIÓN INTEGRAL: TESIS PROFESIONAL "CORPUS EN ESPAÑOL PARA EL RECONOCIMIENTO DE PERSONALIDAD"**, damos a usted nuestra autorización para que proceda a imprimir su trabajo profesional.

Sin otro particular, quedo de usted.

ATENTAMENTE
Excelencia en Educación Tecnológica®

CONCEPCION MENDOZA ROSALES
Jefa de la División de Estudios Profesionales

 **EDUCACIÓN** 
INSTITUTO TECNOLÓGICO DE CULIACÁN
DEPARTAMENTO DE DIVISIÓN DE
ESTUDIOS PROFESIONALES



Juan de Dios Bátiz 310 Pte. Col. Guadalupe C.P. 80220 Culiacán, Sinaloa Tel. 667-454-0100

<https://tecnm.mx>

<https://www.culiacan.tecnm.mx>

Dedicatoria

Dedico este trabajo especialmente a mi madre Teresa Elizabeth Ruiz T. por el apoyo de todos los días, la confianza que tiene en mí y la motivación en estudiar para poder ser una persona preparada y tener una carrera profesional.

A toda mi familia y a mis amigos que he hecho a lo largo de todos mis estudios por motivarme a siempre conseguir lo que me he propuesto.

Agradecimientos

Quiero agradecer formalmente al Instituto Tecnológico de Culiacán, al personal de la División de Estudios de Posgrado e Investigación y a todos mis profesores que compartieron su conocimiento y apoyaron en mi formación profesional.

Agradezco a la Dra. María Lucía Barrón Estrada por la atención, la guía y la retroalimentación para poder realizar correctamente mi trabajo de tesis.

De la misma manera, agradecer al M.C. Víctor Manuel Bátiz Beltrán por la oportunidad de realizar las residencias profesionales con un proyecto que me aportó mucho conocimiento y por siempre resolver mis dudas.

A todos mis amigos y compañeros que siempre estuvieron acompañándome y fueron un gran apoyo y motivación.

Declaración de autenticidad

Por la presente declaro que, salvo cuando se haga referencia específica al trabajo de otras personas, el contenido de esta tesis es original y no se ha presentado total o parcialmente para su consideración para cualquier otro título o grado en esta o cualquier otra Universidad. Esta tesis es resultado de mi propio trabajo y no incluye nada que sea resultado de algún trabajo realizado en colaboración, salvo que se indique específicamente en el texto.

Gerardo Angel Beltrán Ruiz.

Culiacán, Sinaloa, México, 2024

Resumen

Un rasgo importante de las personas es su personalidad, la cual se distingue principalmente por el comportamiento, carácter, intelecto o el físico. Existen diversos modelos de personalidad, los cuales usan diferentes dimensiones o rasgos para su representación.

En el Laboratorio de Computación Afectiva del Instituto Tecnológico de Culiacán se desarrolló un corpus en español llamado PersonText que contiene 213 textos, el cual es utilizado para el reconocimiento automático de la personalidad usando inteligencia artificial. Sin embargo, un modelo de aprendizaje máquina obtiene mejores resultados si utiliza un conjunto de datos (corpus) con una cantidad elevada de textos y estos son de buena calidad, mejorando la exactitud en la predicción de los rasgos de la personalidad.

El incrementar el número de textos de PersonText tiene como principal objetivo que los modelos de aprendizaje automático que utilicen este corpus sean útiles para aplicarse en otras áreas, por ejemplo, en la educación.

Para incrementar la cantidad de textos de PersonText se realizó la traducción de corpus en idiomas diferentes al español que contienen textos relacionados al modelo de personalidad Big Five, de estos corpus se recolectaron textos los cuales fueron utilizados para combinarse con los textos y etiquetas de PersonText.

Como resultado, se obtuvo una nueva versión de PersonText, la cual fue llamada PersonText-myPersonality, esta nueva versión cuenta con 2,329 textos etiquetados en cada uno de los rasgos del modelo Big Five. Al probar los modelos de reconocimiento entrenados con este nuevo corpus se obtuvieron buenos resultados al predecir el rasgo de Apertura con una exactitud de alrededor de 70%, sin embargo, al predecir el rasgo de Amabilidad se obtuvo una exactitud de alrededor de 55%.

Palabras clave

Reconocimiento de personalidad,

Corpus en español,

PersonText,

MyPersonality,

Modelo Big Five,

Rasgos de personalidad,

Aprendizaje automático,

Inteligencia artificial,

Redes neuronales.

Índice general

1.	Introducción	1
1.1.	Problema por resolver, priorizándolo	2
1.2.	Objetivo General.....	3
1.3.	Objetivos Específicos	3
1.4.	Justificación	4
2.	Marco teórico	5
2.1.	Modelos de personalidad	5
2.1.1.	Personalidad	5
2.1.2.	Modelo PEN de Eysenck.....	6
2.1.3.	Modelo Alternativo de personalidad de cinco factores	7
2.1.4.	Modelo Big-Five	8
2.2.	Conjuntos de datos	9
2.3.	Corpus.....	10
2.4.	Modelos de reconocimiento	10
2.4.1.	Aprendizaje Automático.....	10
2.4.2.	Clasificación automática de textos	11
2.4.3.	Reconocimiento automático de la personalidad	11
2.4.4.	Modelo Bolsa de Palabras (<i>Bag of Words</i>).....	11
2.4.5.	Redes Neuronales Recurrentes (RNN).....	12
2.4.6.	Perceptrón multicapa	12
2.4.7.	LSTM Sencilla.....	13
2.4.8.	LSTM Bidireccional	13
2.4.9.	CNN.....	13
2.5.	Python.....	14
2.6.	Google Colab.....	14
2.7.	Tensor Flow	15
2.8.	Pandas	15
3.	Estado del arte	16
3.1.	Corpus MyPersonality	16
3.2.	PersonText	16
3.3.	<i>BERT base model (uncased)</i>	17

3.4.	GPT-2	18
4.	Desarrollo del proyecto	19
4.1.	Análisis de técnicas de crecimiento de corpus, diseño e implementación de estrategias para incrementar el corpus <i>PersonText</i>	19
4.1.1.	Traducción <i>MyPersonality</i>	19
4.1.2.	Combinación de corpus <i>PersonText</i> y <i>MyPersonality</i>	22
4.2.	Diseño, entrenamiento y optimización de modelos de reconocimiento automático de la personalidad utilizando el corpus generado.	23
4.2.1.	Preparación del <i>dataset</i>	24
4.2.2.	Construcción de modelos.	26
4.2.3.	Evaluación de resultados obtenidos.	33
4.2.4.	Graficación de los resultados.	35
5.	Pruebas y Resultados	37
5.1.	Resultados de los modelos.	37
5.1.1.	Resultados <i>MyPersonality</i> en español	37
5.1.2.	Resultados <i>PersonText-myPersonality</i>	40
6.	Conclusiones y trabajo futuro	43
6.1.	Conclusiones del proyecto.	43
6.2.	Recomendaciones	44
7.	Bibliografía.	46

Índice de figuras

Figura 2-1 Espacio tridimensional definido por las dimensiones eysenckianas E, N y P (extraído de Ortet, 2001)	7
Figura 2-2 Rasgos típicos que definen cada dimensión. Modelo Psicobiológico de Personalidad de Eysenck: una historia proyectada hacia el futuro (p. 11) por Schmidt, (2010)	7
Figura 2-3 Perceptrón multicapa Tomado de “Multi-layer Perceptrón in TensorFlow”, 2023, https://www.javatpoint.com/multi-layer-perceptron-in-tensorflow	12
Figura 3-1 Vista parcial del corpus PersonText versión Final. Creación de un corpus y clasificación de textos en español para reconocimiento de personalidad. (p. 59) por Roldan, (2022).	17
Figura 4-1 Preparación biblioteca Translator de googlettrans.....	20
Figura 4-2 Función traducir	20
Figura 4-3 Ciclo para traducir los textos de MyPersonality	20
Figura 4-4 Estructura MyPersonality en español	21
Figura 4-5 Estructura de MyPersonality en español ajustado para que coincida con PersonText.....	21
Figura 4-6 Diagrama de Flujo de la traducción de MyPersonality.....	22
Figura 4-7 Proceso de combinación de corpus.....	22
Figura 4-8 Función depurar datos.....	25
Figura 4-9 Elegir el rasgo a evaluar	25
Figura 4-10 Etiquetas del corpus PersonText-myPersonality para las clases de los modelos	26
Figura 4-11 Código Python para obtener la secuencia de enteros de los textos.....	26
Figura 4-12 Código Python del modelo Perceptrón Multicapa.....	27
Figura 4-13 Código Python del modelo LSTM Sencilla.....	28
Figura 4-14 Código Python del modelo LSTM Bidireccional	29
Figura 4-15 Código Python del modelo CNN1D	30
Figura 4-16 Clase para clasificar rasgos de la personalidad con el modelo BERT.....	32
Figura 4-17 Entrenamiento del modelo con BERT en 5 épocas	33
Figura 4-18 Estructura del archivo csv para resultados.....	34
Figura 4-19 Obtener métricas del Perceptrón multicapa	34
Figura 4-20 Guardar los resultados del modelo en el csv de resultados.....	34
Figura 4-21 Cargar archivo de resultados.....	35
Figura 4-22 Valores de los resultados para graficar	35

Índice de tablas

Tabla 1 Mejores resultados.....	42
---------------------------------	----

Capítulo 1

1. Introducción

En el área de psicología, se han realizado muchas investigaciones para caracterizar la personalidad de las personas. Estos estudios han dado como resultado diferentes modelos de personalidad que consideran diversas características para representar la personalidad con rasgos o dimensiones; los cuales se pueden utilizar para identificar a una persona. Las personas se pueden identificar por su personalidad, su apariencia física y la forma de hablar, conocer los rasgos de la personalidad puede ser útil para que algunos sistemas computacionales se adapten a las características de la persona que los utiliza y respondan de acuerdo con sus rasgos.

En este proyecto se utilizó un corpus en español llamado *PersonText* para el reconocimiento automático de la personalidad el cual está basado en el modelo Big Five.

Con la información de *PersonText* se construyeron modelos de reconocimiento automático para predecir cada uno de los rasgos del modelo Big Five, sin embargo, para obtener buenos resultados en estos modelos de reconocimiento es necesario contar un buen dataset.

Con la intención de obtener mejores resultados se realizó un proceso para incrementar *PersonText*, para que posteriormente los modelos de reconocimiento pudieran utilizarlo.

El poder predecir la personalidad de las personas presenta diversos beneficios, por ejemplo, en la educación, saber la personalidad de un estudiante abre la posibilidad de emplear una forma de impartir las clases de manera que el estudiante obtenga mejores resultados en su formación.

Este proyecto contribuye en crear una nueva versión del corpus *PersonText* la cual cuenta con un número mayor de textos en español y la creación de modelos de reconocimiento automático de la personalidad probados con los diferentes rasgos de la personalidad, los cuales se pueden utilizar para tareas específicas en cada rasgo de la personalidad tomando en cuenta su desempeño.

1.1. Problema por resolver, priorizándolo

La personalidad es un rasgo distintivo en cada persona, comúnmente identificamos a cada individuo por su comportamiento, vestimenta, forma de hablar, opiniones y por como interactúa con su entorno. En el área de la psicología se han realizado investigaciones para crear un modelo que logre representar la personalidad, uno de los modelos más reconocidos y aceptados es el modelo Big Five, el cual cuenta con 5 rasgos o dimensiones para representar la personalidad, son los siguientes:

1. Apertura.
2. Sociabilidad.
3. Amabilidad.
4. Responsabilidad.
5. Neuroticismo.

Estos rasgos del modelo Big Five se pueden identificar en una persona mediante una prueba llamada IPIP (*International Personality Item Pool*), esta prueba consta de 200 elementos, cada uno de los cuales se califica en una escala Likert de 5 puntos y tarda aproximadamente 15-20 minutos en completarse.

Se ha demostrado que la prueba IPIP tiene una alta confiabilidad y es válida para predecir otras áreas como el desempeño laboral, el rendimiento académico y la salud mental (John, 2008).

Sin embargo, el tiempo que se necesita para aplicar esta y otras pruebas de personalidad implica que no sea tan eficiente utilizarlas, por esta razón la automatización en la predicción de la personalidad con modelos de reconocimiento automático es una buena alternativa. Para construir estos modelos de reconocimiento automáticos es necesario contar con conjuntos de datos (*dataset*) los cuales contengan información útil para el entrenamiento de los modelos.

Actualmente existen modelos de reconocimiento automático para los sentimientos, sin embargo, el área del reconocimiento automático de la personalidad no ha sido investigada de la misma manera.

Existe un corpus en español llamado *PersonText* que cuenta con 213 textos de opiniones de voluntarios, estos datos fueron recolectados de la plataforma *PersonApp* donde los voluntarios realizaron la prueba IPIP.

A partir de este corpus de textos se puede partir para construir modelos de reconocimiento automático de la personalidad que puedan ser aprovechados en diferentes tareas específicas, sin embargo, para un corpus es de suma importancia la cantidad, diversidad y la calidad de los textos, por esta razón una de las actividades importantes de este trabajo es el incrementar el tamaño de *PersonText*.

1.2. Objetivo General

Incrementar el conjunto de datos *PersonText*, utilizando la plataforma *PersonApp* y la traducción de corpus de textos en idiomas diferentes al español para construir modelos de reconocimiento automático de la personalidad que puedan ser utilizados por diversas aplicaciones.

1.3. Objetivos Específicos

1. Traducir corpus existentes orientados al reconocimiento de la personalidad en lenguajes diferentes al español con el fin de incrementar el corpus *PersonText*.
2. Utilizar la plataforma *PersonApp* para incrementar el corpus *PersonText*.
3. Diseñar, entrenar y optimizar modelos de reconocimiento automático de la personalidad utilizando técnicas tradicionales.
4. Diseñar, entrenar y optimizar modelos de reconocimiento automático de la personalidad utilizando redes neuronales.
5. Realizar comparativa de los resultados obtenidos al utilizar los modelos de reconocimiento automático de la personalidad generados con otros enfoques del estado del arte.

1.4. Justificación

El reconocimiento de la personalidad actualmente se efectúa a través de pruebas estandarizadas donde las personas responden a un conjunto de preguntas de opción múltiple de acuerdo con sus preferencias. Los resultados de estas pruebas arrojan las características de la personalidad de los participantes. Sin embargo, responder pruebas con muchas preguntas tiene la desventaja que los participantes tienden a responder aleatoriamente por cansancio, aburrimiento o desesperación por finalizar. Para subsanar estas desventajas, se desarrolló una aplicación llamada *PersonApp* que implementa un proceso automático en donde a través de videos de una persona y clasificadores de inteligencia artificial se detecten automáticamente los rasgos de personalidad que están presentes de acuerdo con el modelo de los cinco grandes. En este trabajo, se plantea la posibilidad de realizar el reconocimiento de personalidad a través de texto escrito por una persona, para lo cual es necesario contar con un corpus para entrenamiento de los modelos de clasificación.

La creación de un corpus en español orientado al reconocimiento de la personalidad que permita el desarrollo de modelos de reconocimiento automático es de vital importancia para utilizar dichos reconocedores automáticos de la personalidad en Sistemas Tutores Inteligentes que se adapten a la personalidad del estudiante con el fin de que los estudiantes alcancen mejores resultados en el aprendizaje. Así mismo, dichos reconocedores pueden ser de gran utilidad en el ámbito laboral para seleccionar a las personas adecuadas para un puesto con base en el perfil requerido.

Capítulo 2

2. Marco teórico

En esta sección se presentan diversos temas relacionados con las áreas de conocimiento necesarias para abordar el trabajo de tesis. Principalmente se abordan tres temas sobre modelos de personalidad, conjuntos de datos y corpus, y modelos de clasificadores automáticos.

2.1. Modelos de personalidad

Los modelos de la personalidad son marcos teóricos que tienen el objetivo de representar la personalidad de las personas, estos modelos tienen principalmente en común que representan o identifican la personalidad mediante un conjunto de rasgos que deben ser estables y universales, esto significa que se puede identificar la personalidad de las personas con estos rasgos o también llamadas dimensiones (McCrae, 1999).

Algunos ejemplos de modelos de personalidad son el de PEN de Eysenck, Modelo Alternativo de personalidad de Cinco factores y el modelo Big Five.

Estos modelos tienen diferencias entre sí, sobre todo los rasgos que utilizan para medir la personalidad que identifican y la forma en la que explican la personalidad, aun así, tienen en común el mismo objetivo que es comprender la estructura de la personalidad, por lo tanto, para este proyecto, lo ideal es basarse en uno de estos modelos de personalidad.

2.1.1. Personalidad

La definición de la Personalidad de Eysenck (1994) dice que “La personalidad puede entenderse como una organización más o menos estable y duradera del carácter, el

temperamento, el intelecto y el físico de una persona, que determina su adaptación única al ambiente”. De acuerdo con esta definición, la personalidad se puede entender como el conjunto de rasgos que muestran las personas, tanto físicas, intelectuales y sentimentales al interactuar con otras personas de una sociedad. De esta manera, entendiendo lo que significa personalidad podemos utilizar uno de los métodos existentes para poder representarla con el fin de usar los resultados para tareas útiles que beneficien a una sociedad, o en este caso, a la educación.

2.1.2. Modelo PEN de Eysenck

El modelo PEN (Eysenck, 1967) fue creado por Eysenck a partir del año 1950 con el fin de representar la personalidad de las personas, utilizando principalmente tres dimensiones las cuales son:

1. Extraversión: Esta dimensión representa la tendencia en las personas a ser más sociables y extrovertidas, las cuales buscan frecuentemente la interacción con otras personas.
2. Neuroticismo: Representa la tendencia en las personas a experimentar emociones negativas, presentan frecuentemente depresión, problemas emocionales, ira fácilmente y estrés.
3. Psicoticismo: Representa la tendencia en las personas a ser impersonales, con falta de empatía, frecuentemente las personas con esta característica son agresivas.

El modelo de Eysenck representa con estas tres dimensiones la personalidad de las personas, como se muestra en la figura 2-1 los tres ejes de este modelo representan a cada una de las dimensiones: Neuroticismo, Extraversión y Psicoticismo.

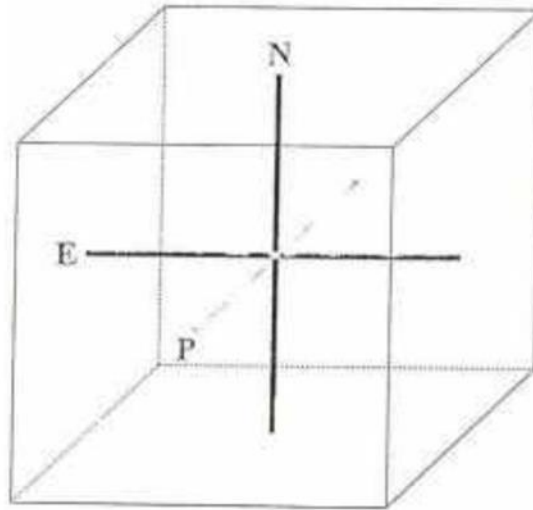


Figura 2-1 Espacio tridimensional definido por las dimensiones eysenckianas E, N y P (extraído de Ortet, 2001)

En la figura 2-2 se muestran algunos rasgos que presentan las personas en cada dimensión del modelo de Eysenck.

<i>N</i>
<i>tristeza – depresión - timidez – ansiedad – tensión - miedo – culpa – irracionalidad - vergüenza – mal humor – emotividad - preocupación</i>
<i>E</i>
<i>sociabilidad – actividad – asertividad – despreocupación – dominancia – búsqueda de sensaciones (socializada) – osadía - espontaneidad - rapidez</i>
<i>P</i>
<i>impulsividad – agresividad – hostilidad - frialdad – egocentrismo – falta de empatía – crueldad - creatividad – falta de conformismo - dureza mental</i>

Figura 2-2 Rasgos típicos que definen cada dimensión. Modelo Psicobiológico de Personalidad de Eysenck: una historia proyectada hacia el futuro (p. 11) por Schmidt, (2010)

El modelo de Eysenck proporcionó las bases para nuevas investigaciones acerca de la personalidad, así como nuevos modelos, los cuales son relevantes en esta investigación para la predicción automática de la personalidad.

2.1.3. Modelo Alternativo de personalidad de cinco factores

El modelo Alternativo de cinco factores fue creado por Marvin Zuckerman (Zuckerman, 1994) para representar la personalidad de las personas con cinco factores que representan un rasgo, los cuales describe como:

1. Búsqueda impulsiva de sensaciones (imps): las personas que presentan un alto nivel en este rasgo frecuentemente buscan emociones fuertes, nuevas experiencias comúnmente riesgosas.
2. Neuroticismo-ansiedad (N-Anx): las personas con alto nivel en este rasgo tienden más a tener problemas emocionales, estrés, ira y preocupación.
3. Agresión-hostilidad (Agg-Host): las personas que presentan un alto nivel en este rasgo se dice que son agresivas, violentas, impulsivas, sin empatía y tienen conductas delictivas.
4. Sociabilidad (Sy): las personas que tienen un alto nivel en este rasgo participan más en actividades sociales, tienen más contacto con otras personas en actividades grupales, culturales y sociales. También tienen una conducta asertiva y extrovertida.
5. Actividad (Act): las personas que tienen un alto nivel de este rasgo realizan mucha actividad física, como deportes, tienen mucha energía y tienden a ser inquietos.

2.1.4. Modelo Big-Five

The Big Five Factor Structure es un modelo de personalidad que se dio como resultado de diversas investigaciones de la personalidad y otros de sus modelos, el cual representa la personalidad humana en cinco dimensiones o rasgos que son medibles o que se pueden identificar (Goldberg, 1990), los cuales son:

1. Extraversión: es la dimensión en las personas que dice que tan extrovertida y sociable es, tienden a tener mayor interacción social y disfrutan de la compañía de los demás. Por el contrario, una persona que presenta menos este rasgo tiende a ser más reservada y disfrutan menos la compañía de otras personas.
2. Amabilidad: esta dimensión representa a las personas que tienen un nivel de cooperación mayor con las demás personas, tienen cualidades como la amabilidad, el ser servicial y la confianza al convivir con otras personas. Las personas que presentan menos este rasgo de la personalidad son desagradables, es decir, son competitivas, desconfiadas y menos serviciales.
3. La conciencia: es la dimensión representa a las personas con cualidades de organización, confiabilidad y que es responsable con sus obligaciones. Las personas

con bajo nivel en esta dimensión tienden a ser indisciplinadas, irresponsables y no cumplen las metas establecidas.

4. Neuroticismo: es la dimensión que representa la estabilidad emocional en las personas, entre más alto es el nivel que presenta una persona en este rasgo tiende a tener problemas emocionales, emociones negativas y se preocupan fácilmente.
5. Apertura a la experiencia: las personas que presentan un alto nivel en este rasgo tienden a tener curiosidad, iniciativa, presentan interés al cambio, a la imaginación y sobre todo a nuevas experiencias. Las personas que presenta un bajo nivel tienden a ser evitar o presentar menos interés en los cambios o nuevas ideas.

Este modelo, es uno de los más utilizados en investigaciones debido a que ha demostrado tener buenos resultados al predecir los rasgos de la personalidad, esta es la principal razón por la cual este modelo se utiliza principalmente en esta investigación. Siendo de utilidad para aplicarse en diferentes áreas, por ejemplo, la educación, el rendimiento laboral y la interacción social. Esto es tomando en cuenta que no es una representación exacta, si no que sirve como un criterio más al representar la personalidad de una persona.

2.2. Conjuntos de datos

Los conjuntos de datos o *datasets* en inglés, son colecciones de datos que se almacenan con el fin de ser analizados, estas colecciones pueden contener diferentes tipos de información, por ejemplo, textos e imágenes, pueden ser grandes y también pequeños. Un tipo de *dataset* es el estructurado y otro tipo es el no estructurado (Cios, 2014).

Un *dataset* no estructurado se refiere a que los datos son libres, no cumplen con un formato o algunas reglas como si lo hacen los estructurados, se dice que, al no estar organizados se dificulta el análisis de este. Por esta razón, los datos tienden a tener una gran variedad, lo que significa un beneficio para el *dataset*.

Un *dataset* estructurado cuenta datos organizados, por ejemplo, en tablas, las cuales pueden tener diferentes columnas, de esta manera se facilita el análisis de los datos.

Contar con un buen *dataset* aporta beneficios a un trabajo de investigación, al existir diferentes tipos de conjuntos de datos se puede elegir uno que se adapte mejor a las tareas que se deben realizar.

2.3. Corpus

Bolívar (2013) establece que “El corpus de la investigación se define como el conjunto de materiales lingüísticos o no, que conforman el objeto de estudio en una investigación o en una parte de ella.”.

Además, Bolívar (2013) explica que una característica especial de los corpus es que están contruidos comúnmente con un propósito de investigación o de alguna disciplina, por eso es importante distinguir un conjunto de datos de textos entre un verdadero corpus y un requisito es que los textos se deben mantener en la forma original en el que fueron escritos por lo que no deben ser manipulados. Esto quiere decir que, en los corpus debemos respetar los textos aún con faltas de ortografía u otros elementos que puedan contener, posteriormente al utilizar estos textos para diferentes tareas, se puede llegar a procesar los textos siempre y cuando se mantenga el corpus original sin modificaciones.

2.4. Modelos de reconocimiento

Un modelo de reconocimiento es un tipo de modelo que utiliza aprendizaje automático para clasificar diferentes objetos u otras cosas, por ejemplo, la voz e imágenes. Otro uso de estos modelos es el que son útiles para el procesamiento del lenguaje natural (Bishop, 2006). Estos modelos funcionan aprendiendo a identificar patrones en los datos que son ingresados.

2.4.1. Aprendizaje Automático

El aprendizaje automático es un tipo de inteligencia artificial (IA) que permite que las aplicaciones de software sean más precisas en la predicción de resultados sin estar programadas explícitamente para hacerlo. Los algoritmos de aprendizaje automático utilizan datos históricos como entrada para predecir nuevos valores de salida (Alpaydin, 2016).

Existen tres tipos de aprendizaje automático de acuerdo con Mitchell (1997), los cuales son:

1. Aprendizaje supervisado: En este aprendizaje, el modelo se entrena con un conjunto de datos con etiquetas, de esta manera el modelo aprende a dar mejores predicciones tomando como referencia los valores esperados que son las etiquetas que fueron introducidas.
2. Aprendizaje no supervisado: En este aprendizaje, los datos de entrada del modelo no cuentan con etiquetas, por lo tanto, el modelo debe encontrar patrones en los datos sin la guía de las etiquetas, esto sirve, por ejemplo, para realizar agrupaciones de datos.
3. Aprendizaje por esfuerzo: En este tipo, el modelo realiza el aprendizaje mediante prueba y error. Esto se realiza recompensando al modelo por obtener resultados deseados, y penalizado por obtener resultados erróneos.

El aprendizaje automático es una herramienta muy poderosa, la cual se puede utilizar para tener modelos que realicen tareas específicas que sean de utilidad en diferentes contextos.

2.4.2. Clasificación automática de textos

De acuerdo con (Manning, 2009) la clasificación automática de texto es un proceso para categorizar un texto proporcionado como entrada. Para realizar la clasificación de textos se utilizan comúnmente técnicas de procesamiento del lenguaje natural, aprendizaje automático y sistemas basados en reglas.

2.4.3. Reconocimiento automático de la personalidad

Hernández (2016) dice que el Reconocimiento automático de la personalidad “consiste en la clasificación automática de los rasgos de personalidad de autores, a través de fragmentos de texto que hayan escrito”. Y explica que se necesitan técnicas de diversas disciplinas, tales como la Lingüística, Psicología y de la informática como la Minería de Datos.

2.4.4. Modelo Bolsa de Palabras (*Bag of Words*)

Arias (2016) define la bolsa de palabras o *Bag of Words* como “una técnica utilizada para simplificar la representación de los objetos. Principalmente se utiliza para la representación de documentos a través de las palabras que los constituyen.”

Para la representación de las imágenes se utilizan las palabras que contiene la descripción de una imagen, esta descripción debería especificar partes de la imagen, textura y los elementos que contiene, de esta manera se puede obtener a una correcta representación de la imagen.

2.4.5. Redes Neuronales Recurrentes (RNN)

Bonet (2007) explica que una RNN o red neuronal “puede ser caracterizada por el modelo de la neurona, el esquema de conexión que presentan sus neuronas, o sea su topología, y el algoritmo de aprendizaje empleado para adaptar su función de cómputo a las necesidades del problema particular”. Además, menciona que existen diferentes modelos de neuronas, las cuales tienen diferentes funciones de activación para obtener los valores de salida de cada neurona, conocer sobre los diferentes modelos o arquitecturas de las redes neuronales es importante para esta investigación.

2.4.6. Perceptrón multicapa

Según (Toro Bayona & Alberto, 2012), el modelo Perceptrón Multicapa se compone de capa de entrada, capas ocultas y capa de salida como se muestra en la figura 2-3 Cada capa está compuesta por una serie de neuronas que se encargan de recibir, procesar y enviar datos hacia otras neuronas procesando la información mediante distintas funciones matemáticas.

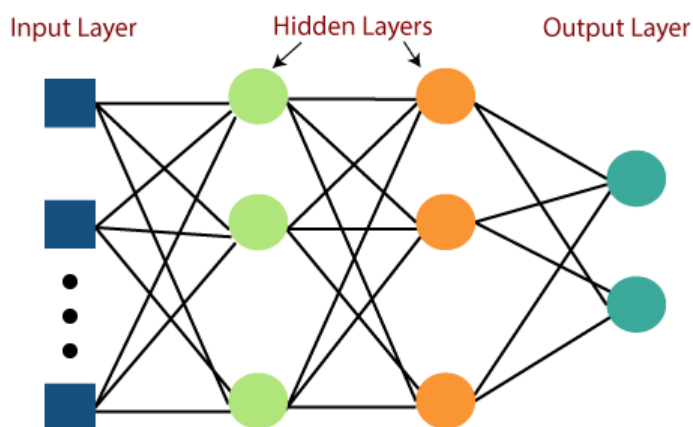


Figura 2-3 Perceptrón multicapa Tomado de “Multi-layer Perceptrón in TensorFlow”, 2023, <https://www.javatpoint.com/multi-layer-perceptron-in-tensorflow>

Cada neurona tiene en cada una de sus entradas valores llamados “pesos” los cuales se deben ajustar automáticamente durante el entrenamiento automático para obtener una salida mediante una función de activación en la que la salida corresponda al valor esperado, esto en el caso de un entrenamiento supervisado.

El Perceptrón multicapa es un modelo de red neuronal muy conocido y utilizado. Y en esta investigación se hace mención, gracias a que es útil su aplicación en el procesamiento del lenguaje natural.

2.4.7. LSTM Sencilla

Las redes LSTM son una variante a las RNN o su versión mejorada, por lo tanto, tienen diferencias, que en el desarrollo de esta investigación son relevantes por lo útiles que son para el ámbito del procesamiento del lenguaje natural.

Explica (Fernández J. J., 2021) que las RNN no son adecuadas para modelos que necesitan mayor memoria a largo plazo en el estado actual u oculto de cada capa de la red neuronal.

Y para solucionar este problema se desarrollaron las redes LSTM que tienen una larga memoria de corto plazo, las cuales mantienen en memoria información durante más tiempo.

2.4.8. LSTM Bidireccional

Montanés (2018) explica que Zhou presento las redes LSTM bidireccionales, que además de preservar la información pasada también logran preservar información también futura. Este tipo de redes son unas de las más utilizadas para la representación de textos por su eficiencia en la actualidad.

2.4.9. CNN

Las *Convolutional Neural Networks* (CNN) conocidas como redes convolucionales, son descritas por (Fernández G. , 2020) como modelos de aprendizaje profundo que se utilizan para analizar y clasificar características de una imagen de entrada. Cada neurona dentro de una CNN se organiza en capas, y cada conjunto de neuronas se encarga de analizar una pequeña región o característica específica de la imagen.

La capa convolucional es responsable de crear un mapa de características aplicando un filtro a la entrada de la red. Este mapa muestra la ubicación y el tipo de características detectadas en la imagen de entrada.

La capa de agrupación reduce la cantidad de información del mapa de características, manteniendo solo la información esencial.

La capa de aplanamiento o *flatten* convierte la salida de las capas convolucionales en un vector unidimensional.

Finalmente, la capa conectada es una capa sencilla de una red neuronal multicapa que se encarga de realizar la clasificación final basada en las características extraídas.

En resumen, las CNN utilizan capas convolucionales para tomar características de una imagen, capas de agrupación para reducir la información, una capa de aplanamiento para convertir la salida en un vector y una capa conectada para realizar la clasificación.

2.5. Python

Challenger Pérez (2014) dice que Python es “un lenguaje de alto nivel ya que contiene implícitas algunas estructuras de datos como listas, diccionarios, conjuntos y tuplas, que permiten realizar algunas tareas complejas en pocas líneas de código y de manera legible”. Además, explica que Python es un lenguaje multiparadigma, por lo tanto, puedes trabajar con programación orientada a objetos, imperativa y funcional.

Una de las características de Python más importante que se pueden destacar es su facilidad de trabajar con estructuras de datos, al trabajar con inteligencia artificial se aprovecha esta característica.

En este trabajo se utilizaron una gran cantidad de bibliotecas externas, esa es otra de las fortalezas que tiene este lenguaje, el poder instalar bibliotecas que sean de utilidad para realizar diversas tareas.

2.6. Google Colab

Google Colab es un servicio en la nube basado en *Jupyter Notebooks* el cual te permite programar y ejecutar código Python. Además, te permite configurar un entorno de ejecución con una GPU, subir y descargar archivos (Bodero, 2020).

Uno de los principales enfoques de *Google Colab* es la investigación del aprendizaje automático, por lo tanto, aporta algunas cualidades que facilitan la investigación, por ejemplo, poder compartir una notebook para trabajar en colaboración con otras personas.

Google Colab es una opción ideal para este trabajo debido a que permite construir modelos de reconocimiento automático sin necesidad de contar con una computadora con una alta capacidad de procesamiento, ya que todo está en la nube y accedes mediante el navegador.

2.7. Tensor Flow

TensorFlow es una biblioteca de Python y otros lenguajes como JavaScript que es de código abierto para el cálculo numérico mediante grafos. Popularmente se utiliza para aplicaciones de aprendizaje automático e inteligencia artificial. Esta biblioteca fue desarrollada por el equipo de *Google Brain*, Google lo utiliza para muchos de sus productos, por ejemplo, *Google Search*, *Google Photos* y *Google Translate* (TensorFlow, 2023).

En este trabajo *TensorFlow* fue la biblioteca principal y que más ayuda aportó, ya que permite el entrenamiento de modelos de reconocimiento automático.

2.8. Pandas

De acuerdo con la página oficial de Pandas (2023), Pandas es una herramienta de manipulación y análisis de datos de código abierto rápida, potente, flexible y fácil de usar, construida sobre el lenguaje de programación Python.

Esta biblioteca se utiliza principalmente para la creación de objetos llamados *DataFrame*, estos objetos cuentan con una estructura de filas y columnas que permiten almacenar, manipular y acceder a los datos de manera rápida, estos son muy útiles en esta investigación ya que estos están directamente relacionados al entrenamiento de los modelos de reconocimiento que se construyeron.

Capítulo 3

3. Estado del arte

En este capítulo se presentan las investigaciones relacionadas con este trabajo de tesis. A continuación, se describe un corpus de personalidad con textos en inglés, después se describe un corpus en español con textos relacionados a la personalidad y el modelo Big-Five, se describe al igual se menciona un modelo pre entrenado con un conjunto de datos en inglés y por último se menciona un modelo pre entrenado manera auto supervisada que utiliza la biblioteca *Transformers*.

3.1. Corpus MyPersonality

MyPersonality es un corpus de personalidad generado a partir de una aplicación de Facebook que permitía a sus usuarios participar en investigaciones psicológicas completando un cuestionario de personalidad. A los usuarios que participaron se les ofreció retroalimentación sobre sus puntajes. El corpus fue creado en 2007 por David Stillwell y en 2009, David se unió a Michal Kosinski (Stillwell, 2015).

En 2018, los creadores decidieron dejar de compartir los datos con otros académicos. Mantener el conjunto de datos, examinar los proyectos, responder a las consultas y cumplir con varias regulaciones se había vuelto demasiado difícil para ellos (Stillwell, 2015).

3.2. PersonText

La definición de *PersonText* de acuerdo con Roldan (2022) es “un conjunto de datos que contiene textos de opiniones de voluntarios, y datos de información personal de los mismos, como su edad, sexo y las evaluaciones de los cinco rasgos de personalidad del modelo Big-Five”. Además, Roldan (2022) explica que los datos fueron recolectados de la plataforma *PersonApp*, cuyos datos son de una prueba estandarizada llamada IPIP que se basa en el modelo OCEAN para la representación de la personalidad.

PersonText cuenta con 213 textos etiquetados con información de cada rasgo de la personalidad del modelo Big Five, por lo tanto, en esta investigación es de gran utilidad para la implementación en modelos de reconocimiento automático de la personalidad y otras posibles tareas. En la figura 3-1, se muestra la estructura del corpus *PersonText*.

uid	texto	fecha_prueb	apertura	presenta_ap	responsabili	presenta_re	sociabilidad	presenta_so	amabilidad	presenta_an	neuroticism	presenta_ne	fechaNadimi	sexo	edad_calcul	Experim
n84XTQwKh	Un tema qui	23/04/2021	0.7	Si	0.8	Si	0.6	No	0.9	Si	0.7	Si	03/09/1991	Femenino	29	Exp 1-1
iXWwNMyE	Si a mí me p	04/03/2021	0.8	Si	0.8	Si	0.3	No	0.8	Si	0.9	Si	23/07/1994	Masculino	26	Exp 1-1
n3a5RSloo	C2 Qué tal a mí	04/03/2021	0.5	No	0.5	No	0.6	No	0.6	No	0.7	Si	24/10/1995	Masculino	25	Exp 1-1
CTyZqFJp	O: Me gusta mi	23/04/2021	1	Si	0.4	No	0.6	No	0.9	Si	0.7	Si	25/10/1980	Femenino	40	Exp 1-1
B6Ajr18EC	W: Les hablaré	04/03/2021	0.7	Si	0.8	Si	0.7	Si	0.5	No	0.9	Si	02/12/1995	Masculino	25	Exp 1-1
pkcrgdVam	v: Hola pues le	23/04/2021	0.8	Si	0.6	No	0.5	No	0.8	Si	0.8	Si	15/11/1986	Femenino	34	Exp 1-1
Z3btzkauQ	O: Hola Pues a	23/04/2021	0.8	Si	0.6	No	0.8	Si	0.6	No	1	Si	22/09/1995	Masculino	25	Exp 1-1
W5VPHUUV	I: Hola Mi non	23/04/2021	0.7	Si	0.6	No	0.5	No	0.9	Si	0.5	No	05/05/1987	Femenino	33	Exp 1-1
aq4nXJ73Z	ed: Hola Mi non	23/04/2021	0.8	Si	0.6	No	0.4	No	0.8	Si	0.3	No	07/08/1995	Masculino	25	Exp 1-1
bzGverJUCH	Y: En esta ocas	10/02/2021	0.8	Si	0.8	Si	0.6	No	0.8	Si	0.6	No	15/03/1984	Masculino	36	Exp 1-1
E340bUlm6	P: El tema del	26/03/2021	0.6	No	0.7	Si	0.6	No	0.7	Si	0.3	No	01/03/1996	Masculino	25	Exp 1-1
T8rWffMhL	g: El otro día e	26/03/2021	0.7	Si	0.9	Si	0.9	Si	0.5	No	0.5	No	12/04/1994	Masculino	26	Exp 1-1
3Fh0E2ixc	O: Bueno yo so	28/04/2021	0.9	Si	0.9	Si	0.6	No	0.6	No	0.9	Si	02/09/1996	Masculino	24	Exp 1-1
21850VvAn	f: Bueno y con	26/03/2021	0.6	No	0.4	No	0.6	No	0.9	Si	0.8	Si	30/01/1997	Masculino	24	Exp 1-1
RSBZRQmX	t: Bueno uno c	26/03/2021	0.6	No	0.7	Si	0.4	No	0.7	Si	0.3	No	09/06/1996	Femenino	24	Exp 1-1
fA24EvOc2	7C: Bueno un ur	26/04/2021	0.7	Si	0.6	No	0.8	Si	0.9	Si	0.2	No	24/12/1986	Femenino	34	Exp 1-1
m8sp2e9P	x: Bueno pues	26/03/2021	0.9	Si	0.7	Si	0.2	No	0.7	Si	0.6	No	10/07/1997	Masculino	23	Exp 1-1
4F7HIELAo	J: A mí me gus	26/03/2021	0.8	Si	0.9	Si	0.6	No	0.7	Si	0.8	Si	06/11/1996	Masculino	24	Exp 1-1
J5ds52xMCI	: A mí me gus	26/04/2021	0.8	Si	0.8	Si	0.8	Si	1	Si	0.3	No	06/09/1995	Femenino	25	Exp 1-1
n84XTQwKh	Bueno mis f	23/04/2021	0.7	Si	0.8	Si	0.6	No	0.9	Si	0.7	Si	03/09/1991	Femenino	29	Exp 1-2
iXWwNMyE	f: Pues porqu	04/03/2021	0.8	Si	0.8	Si	0.3	No	0.8	Si	0.9	Si	23/07/1994	Masculino	26	Exp 1-2
n3a5RSloo	C2 Una de mis	04/03/2021	0.5	No	0.5	No	0.6	No	0.6	No	0.7	Si	24/10/1995	Masculino	25	Exp 1-2
CTyZqFJp	O: Yo creo que	23/04/2021	1	Si	0.4	No	0.6	No	0.9	Si	0.7	Si	25/10/1980	Femenino	40	Exp 1-2
B6Ajr18EC	W: Bueno pues	04/03/2021	0.7	Si	0.8	Si	0.7	Si	0.5	No	0.9	Si	02/12/1995	Masculino	25	Exp 1-2
pkcrgdVam	v: Bueno Mi pr	23/04/2021	0.8	Si	0.6	No	0.5	No	0.8	Si	0.8	Si	15/11/1986	Femenino	34	Exp 1-2
Z3btzkauQ	O: Hola buena	26/03/2021	0.8	Si	0.6	No	0.8	Si	0.6	No	1	Si	22/09/1995	Masculino	25	Exp 1-2

Figura 3-1 Vista parcial del corpus *PersonText* versión Final. Creación de un corpus y clasificación de textos en español para reconocimiento de personalidad. (p. 59) por Roldan, (2022).

3.3. BERT base model (uncased)

De acuerdo con la página de (Hugging Face, 2023) BERT es un modelo que utiliza la biblioteca Transformers, el cual se ha pre entrenado con un gran conjunto de datos en inglés de manera auto supervisada. En este proceso de preentrenamiento, el modelo utiliza textos sin procesar y sin etiquetas, de esta manera se aprovechan una gran cantidad de datos disponibles para el público. El preentrenamiento de BERT tiene objetivos principales. El primero es utilizar el Modelo de Lenguaje Enmascarado (MLM), donde se enmascara aleatoriamente el 15% de las palabras en una oración y el modelo debe predecir esas palabras enmascaradas al procesar la oración completa. Esta es una de las principales diferencias con las redes neuronales recurrentes tradicionales que ven las palabras secuencialmente y de los modelos autorregresivos que enmascaran internamente los tokens futuros. Esta estrategia permite que BERT aprenda una representación bidireccional de las oraciones.

Además, BERT hace la Predicción de la Siguiente Oración (NSP por sus siglas en inglés), donde se concatenan dos oraciones enmascaradas como entradas durante el

preentrenamiento. Estas oraciones pueden estar adyacentes en el texto original o no, es decir, que una oración sigue de la otra. El modelo debe predecir si las dos oraciones estaban en secuencia o no.

Con este proceso, el modelo BERT posee una representación interna del idioma inglés que se puede utilizar para extraer características útiles en tareas específicas o personalizadas. Por lo tanto, se puede ajustar el modelo BERT para que realice una actividad específica relacionada a esta investigación.

3.4. GPT-2

GPT-2 de acuerdo con la página de (Hugging Face, 2023) es un modelo de que utiliza la biblioteca Transformers, el cual se ha pre entrenado con un gran corpus de datos en inglés de manera auto supervisada. En el preentrenamiento, el modelo utiliza textos sin procesar y sin etiquetas, por lo tanto, le permite aprovechar una gran cantidad de datos disponibles públicamente.

En detalle, las entradas son secuencias continuas de texto con una longitud determinada, y los objetivos son la misma secuencia, pero desplazada una posición hacia la derecha. GPT-2 utiliza un mecanismo de máscara interna para asegurarse de que las predicciones para el token "i" solo utilicen las entradas desde el primer token hasta el "i", excluyendo los tokens futuros.

Capítulo 4

4. Desarrollo del proyecto

En este capítulo se describen las actividades realizadas en el desarrollo de este proyecto, todas las actividades en conjunto fueron pensadas para alcanzar los objetivos establecidos, se realizaron en orden y cada una de ellas aporta valor al proyecto.

4.1. Análisis de técnicas de crecimiento de corpus, diseño e implementación de estrategias para incrementar el corpus *PersonText*.

Esta es una de las actividades más importantes, debido a que se analizaron diferentes opciones para incrementar el corpus, la opción elegida fue utilizar corpus en idiomas diferentes al español, para realizar una combinación, se revisó que el corpus elegido contara con información de cada uno de los cinco rasgos del modelo Big Five al igual que *PersonText* para que fueran compatibles.

4.1.1. Traducción *MyPersonality*

Para incrementar el tamaño del corpus *PersonText* se realizó la traducción del corpus en inglés llamado *MyPersonality*, esta versión del corpus contiene 2221 textos en inglés.

Se utilizó *Google Colab* para ejecutar *notebooks* de *Jupyter*, en la se importó la biblioteca de Python llamada *pandas* para que fuera posible el trabajar con objetos *DataFrame*. En el entorno de ejecución se cargó el corpus de textos *MyPersonality* al cual se le aplicaron funciones de limpieza con el objetivo de eliminar textos que no cumplieran con ciertos criterios, por ejemplo, los textos con menos de 20 palabras.

Para realizar la traducción se utilizó la biblioteca de Google llamada *googletrans*, debido a que en las pruebas previas fue la que mejor resultados proporciono.

Se establecieron las especificaciones para el objeto de la biblioteca para que traduzca del inglés al español como se muestra en la figura 4-1.

```
from googletrans import Translator, constants
from pprint import pprint

[ ] # Inicializa la API Google API translator
translator = Translator()

[ ] # translate a english text to spanish text
translation = translator.translate("Note that the Translator class has several optional arguments", src="en", dest="es")
print(f"{translation.origin} ({translation.src}) --> {translation.text} ({translation.dest})")
print(translation.text)

Note that the Translator class has several optional arguments (en) --> Tenga en cuenta que la clase Traductor tiene varios argumentos opcionales (es)
Tenga en cuenta que la clase Traductor tiene varios argumentos opcionales
```

Figura 4-1 Preparación biblioteca Translator de googletrans

Se definió una función que realiza la traducción de un texto de entrada por parámetro y que retorna el texto en español como se muestra en el código de la figura 4-2.

```
def traducir(texto):
    resultado = translator.translate(texto, src="en", dest="es")
    return resultado
```

Figura 4-2 Función traducir

Posteriormente se realizó la traducción de todos los textos que contenía el *dataframe* de MyPersonality en ingles con un ciclo el cual recorre todos los textos y llama la función para traducir definida anteriormente. Como se muestra en el fragmento de código de la figura 4-3.

```
mpEspañol= pd.DataFrame()
mpEspañol = datos_limpios.copy()
textos = []
confiabilidad = []
for x in datos_a_lista:
    t=traducir(x)
    textos.append(t.text)
    confiabilidad.append([t.extra_data['confidence']])
```

Figura 4-3 Ciclo para traducir los textos de MyPersonality

La estructura del nuevo *dataframe* resulto ser similar a la del corpus *PersonText*, sin embargo, fue necesario modificar los nombres de algunos atributos para que existiera una compatibilidad entre los dos corpus y se pudieran fusionar.

En la figura 4-4 se muestra la estructura del *dataframe* con la traducción de *MyPersonality*.

```
mpEspañol.head()
```

	#AUTHID	STATUS	sEXT	sNEU	sAGR	sCON	sOPN	cEXT	cNEU	cAGR	cCON	cOPN	NumOfWords	confiabilidad_traducción
2	b7b7764cfa1c523e4e93ab2a79a946c4	está adolorida y quiere que el nudo de músculo...	2.65	3.0	3.15	3.25	4.4	n	y	n	n	y	26	1.0
8	b7b7764cfa1c523e4e93ab2a79a946c4	estaba a punto de terminar una pintura digital...	2.65	3.0	3.15	3.25	4.4	n	y	n	n	y	27	1.0
30	b7b7764cfa1c523e4e93ab2a79a946c4	tiene un reloj que hace juego con sus gafas. D...	2.65	3.0	3.15	3.25	4.4	n	y	n	n	y	21	1.0
32	b7b7764cfa1c523e4e93ab2a79a946c4	no puede sentarse en su sofá porque sabe que ...	2.65	3.0	3.15	3.25	4.4	n	y	n	n	y	26	1.0
33	b7b7764cfa1c523e4e93ab2a79a946c4	debería pasar menos tiempo escribiendo capitul...	2.65	3.0	3.15	3.25	4.4	n	y	n	n	y	26	1.0

Figura 4-4 Estructura *MyPersonality* en español

Para realizar la modificación del encabezado del nuevo *dataframe* se utilizó la función llamada “rename()” que es llamada al *dataframe*, en la figura 4-5 se muestra el *dataframe* después del ajuste en el encabezado para que coincida con el corpus *PersonText*.

uid	texto	apertura	presenta_apertura	responsabilidad	presenta_responsabilidad	sociabilidad	presenta_sociabilidad	maabilidad	presenta_maabilidad	neuroticismo	presenta_neuroticismo	NumOfWords
946c4	está adolorida y quiere que el nudo de músculo...	4.4	y	3.25	n	2.65	n	3.15	n	3.0	y	26

Figura 4-5 Estructura de *MyPersonality* en español ajustado para que coincida con *PersonText*

Para finalizar, se realizó la exportación del nuevo corpus a un archivo de tipo csv, de esta manera se guardó el nuevo corpus de la traducción de *MyPersonality* que se utilizó en posteriores actividades.

En la figura 4-6 se resume el flujo de traducción del corpus *MyPersonality*.



Figura 4-6 Diagrama de Flujo de la traducción de *MyPersonality*

4.1.2. Combinación de corpus *PersonText* y *MyPersonality*

De forma general, el proceso realizado para la combinación del corpus *PersonText* y *MyPersonality* en español se muestra en la figura 4-7. Primero se cargó el corpus *MyPersonality*, se realizó una limpieza, se modificó su estructura para que coincidiera con *PersonText*, después se recolectaron los elementos de *PersonText* de manera aleatoria para concluir con la inserción a una nueva estructura llamada *PersonText-myPersonality*.

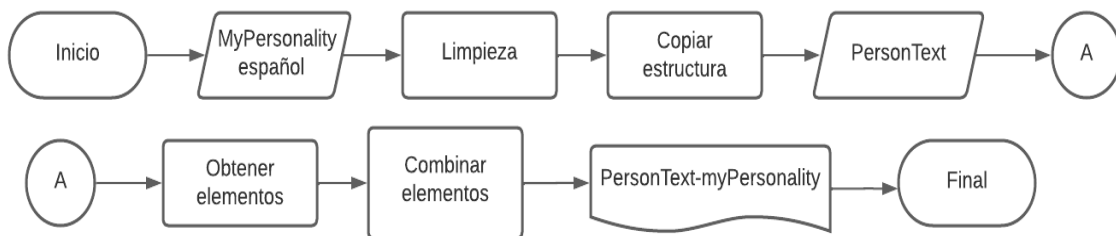


Figura 4-7 Proceso de combinación de corpus

En el trabajo realizado para incrementar el corpus se utilizó el lenguaje Python en notebooks de *Google Colab*, las cuales permiten la ejecución en orden de las líneas de código por secciones.

Con la biblioteca Pandas se cargaron los dos corpus en su respectivo *dataframe* para manipularlos en las siguientes tareas. Posteriormente se realizó un proceso de limpieza de textos, en los cuales se descartaron los textos del corpus *MyPersonality* en español que no cumplieran con ciertas características como el número de palabras mayor a 20.

Para la combinación de ambos corpus que tenían diferente estructura fue necesario agregar algunas columnas con la intención de respetar la estructura original de *PersonText*. Las columnas que fueron agregadas fueron 'fecha_prueba', 'fechaNacimiento', 'sexo', 'edad_calculada' y 'Experimento'.

Debido a que la cantidad de textos en el corpus *MyPersonality* en español era mayor que la de *PersonText* se realizaron instrucciones las cuales retiraban elementos del corpus *PersonText* para insertarlos en el corpus *MyPersonality*.

La Primer instrucción fue obtener la cantidad de elementos que existían en *PersonText*, los cuales eran 213, esto con la finalidad de realizar una serie de instrucciones que se ejecutaran una vez por cada texto del corpus.

Después de obtener la cantidad de elementos a tomar del corpus se obtuvo una muestra aleatoria de textos del corpus y se realizó una iteración en las muestras para insertarlas en el corpus *MyPersonality* en español.

Al final se obtuvo un nuevo corpus que fue llamado 'PersonText-myPersonality.csv' el cual contiene 2329 textos y se exportó a un documento de formato csv para ser utilizado en la construcción de modelos de reconocimiento automático de la personalidad.

4.2. Diseño, entrenamiento y optimización de modelos de reconocimiento automático de la personalidad utilizando el corpus generado.

En esta actividad, se realizó la construcción de diferentes modelos de reconocimiento, la mayoría con redes neuronales, fueron entrenados tanto como con el corpus *MyPersonality* en español, tanto como con el corpus *PersonText-myPersonality*.

4.2.1. Preparación del *dataset*.

Para poder trabajar con la información del corpus se utilizó la biblioteca Pandas como en actividades anteriores. La preparación del *dataset* fue necesaria para obtener datos compatibles y que sirvan de entrada en el entrenamiento de nuestros modelos de reconocimiento automático. Primero se aplicó un filtro al *dataset* para asegurar que todos los textos tuvieran una cantidad de palabras mayor a 20 de esta manera el entrenamiento obtiene mejores resultados.

Con el objetivo de tener datos sin caracteres extraños se aplicaron una serie de pasos a los textos y además se realizó la lematización de estos. La secuencia de pasos fueron los siguientes:

- a. Separar el texto en Tokens.
- b. Convertir palabras a minúscula.
- c. Expandir contracciones.
- d. Remover urls, correos, saltos de línea.
- e. Eliminar caracteres repetidos.
- f. Eliminar nuevas líneas y pestañas.
- g. Remover saltos de línea.
- h. Remover comillas simples.
- i. Eliminar comas ",".
- j. Remover números.
- k. Remover Caracteres no alfanuméricos.
- l. Eliminar guiones entre palabras.
- m. Eliminar los guiones dobles y triples.
- n. Eliminar espacios en blanco (al principio, final y espacios dobles).
- o. Eliminar *stop words*.
- p. Realizar *stemmin*/Lematización.
- q. Remover signos de puntuación.
- r. Destokenizar.

La figura 4-8 muestra una función que fue utilizada para procesar los datos.

```
def depurar_datos(data):
    #Remover URLs
    url_pattern = re.compile(r'https?://\S+|www\.\S+')
    data = url_pattern.sub(' ', data)
    #Eliminar caracteres repetidos y limitando toda la repetición a dos caracteres. (english) #I
    Pattern_alpha = re.compile(r'([A-Za-z])\1{1,}', re.DOTALL)
    data = Pattern_alpha.sub(r'\1', data)
    # Remover correos
    data = re.sub('\S*@\S*\s?', '', data)
    #eliminar nuevas líneas y pestañas
    data = data.replace('\n', ' ').replace('\t', ' ').replace('\ ', ' ').replace('. com', '.com')
    # Remover saltos de línea
    data = re.sub('\s+', ' ', data)
    # Remover comillas simples
    data = re.sub("'", "", data)
    #eliminar comas " , "
    data=re.sub(",", "",data)
    # Remover numeros
    data = re.sub(r'\d+', '', data)
    #Remover Caracteres especiales
    data = re.sub(r'^a-zA-Z-á-é-í-ó-ú-ü-Á-É-Í-Ó-Ú-Û-Ñ$', "",data)
    #Convertir palabras a minúscula (Español)
    data=data.lower()
    #eliminar guiones entre palabras ciencia-fiction >>> ciencia fiction
    data=re.sub("-", " ",data)
    #Eliminar los guiones dobles y triples
    data= re.sub("--", " ", data)
    data= re.sub("---", " ", data)
    #Eliminar los espacios en blanco al principio
    data= re.sub(r"^\s+", "", data)
    #Eliminar los espacios en blanco al final
    data= re.sub(r"\s+$", "", data)
    #remover espacios dobles
    data = " ".join(data.split())
    return data
```

Figura 4-8 Función depurar datos

La lematización realizada fue útil para tener cada palabra en su forma base de esta manera las entradas a nuestros modelos son de mayor calidad.

Después de tener los textos procesados, se prepararon las etiquetas que representan la presencia o no de un rasgo de la personalidad, cada notebook se estructuro para que se pudiera elegir el rasgo a reconocer, en la figura 4-9 se muestra el fragmento de código utilizado.

```
#Lo siguiente es para elegir el rasgo a evaluar
#           0           1           2           3           4
rasgos = ['presenta_sociabilidad', 'presenta_apertura', 'presenta_responsabilidad', 'presenta_amabilidad', 'presenta_neuroticismo']
datos_limpios = datos_limpios[datos_limpios.confiableidad_traduccion==1.0]
rasgo = rasgos[3]
print("Rasgo a evaluar: " + rasgo)
datos_limpios = datos_limpios[['texto', rasgo]]
```

Figura 4-9 Elegir el rasgo a evaluar

Por lo tanto, para obtener las etiquetas se utilizó el fragmento de código de la figura 4-10, en el cual se utilizó la biblioteca *TensorFlow* para agregar las etiquetas a en un formato el cual se utilizó para validar y realizar las pruebas en los modelos de reconocimiento automático.

Es importante aclarar que las etiquetas son 0 para indicar que no se presentó el rasgo de la personalidad evaluada y 1 para indicar que si presenta el rasgo.

```

etiquetas = np.array(datos_limpios[rasgo])
y = []
for i in range(len(etiquetas)):
    if etiquetas[i] == 'n' or etiquetas[i] == 'No':
        y.append(0)
    if etiquetas[i] == 'y' or etiquetas[i] == 'Sí':
        y.append(1)
y = np.array(y)
etiquetas = tf.keras.utils.to_categorical(y, 2, dtype="float32")
del y

```

Figura 4-10 Etiquetas del corpus PersonText-myPersonality para las clases de los modelos

Un último paso de procesamiento de los textos en la preparación del *dataset* fue transformar cada texto a secuencias de enteros utilizando un *tokenizer* como se muestra en la figura 16, donde se guarda la secuencia de textos para utilizarse en los modelos de reconocimiento.

```

from keras.models import Sequential
from keras import layers
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.callbacks import ModelCheckpoint

tokenizer = Tokenizer()
tokenizer.fit_on_texts(new_data)
word_index = tokenizer.word_index
total_unique_words = len(tokenizer.word_index) + 1
sequences = tokenizer.texts_to_sequences(new_data)
max_seq_length = max([len(x) for x in sequences])
print(sequences)
print(max_seq_length)

textos = pad_sequences(sequences, maxlen=max_seq_length)

print(textos[100:110])
max_words = total_unique_words
max_len = max_seq_length

print(textos[0])

```

Figura 4-11 Código Python para obtener la secuencia de enteros de los textos

Al final, se realizó la división del *dataset* en un 70% para el entrenamiento y el 30% para las pruebas utilizando la biblioteca llamada *sklearn*.

Con el *dataset* que contenía el 70% de entrenamiento se dividió nuevamente en un 70% para el entrenamiento de los modelos y un 30% para la validación, de esa manera se obtuvieron los datos necesarios para crear los modelos de reconocimiento.

4.2.2. Construcción de modelos.

Se construyeron cuatro modelos de reconocimiento utilizando redes neuronales, estas cuentan con una capa de entrada, una serie de capas ocultas que procesan la información de entrada y la última capa es la de salida, la cual nos muestra el resultado del modelo. Este

proceso creación de modelos de redes neuronales se realizó para cada rasgo, en 10, 20 y 50 épocas, utilizando diferentes notebooks en las que se realizó un proceso similar, pero especificando el rasgo a predecir y la cantidad de épocas a entrenar.

4.2.2.1. Perceptrón Multicapa

El primer modelo fue el Perceptrón multicapa, es un modelo flexible y ajustable a una gran variedad de tareas, en la figura 4-12 se muestra el código utilizado para su estructura, así como algunas capas y la función de activación.

```
model = Sequential()
model.add(layers.Embedding(max_words, 128, input_length=max_len))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(2, activation='softmax')) |
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()

earlyStopping = keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=5, verbose=1, mode='min')
mcp = ModelCheckpoint(nombrePerceptron, monitor="val_accuracy", save_best_only=True, save_weights_only=False, verbose=1)
model_history = model.fit(X_train, y_train, verbose=1,
                          validation_data = (X_test, y_test), epochs=epocas,
                          callbacks=[earlyStopping,mcp], batch_size= 64, shuffle=True)
```

Figura 4-12 Código Python del modelo Perceptrón Multicapa

Primero, se crea un modelo secuencial vacío, que es la estructura básica para construir una red neuronal en *Keras*. Este modelo se compone de una secuencia lineal de capas.

Se agregó una capa de incrustación (*embedding layer*) al modelo. Esta capa se utiliza para convertir secuencias de palabras en vectores densos de longitud fija. `max_words` representa el tamaño del vocabulario utilizado, 128 es la dimensión de los vectores de incrustación y `input_length` es la longitud máxima de las secuencias de entrada.

Se agregó una capa de aplanamiento (*flatten layer*) al modelo. Esta capa se utiliza para aplanar los datos de entrada en un vector unidimensional, que luego se pasa como entrada a las capas completamente conectadas.

Se agregó una capa densa (*fully connected layer*) con 512 unidades y función de activación ReLU. Esta capa está diseñada para realizar transformaciones lineales en los datos de entrada.

Se agregó otra capa densa con 2 unidades y función de activación *softmax*. Esta capa se utiliza para obtener las probabilidades de pertenencia a cada una de las dos clases en el problema de clasificación. Si se desea realizar una clasificación en tres clases, se puede cambiar el número de unidades a 3 y ajustar la función de activación en consecuencia.

Se compiló el modelo especificando la función de pérdida, el optimizador y las métricas a utilizar durante el entrenamiento. En este caso, se utilizó la función de pérdida 'categorical_crossentropy', el optimizador 'adam' y la métrica de precisión ('accuracy').

Se definió el *callback earlyStopping* para detener el entrenamiento en caso de que no haya una mejora en la exactitud de la validación y se guarda un *checkpoint* con el modelo que obtuvo mejores resultados.

Al final se realizó el entrenamiento con la función *'fit'*, donde se especificaron los datos de entrenamiento y validación y el número de épocas.

4.2.2.2. LSTM Sencilla

Esta arquitectura de red neuronal fue construida como se muestra en la figura 4-13.

LSTM Sencilla

+ Código

+ Markdown

```
model1 = Sequential()
model1.add(layers.Embedding(max_words, 20))
model1.add(layers.LSTM(15, dropout=0.5))
model1.add(layers.Dense(2, activation='softmax'))

model1.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
#Implementing model checkpoints to save the best metric and do not lose it on training.
checkpoint1 = ModelCheckpoint(nombreLSTMSencilla, monitor='val_accuracy', verbose=1, save_best_only=True, mode='auto', save_freq='epoch', save_weights_only=False)
history = model1.fit(X_train, y_train, epochs=epocas, validation_data=(X_test, y_test), callbacks=[checkpoint1])
```

Figura 4-13 Código Python del modelo LSTM Sencilla

Se creó un modelo secuencial vacío para después agregar cada capa.

Posteriormente se agregó una capa de incrustación (*embedding layer*) al modelo. Esta capa se utiliza para convertir secuencias de palabras en vectores densos de longitud fija. *max_words* representa el tamaño del vocabulario utilizado, y 20 es la dimensión de los vectores de incrustación.

Se agregó una capa LSTM (*Long Short-Term Memory*) al modelo. LSTM es un tipo de unidad recurrente que permite a la red aprender a recordar información a largo plazo. 15 representa el número de neuronas en la capa LSTM, y *dropout=0.5* especifica que se aplicará un *dropout* del 50% para evitar el sobreajuste.

Se agregó también una capa densa (*fully connected layer*) con 2 unidades y función de activación *softmax*. Esta capa se utiliza para obtener las probabilidades de pertenencia a cada una de las dos clases en el problema de clasificación.

Se compiló el modelo especificando el optimizador, la función de pérdida y las métricas a utilizar durante el entrenamiento. Se agregó el optimizador 'rmsprop', la función de pérdida 'categorical_crossentropy' y se registra la métrica de exactitud ('accuracy').

Se definió un *callback* para guardar el modelo con los mejores pesos durante el entrenamiento. El modelo se guarda en un archivo nombrado cuando se observa una mejora en la métrica de precisión de validación.

Se entrenó el modelo utilizando el método *fit* con los datos de entrenamiento (X_train y y_train), los datos de validación (X_test y y_test), el número de épocas y el *callback* definido.

4.2.2.3. LSTM Bidireccional

Se construyó un modelo LSTM Bidireccional con la estructura que se muestra en el código de la figura 4-14.

LSTM Bidireccional

```
model2 = Sequential()
model2.add(layers.Embedding(max_words, 40, input_length=max_len))
model2.add(layers.Bidirectional(layers.LSTM(20, dropout=0.6)))
model2.add(layers.Dense(2, activation='softmax'))
model2.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

checkpoint2 = ModelCheckpoint('lstm2bi.h5', monitor='val_accuracy', verbose=1, save_best_only=True, mode='auto', period=1, save_weights_only=False)
history = model2.fit(X_train, y_train, epochs=epocas, validation_data=(X_test, y_test), callbacks=[checkpoint2])
```

Python

Figura 4-14 Código Python del modelo LSTM Bidireccional

De la misma manera que los anteriores modelos, se creó un modelo secuencial vacío.

Se agregó una capa de incrustación (*embedding layer*) al modelo. Esta capa se utiliza para convertir secuencias de palabras en vectores densos de longitud fija. *max_words* representa el tamaño del vocabulario utilizado, 40 es la dimensión de los vectores de incrustación y *input_length* es la longitud máxima de las secuencias de entrada.

Se agregó una capa bidireccional LSTM (*Long Short-Term Memory*) al modelo. Una capa LSTM bidireccional procesa la secuencia de entrada hacia adelante y hacia atrás y combina las representaciones aprendidas en ambas direcciones. El número 20 representa el número de neuronas en la capa LSTM, y *dropout*=0.6 especifica que se aplicará un *dropout* del 60% para evitar el sobreajuste.

Se agregó una capa densa (*fully connected layer*) con 2 unidades y función de activación *softmax*. Esta capa se utiliza para obtener las probabilidades de pertenencia a cada una de las dos clases en el problema de clasificación.

Se compiló el modelo especificando el optimizador, la función de pérdida y las métricas a utilizar durante el entrenamiento. En este modelo también se utilizó el optimizador 'rmsprop', la función de pérdida '*categorical_crossentropy*' y se registra la métrica de exactitud ('*accuracy*').

Se definió un *callback* para guardar el modelo con los mejores pesos durante el entrenamiento. El modelo se guarda en el archivo nombreLSTMBi cuando se observa una mejora en la métrica de precisión de validación.

Se entrenó el modelo utilizando el método *fit* con los datos de entrenamiento (X_train y y_train), los datos de validación (X_test y y_test), el número de épocas y el *callback* definido.

4.2.2.4. CNN1D

El último modelo con redes neuronales que se construyó fue un CNN1D el cual se contiene la estructura mostrada en el código de la figura 4-15.



```
from keras import regularizers
model3 = Sequential()
model3.add(layers.Embedding(max_words, 40, input_length=max_len))
model3.add(layers.Conv1D(20, 6, activation='relu', kernel_regularizer=regularizers.l1_l2(l1=2e-3, l2=2e-3), bias_regularizer=regularizers.l2(2e-3)))
model3.add(layers.MaxPooling1D(5))
model3.add(layers.Conv1D(20, 6, activation='relu', kernel_regularizer=regularizers.l1_l2(l1=2e-3, l2=2e-3), bias_regularizer=regularizers.l2(2e-3)))
model3.add(layers.GlobalMaxPooling1D())
model3.add(layers.Dense(2, activation='softmax'))
model3.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['acc'])
metric = 'val_acc'

checkpoint3 = ModelCheckpoint(nombreCNN1D, monitor=metric, verbose=2, save_best_only=True, mode='auto', save_weights_only=False)

history = model3.fit(X_train, y_train, epochs=epocas, validation_data=(X_test, y_test), callbacks=[checkpoint3])
```

Figura 4-15 Código Python del modelo CNN1D

Se creó un modelo secuencial vacío para agregar posteriormente las demás capas para la red convolucional.

Se agregó una capa de incrustación al modelo al igual que los demás modelos.

Después, se agregó una capa convolucional 1D al modelo, esta es la principal diferencia entre los otros modelos. Esta capa utiliza filtros convolucionales de tamaño 6 y 20 canales. Se aplica la función de activación ReLU a la salida de esta capa. Además, se aplica

regularización L1 y L2 a los pesos y sesgos de la capa respectivamente, con coeficientes de regularización especificados.

Se agregó una capa de Max Pooling 1D al modelo. Esta capa reduce la dimensionalidad de las características obtenidas de la capa convolucional, tomando el máximo valor dentro de ventanas de tamaño 5.

Se agregó otra capa convolucional 1D al modelo con las mismas configuraciones que la capa anterior.

Se agregó una capa de Max Pooling global 1D al modelo. Esta capa toma el máximo valor a lo largo de cada canal de características.

Se agregó una capa densa con 2 unidades y función de activación *softmax*. Esta capa se utiliza para obtener las probabilidades de pertenencia a cada una de las dos clases en el problema de clasificación.

Se compiló el modelo especificando el optimizador, la función de pérdida y las métricas a utilizar durante el entrenamiento. En este modelo también se utilizó el optimizador 'rmsprop', la función de pérdida '*categorical_crossentropy*' y se registra la métrica de exactitud ('acc').

Se define un *callback* para guardar el modelo con los mejores pesos durante el entrenamiento. El modelo se guarda en el archivo nombreCNN1D cuando se observa una mejora en la métrica.

Al final, se ejecuta la instrucción para entrenar el modelo con los datos de entrenamiento, validación, las épocas y el *checkpoint* para usar el mejor modelo.

4.2.2.5. *BERT Multilingual*

El modelo *BERT Multilingual* es un modelo pre entrenado que utiliza la biblioteca Transformers de Google, este modelo es especial para trabajar con el procesamiento del lenguaje natural y se puede ajustar a una tarea específica, por lo tanto, con la intención de obtener buenos resultados al predecir un rasgo de la personalidad se utilizó el corpus de *MyPersonality* en español para entrenar este modelo llamado BERT, el cual se entrenó con 5 épocas para predecir el rasgo de “Apertura”.

El proceso realizado fue el siguiente:

1. Se realizó la importación las bibliotecas necesarias.
2. Se cargó el corpus desde Drive.
3. Se definió una clase para hacer compatible el conjunto de datos con el flujo de trabajo de entrenamiento de *PyTorch*.
4. Se creó un *dataLoader* que permite cargar los datos en lotes durante el entrenamiento.
5. Se realizó la división del *dataset* para entrenamiento con un 80% y 20% para las pruebas.
6. Se definió un modelo de clasificación de rasgos que utiliza BERT como base:

El modelo se inicializó con una instancia de BERT pre entrenada y se adapta con capas adicionales para la clasificación de múltiples clases. Durante la propagación hacia adelante, se extrae la representación del token [CLS] de BERT, se aplica *Dropout* y se realiza una transformación lineal para obtener las predicciones como se muestra en la figura 21.

```
class BERTTraitClassifier(nn.Module):

    def __init__(self, n_classes):
        super(BERTTraitClassifier, self).__init__()
        self.bert = BertModel.from_pretrained(PRE_TRAINED_MODEL_NAME, return_dict=False)
        self.drop = nn.Dropout(p=0.3)
        self.linear = nn.Linear(self.bert.config.hidden_size, n_classes)

    def forward(self, input_ids, attention_mask):
        _, cls_output = self.bert(
            input_ids = input_ids,
            attention_mask = attention_mask
        )
        drop_output = self.drop(cls_output)
        output = self.linear(drop_output)
        return output
```

Figura 4-16 Clase para clasificar rasgos de la personalidad con el modelo BERT

7. Se instanció el modelo clasificador con las clases.
8. Se definió una función para entrenar el modelo en un conjunto de datos y realizar una iteración sobre los lotes de datos.
9. Se declara una función para evaluar el modelo en un conjunto de datos de prueba.
10. Se realizó el entrenamiento y la evaluación del modelo en cada época con el código que se muestra en la figura 4-17.

```

for epoch in range(EPOCHS):
    print('Epoch {} de {}'.format(epoch+1, EPOCHS))
    print('-----')
    train_acc, train_loss = train_model(
        model, train_data_loader, loss_fn, optimizer, device, scheduler, len(df_train)
    )
    test_acc, test_loss = eval_model(
        model, test_data_loader, loss_fn, device, len(df_test)
    )
    print('Entrenamiento: Loss: {}, accuracy: {}'.format(train_loss, train_acc))
    print('Validación: Loss: {}, accuracy: {}'.format(test_loss, test_acc))
    print('')

Epoch 1 de 5
-----
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2364:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2364:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2364:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2364:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2364:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2364:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2364:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2364:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2364:
  warnings.warn(
Entrenamiento: Loss: 0.5992536990081563, accuracy: 0.7250554323725056
Validación: Loss: 0.6247685009782965, accuracy: 0.7079646017699115

```

Figura 4-17 Entrenamiento del modelo con BERT en 5 épocas

Por último, se declaró una función *classifyTrait* que toma un texto como entrada y utiliza el modelo entrenado para realizar una clasificación de rasgo de la personalidad en función de un texto proporcionado, de esta manera se pueden realizar predicciones con diferentes textos.

4.2.3. Evaluación de resultados obtenidos.

Después de realizar el entrenamiento de cada modelo, se realizó el *testing* de cada uno de ellos con cada rasgo respectivamente con las diferentes épocas, por lo tanto, fue necesario realizar una recolección de resultados de manera eficiente.

4.2.3.1. Recolección de resultados.

La decisión tomada fue crear un data set para guardar los resultados de cada rasgo, modelo en su respectiva cantidad de épocas de entrenamiento.

El *dataset* creado tiene la estructura que se muestra en la figura 4-18.

	rasgo		modelo	epocas	accuracy	f1	precision	recall
0	presenta_sociabilidad		perceptron_10_presenta_sociabilidad.hdf5	10	0.559055	0.358586	0.279528	0.500000
1	presenta_sociabilidad		lstmSencilla_10_presenta_sociabilidad.hdf5	10	0.559055	0.358586	0.279528	0.500000

Figura 4-18 Estructura del archivo csv para resultados

La estructura de este *dataset* fue utilizada en diferentes archivos de formato csv para tener los resultados de cada sección de modelos con la misma cantidad de épocas.

Los modelos que fueron entrenados se guardan automáticamente, por lo tanto, se utilizaron para realizar las pruebas con los conjuntos de datos para pruebas, como se mencionó anteriormente en su respectiva cantidad de épocas, la figura 4-19 muestra el flujo de instrucciones para lograrlo en el caso del modelo Perceptrón Multicapa.

```
#Cargamos el mejor modelo almacenado
best_model_mlp = keras.models.load_model(nombrePerceptron)
#Calculamos las predicciones
predicts_perceptron = best_model_mlp.predict(textos_prueba)
predicts_mlp = np.around(predicts_perceptron, decimals=0)
#Realizamos la evaluación y obtenemos las métricas
mlp_recall = metrics.recall_score(etiquetas_prueba, predicts_mlp, average='macro')
mlp_f1 = metrics.f1_score(etiquetas_prueba, predicts_mlp, average='macro')
mlp_precision = metrics.precision_score(etiquetas_prueba, predicts_mlp, average='macro')
mlp_accuracy = metrics.accuracy_score(etiquetas_prueba, predicts_mlp)
#Imprimimos los valores de las métricas
print(f"Rasgo evaluado: {rasgo}")
print(f"MLP Accuracy: {mlp_accuracy}")
print(f"MLP F1: {mlp_f1}")
print(f"MLP Precision: {mlp_precision}")
print(f"MLP Recall: {mlp_recall}")

20/20 [=====] - 0s 4ms/step
Rasgo evaluado: presenta_amabilidad
MLP Accuracy: 0.5291338582677165
MLP F1: 0.5272580143168378
MLP Precision: 0.5280822329532932
MLP Recall: 0.5278257589294577
```

Figura 4-19 Obtener métricas del Perceptrón multicapa

Para almacenar en su respectivo archivo csv se cargó mediante la biblioteca Pandas y se registraron los resultados de cada modelo con el código que se muestra en la figura 4-20.

```
resultados = resultados.append({'rasgo': rasgo, 'modelo': nombrePerceptron, 'epocas': epocas, 'accuracy': mlp_accuracy,
                                'f1': mlp_f1, 'precision': mlp_precision, 'recall': mlp_recall}, ignore_index=True)
resultados
```

Figura 4-20 Guardar los resultados del modelo en el csv de resultados

De esta manera, sistemáticamente se realizó la recolección de resultados de cada modelo. Esta información fue necesaria para las siguientes tareas realizadas.

4.2.4. Graficación de los resultados

Para mostrar los resultados se consideró utilizar herramientas de *graficación* como Excel u otro software, sin embargo, se optó por graficar utilizando bibliotecas en el lenguaje Python, porque es más práctico obtener los resultados del archivo csv que los contiene.

La biblioteca que fue utilizada es llamada “*matplotlib*” y nos permite generar graficas de manera práctica.

La figura 4-21 muestra cómo se realizó la carga del archivo que contiene los resultados, en este ejemplo se cargaron los resultados de los modelos que fueron entrenados con 50 épocas.

```
resultados = pd.read_csv("./50 epocas/resultados.csv",sep=";", encoding='UTF-8')
accuracy = resultados['accuracy'].tolist()
```

Figura 4-21 Cargar archivo de resultados

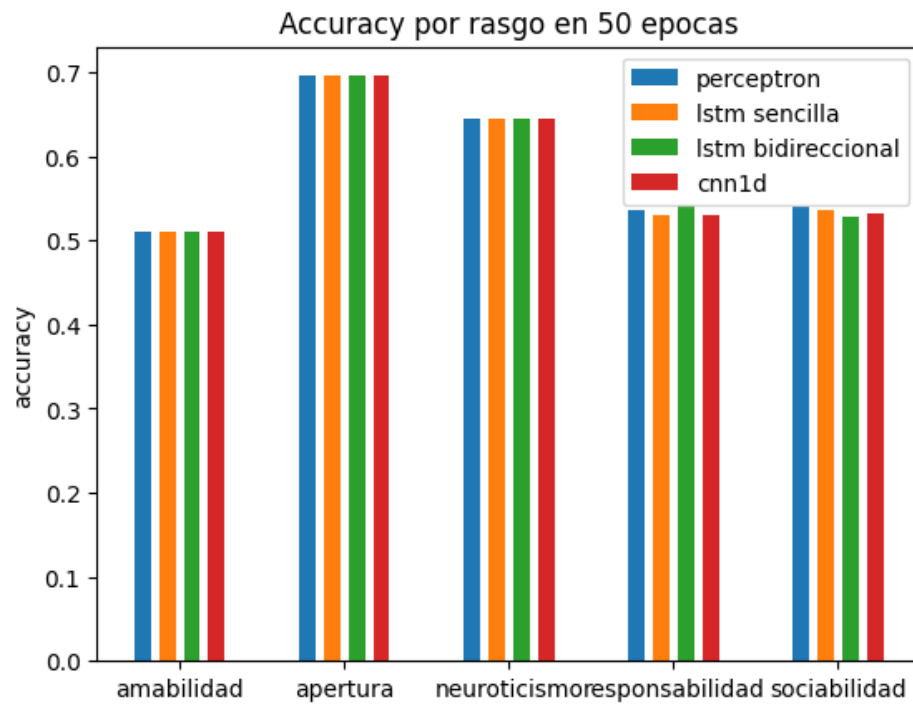
La biblioteca necesita los datos para generar la gráfica, en la figura 4-22 se muestra cómo se especificaron los rasgos a graficar, así como las métricas, después, se agregaron las listas que contienen los resultados a graficar.

```
# Datos
rasgos50 = ['amabilidad','apertura','neuroticismo','responsabilidad','sociabilidad']
metricas50 = ['perceptron', 'lstm sencilla', 'lstm bidireccional', 'cnn1d']
valores50 = np.array([
    lista1,
    lista2,
    lista3,
    lista4,
    lista5
])
```

Figura 4-22 Valores de los resultados para graficar

El valor principal que fue de interés a analizar fue la exactitud (*accuracy*), por lo tanto, las gráficas solo muestran esta métrica, esto permitió observar y contrastar el comportamiento de los modelos al predecir los rasgos de la personalidad.

La gráfica 1 muestra el resultado de graficar la exactitud de los cuatro modelos de redes neuronales con 50 épocas en cada uno de los rasgos de la personalidad del modelo Big Five.



Gráfica 4-1 Resultados de los modelos con 50 épocas

Capítulo 5

5. Pruebas y Resultados

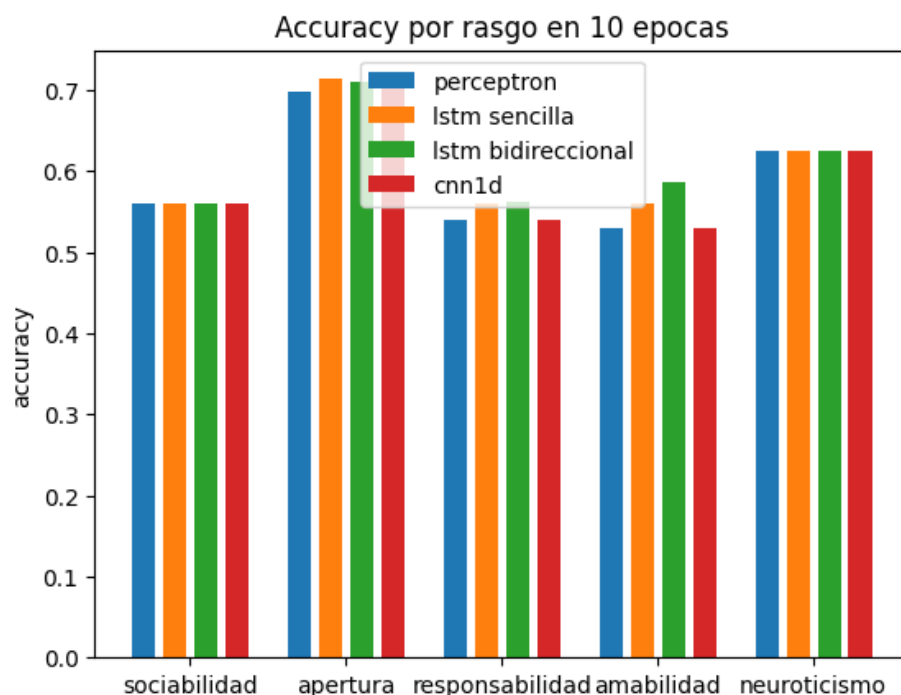
5.1. Resultados de los modelos

En este capítulo se muestran los resultados obtenidos del desarrollo de este proyecto, enfocados principalmente a la exactitud obtenida al predecir cada uno de los rasgos de la personalidad del Modelo Big Five con cada uno de los modelos entrenados.

5.1.1. Resultados *MyPersonality* en español

Para observar y analizar correctamente los resultados obtenidos de la nueva versión del corpus *PersonText*, se optó por evaluar, antes que nada, el corpus *MyPersonality* en español con los diferentes rasgos, modelos y épocas. Esto tiene como utilidad principalmente el poder comparar resultados de ambos corpus y llegar a una conclusión concreta.

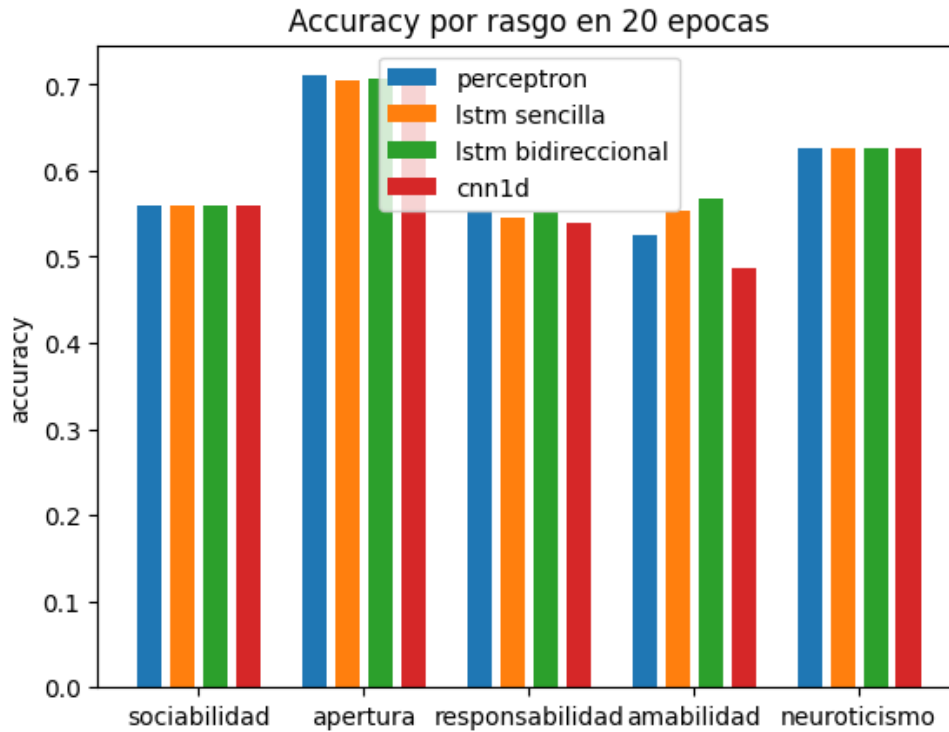
A continuación, en la gráfica 5-1 se observan los resultados de predecir los cinco rasgos del modelo Big Five con diferentes modelos entrenados con 10 épocas.



Gráfica 5-1 Accuracy por rasgo en 10 épocas MyPersonality en español

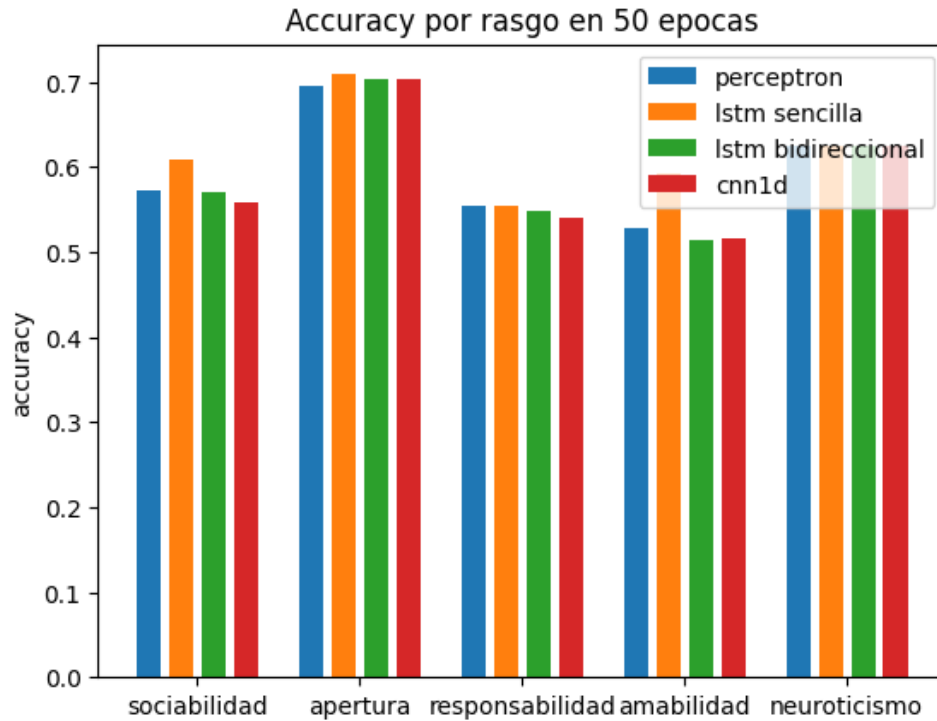
Al analizar la información con la que se generó la gráfica 3, se observó que rasgo que el rasgo que mejor se predijo fue el de apertura acercándose los cuatro modelos al 0.7 de exactitud.

La gráfica 3 muestra los resultados obtenidos de los modelos durante las pruebas, se puede observar una similitud en el rasgo que mejor se predijo, siendo el de Apertura con una exactitud de 0.7 aproximadamente, seguido del Neuroticismo, el cual se obtuvo un poco más de 0.6 de exactitud.



Gráfica 5-2 Accuracy por rasgo 20 épocas MyPersonality español

Por último, la gráfica 4 ilustra los resultados obtenidos de las pruebas de los modelos con 50 épocas, con esta cantidad de épocas se buscó obtener mejores resultados con modelos mejor entrenados. Al analizar la gráfica se observó que existía una tendencia o similitud en los resultados de los modelos anteriores, los cuales predijeron el rasgo de la personalidad de Apertura con un valor cercano a 0.7 de exactitud al igual que los modelos entrenados con 50 épocas. Cabe resaltar que los resultados al predecir los rasgos de Amabilidad, Responsabilidad y Sociabilidad no superaron el valor de 0.6 de exactitud.

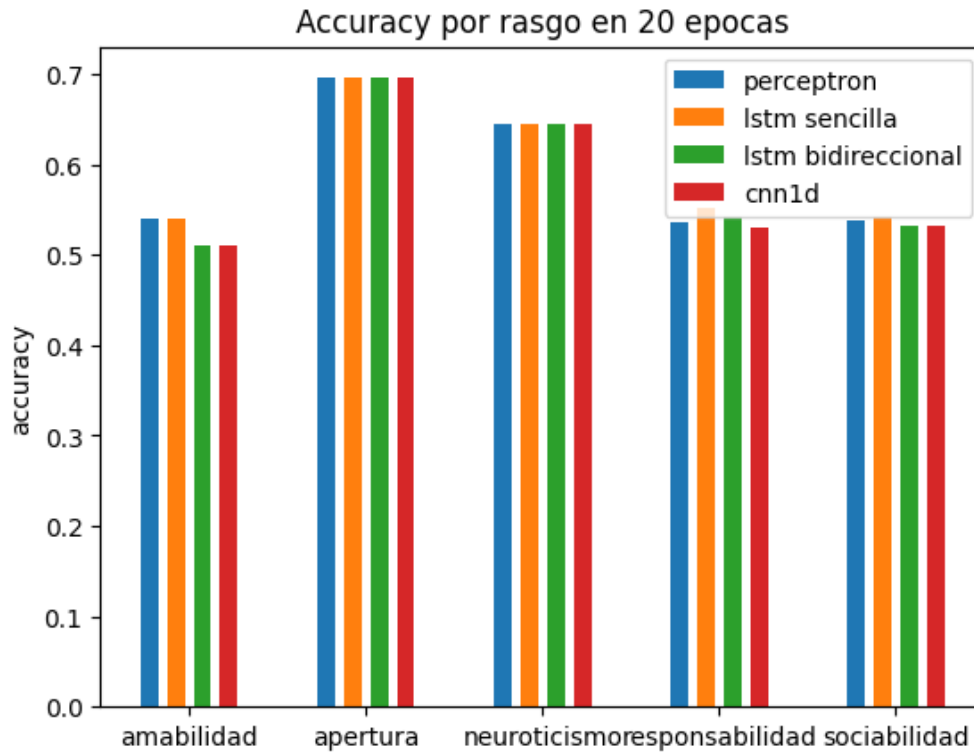


Gráfica 5-3 Accuracy por rasgo 50 épocas

5.1.2. Resultados PersonText-myPersonality

Después de realizar el incremento del corpus *PersonText* y obtener una nueva versión llamada *PersonText-myPersonality* que cuenta con 2329 se realizaron las pruebas con nuevos modelos de redes neuronales, que fueron entrenados con 20 y 50 épocas, los resultados fueron graficados con el lenguaje de programación Python.

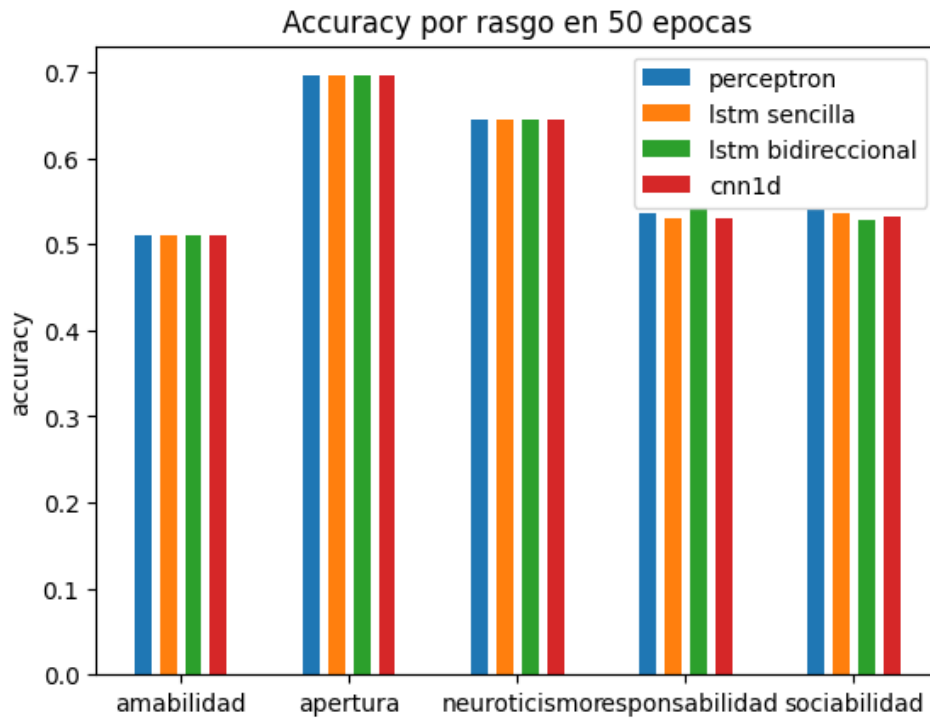
A continuación, se observa la gráfica 5, la cual contiene la gráfica los resultados que se obtuvieron de los modelos al predecir cada uno de los rasgos del modelo Big Five con 20 épocas, dicha figura representa la exactitud obtenida en cada modelo.



Gráfica 5-4 Accuracy PersonText-myPersonality 20 épocas

Se observó que en los resultados que el rasgo con mejor exactitud en todos los modelos fue el de Apertura con valor cercano a 0.7, seguido del neuroticismo en un rango de 0.6 a 0.7 y el rasgo con peores resultados en cuanto a exactitud fue el de amabilidad, donde este rasgo tiene un valor cercano al 0.5.

Para una mayor variedad en los resultados, se realizaron graficas en Python también de los resultados obtenidos de los modelos entrenados con 50 épocas, en la gráfica 6 se observan los resultados por cada rasgo y cada modelo.



Gráfica 5-5 Accuracy corpus PersonText-myPersonality 50 épocas

La tabla 1 presenta un resumen de los mejores resultados obtenidos de los diferentes modelos entrenados, de esta manera podemos llegar a una mejor conclusión en este proyecto. Es importante aclarar, que dicha tabla fue creada para cada rasgo de la personalidad del modelo Big Five, para cada modelo y para cada cantidad de épocas.

Tabla 1 Mejores resultados

Modelo	Épocas	Rasgo	Exactitud
Perceptrón	20	Amabilidad	0.539341917024320
Perceptrón	20	Apertura	0.695278969957081
Perceptrón	20	Neuroticismo	0.645207439198855
LSTM Sencilla	20	Responsabilidad	0.550786838340486
LSTM Sencilla	20	Sociabilidad	0.542203147353362

Capítulo 6

6. Conclusiones y trabajo futuro

En este capítulo se describen las conclusiones del proyecto, recomendaciones para futuros trabajos de investigación y la experiencia personal adquirida durante el desarrollo del proyecto.

6.1. Conclusiones del proyecto.

Para concluir, los modelos de personalidad nos permiten representar la personalidad de las personas mediante rasgos o dimensiones, de esta manera se puede aplicar un modelo de personalidad en diferentes disciplinas, el modelo Big Five utilizado resultó útil para construir modelos de reconocimiento de la personalidad donde se predice cada uno de los rasgos del modelo con la finalidad de aplicarse en diversas aplicaciones, por ejemplo, la educación.

Estos modelos fueron contruidos principalmente con redes neuronales, en diferentes parámetros de entrenamiento. La fuente de información para el entrenamiento de los modelos utilizados fueron los corpus de textos etiquetados, estos corpus fueron *MyPersonality* en español, *PersonText* y la combinación de ambos.

Para un corpus es muy importante la variedad, calidad y cantidad de los textos que lo conforman, por lo tanto, se realizó la combinación del corpus *PersonText* y *MyPersonality* y se obtuvo una nueva versión con una cantidad de 2329 textos. Los modelos de reconocimiento fueron entrenados y probados con estos corpus en un porcentaje de 70% para entrenamiento y validación y un 30% se utilizó para pruebas.

Las pruebas realizadas fueron con la intención de identificar la versión de los modelos que mejor predicen los rasgos de la personalidad.

Con el análisis de los resultados se obtuvo una tabla que resume y muestra los mejores modelos de reconocimiento para cada uno de los rasgos del modelo Big Five, así como las

épocas en las que fue entrenado y la exactitud que obtuvo tal modelo, dicha tabla se puede observar en la figura 32.

El mejor modelo para predecir el rasgo de Amabilidad fue el Perceptrón Multicapa, fue entrenado con 20 épocas y obtuvo una exactitud de 0.5393.

El mejor modelo para predecir el rasgo de Apertura de igual manera el Perceptrón Multicapa, con 20 épocas y con una exactitud obtenida de 0.6952.

En el rasgo de Neuroticismo se obtuvo una exactitud de 0.6452 con el modelo Perceptrón multicapa entrenado con 20 épocas.

El modelo LSTM Sencilla entrenado con 20 épocas obtuvo una exactitud de 0.5507 al predecir el rasgo de Responsabilidad.

El modelo que mejor predijo el rasgo de Sociabilidad fue el de LSTM Sencilla, entrenado con 20 épocas obtuvo una exactitud de 0.5422.

Al analizar estos resultados se concluyó que el número de épocas no es lo más importante al momento de escoger los parámetros de entrenamiento, esto debido a que los modelos también fueron entrenados con 50 épocas y aun así los mejores resultados fueron de modelos con 20 épocas. Esto significa que para construir modelos con buenos resultados de exactitud es necesario enfocarse en la calidad del *dataset* y corpus utilizados.

Además, predecir la personalidad es posible si se cuenta con un buen corpus con textos de calidad, sin embargo, se debe considerar que los modelos de reconocimiento y los corpus pueden obtener diferentes resultados al predecir cada uno de los rasgos, es decir, un modelo entrenado con cierto corpus puede servir para predecir un rasgo de la personalidad pero talvez no para otro, por lo tanto, sería buen tema de investigación construir diferentes corpus con diferentes modelos para cada uno de los rasgos de la personalidad.

6.2. Recomendaciones

- Crear modelos con parámetros específicos que obtengan valores altos de exactitud para rasgos específicos.

- Incrementar el corpus *PersonText* con datos obtenidos de otros corpus con técnicas de probabilidad y estadística.
- Crear diferentes versiones de corpus para predecir rasgos específicos de la personalidad.

7. Bibliografía

- Alpaydin, E. (2016). Machine learning. *MIT Press*.
- Arias, M. (2016). Diseño e implementación de un algoritmo de clasificación de objetos peligrosos en radiografías utilizando el modelo bag of words. (*Tesis de Magister en Ciencias de la Ingeniería*). Pontificia Universidad Católica de Chile, Santiago, Chile.
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- Bodero, E. M. (2020). Google Colaboratory como alternativa para el procesamiento de una red neuronal convolucional. *Revista Espacios*, 41(07).
- Bolívar, A. (2013). La definición del corpus en los estudios del discurso. . *Revista Latinoamericana de Estudios del Discurso*, 13(1), 1-5. .
- Bonet Cruz, I. S. (2007). Redes neuronales recurrentes para el análisis de secuencias. *Revista Cubana de Ciencias Informáticas*, 1(4), 48-57.
- Challenger-Pérez, I. D.-R.-G. (2014). El lenguaje de programación Python. *Ciencias Holguín*, 1-13.
- Cios, K. J. (2014). Data mining and data sets: Concept and terminology. In Handbook of data mining and data sets. *Springer, Cham*, 1-18.
- Eysenck, H. J. (1967). *The biological basis of personality*. Springfield: IL: Charles C Thomas.
- Eysenck, H. J. (1994). *Tabaco, personalidad y estrés*. Barcelona: Herder.
- Fernández, G. (2020). *Clasificación taxonómica de bacterias usando machine learning*. Uniandes.
- Fernández, J. J. (2021). Estimación adelantada del crecimiento regional mediante redes. *Investigaciones Regionales*, 45-64.
- Goldberg, L. R. (1990). An alternative "description of personality": The Big-Five factor structure. *Journal of Personality and Social Psychology*, 59(6), 1216-1229.
- Hernández, J. V. (2016). *Identificación de rasgos de personalidad a través del Análisis*. Hugging Face. (8 de 5 de 2023). Obtenido de GPT-2: <https://huggingface.co/gpt2>

- John, L. A. (2008). *Handbook of Personality: Theory and Research*. New York, NY: The Guilford Press.
- Manning, C. D. (2009). Introduction to information retrieval. *Cambridge University Press*.
- McCrae, R. R. (1999). Reinterpreting the five-factor model. En *Handbook of personality: Theory and research* (págs. 139-153). New York, NY: Guilford Press.
- Mitchell, T. M. (1997). *Machine learning (2nd ed.)*. McGraw-Hill.
- Montanés, R. D. (2018). Aplicación de un modelo híbrido de aprendizaje profundo para el Análisis de Sentimiento en Twitter. In *CEUR workshop proc*.
- Pandas. (2023). *Pandas*. Obtenido de pandas - Python Data Analysis Library: <https://pandas.pydata.org/>
- Roldan, J. (2022). Creación de un corpus y clasificación de textos en español para reconocimiento de personalidad. (*Tesis de Maestría en Ciencias de la Computación*). TecNM-Instituto Tecnológico de Culiacán, Culiacán.
- Stillwell, D. J. (2015). *myPersonality Project*. Obtenido de myPersonality website: <https://sites.google.com/michalkosinski.com/mypersonality>
- TensorFlow. (2023). *TensorFlow*. Obtenido de TensorFlow: <https://www.tensorflow.org/>
- Toro Bayona, G. A., & Alberto, L. S. (2012). Evaluación de las Redes Neuronales Artificiales Perceptron Multicapa y Fuzzy-Artmap en. *Ingeniería*, vol. 17, núm. 1, 2012, pp. 61-72.
- Zuckerman, M. (1994). An alternative five-factor model of personality. *Psychological Bulletin*, 115, 270-291.