INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Detección automática de paráfrasis en preguntas en entornos colaborativos

TESIS

QUE PARA OBTENER EL TÍTULO DE

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

Ing. Tania Gisela Alcantara Medina

Director de tesis:

Dr. Francisco Hiram Calvo Castro

Ciudad de México, Noviembre 2022

SIP-13
REP 2017

INSTITUTO POLITÉCNICO NACIONAL
SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REGISTRO DE TEMA DE TESIS
Y DESIGNACIÓN DE DIRECTOR DE TESIS

Ciudad de México, a 03 de mayo del 2022

El Colegio de Profesores de Posgrado del **Centro de Investigación en Computación** en su Sesión
(Unidad Académica)

Extraordinaria No 06 celebrada el día 28 del mes marzo de 2022, conoció la solicitud presentada por el (la) alumno (a):

Apellido Paterno:	ALCÁNTARA	Apellido Materno:	MEDINA	Nombre (s):	TANIA GISELA

Número de registro: B 2 0 0 4 1 9

del Programa Académico de Posgrado: **Maestría en Ciencias de la Computación**

Referente al registro de su tema de tesis; acordando lo siguiente:

1.- Se designa al aspirante el tema de tesis titulado:

"Detección automática de paráfrasis en preguntas en entornos colaborativos"

Objetivo general del trabajo de tesis:

Detectar paráfrasis de manera automática en pares de preguntas utilizando representaciones de codificación bidireccional de Transformers.

2.- Se designa como Directores de Tesis a los profesores:

Director: **Dr. Francisco Hiram Calvo Castro** 2° Director:

No aplica: x

3.- El Trabajo de investigación base para el desarrollo de la tesis será elaborado por el alumno en:

Centro de Investigación en Computación

que cuenta con los recursos e infraestructura necesarios.

4.- El interesado deberá asistir a los seminarios desarrollados en el área de adscripción del trabajo desde la fecha en que se suscribe la presente, hasta la aprobación de la versión completa de la tesis por parte de la Comisión Revisora correspondiente.

Director(a) de Tesis

Dr. Francisco Hiram Calvo Castro

Aspirante

C. Tania Gisela Alcántara Medina

2º Director de Tesis

Presidente del Colegio

Dr. Francisco Hiram Calvo Castro

SIP-14
REP 2017

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 10:00 horas del día 14 del mes de noviembre del 2022 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Posgrado de: Centro de Investigación en Computación para examinar la tesis titulada:

"Detección automática de paráfrasis en preguntas en entornos colaborativos" del (la) alumno (a):

Apellido Paterno:	ALCANTARA	Apellido Materno:	MEDINA	Nombre (s):	TANIA GISELA

Número de registro: B 2 0 0 4 1 9

Aspirante del Programa Académico de Posgrado: **Maestría en Ciencias de la Computación**

Una vez que se realizó un análisis de similitud de texto, utilizando el software antiplagio, se encontró que el trabajo de tesis tiene 12 % de similitud. **Se adjunta reporte de software utilizado**.

Después que esta Comisión revisó exhaustivamente el contenido, estructura, intención y ubicación de los textos de la tesis identificados como coincidentes con otros documentos, concluyó que en el presente trabajo **SI** ☐ **NO** ☒ **SE CONSTITUYE UN POSIBLE PLAGIO.**

JUSTIFICACIÓN DE LA CONCLUSIÓN: *(Por ejemplo, el % de similitud se localiza en metodologías adecuadamente referidas a fuente original)*
Las coincidencias son de fragmentos de texto pequeños y corresponden a una publicación previa derivada de este trabajo.

****Es responsabilidad del alumno como autor de la tesis la verificación antiplagio, y del Director o Directores de tesis el análisis del % de similitud para establecer el riesgo o la existencia de un posible plagio.**

Finalmente y posterior a la lectura, revisión individual, así como el análisis e intercambio de opiniones, los miembros de la Comisión manifestaron **APROBAR** ☒ **SUSPENDER** ☐ **NO APROBAR** ☐ la tesis por **UNANIMIDAD** ☒ o **MAYORÍA** ☐ en virtud de los motivos siguientes:
Cumple con los requisitos para una tesis de maestría.

COMISIÓN REVISORA DE TESIS

Dr. Francisco Hiram Calvo Castro
Director de Tesis

Dr. Grigori Sidrov

M. en D. José Juan Guzmán Camacho

Dr. Alexander Gelbukh

Dra. Olga Kolesnikova

Dr. Marco Antonio Moreno Armendáriz

INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
DIRECCIÓN

Dr. Francisco Hiram Calvo Castro
PRESIDENTE DEL COLEGIO DE PROFESORES

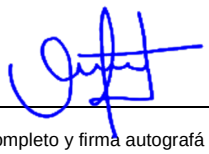
INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA DE AUTORIZACIÓN DE USO DE OBRA PARA DIFUSIÓN

En la Ciudad de México el día 18 del mes de Noviembre del año, el (la) que suscribe Tania Gisela Alcántara Medina alumno(a) del programa Maestría en Ciencias de la Computación con número de registro B200419 adscrito(a) a Ciencias Cognitivas Computacionales manifiesta que es autor(a) intelectual del presente trabajo de tesis bajo la dirección de Dr. Francisco Hiram Calvo Castro y cede los derechos del trabajo intitulado Detección automática de paráfrasis en preguntas en entornos colaborativos, al Instituto Politécnico Nacional, para su difusión con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expresado del autor y/o director(es). Este puede ser obtenido escribiendo a las siguiente(s) dirección(es) de correo. tania.gam@hotmail.com. Si el permiso se otorga, al usuario deberá dar agradecimiento correspondiente y citar la fuente de este.



Nombre completo y firma autografá del (de la) estudiante

Resumen

Las búsquedas en Internet son algo cotidiano, pero debemos ser conscientes de que más de una persona busca el mismo tema con diferentes palabras; esto es una manifestación del fenómeno de *parafraseo*.

Parafrasear implica cambios sintácticos y la superposición de palabras, ligadas a las reglas del idioma en el que trabajamos. La identificación de paráfrasis es un problema de gran importancia para el Procesamiento del Lenguaje Natural (PLN), especialmente parafraseando preguntas con la misma intención.

Además, se ha encontrado que para el estudio de las similitudes no se tienen en cuenta algunas características, lo que hace que la identificación arroje menores resultados.

En esta tesis, abordamos el problema de la identificación automática de paráfrasis en el conjunto de datos *Quora Question Pair* (QQP), prestando especial atención a la forma de los datos a través del análisis exploratorio de datos (EDA) y la búsqueda de patrones, esto es con el fin de obtener mejores resultados en las tareas de identificación, así como comparar diferentes clasificadores con distintas configuraciones.

Palabras clave— PLN, Paráfrasis, EDA, PCA, Machine Learning.

Abstract

Searches on the Internet are commonplace, but we must be aware that more than one person searches for the same topic with different words; this is a manifestation of the *paraphrasing* phenomenon.

Paraphrasing implies syntactic changes and the overlapping of words, linked to the rules of the language in which we work. The identification of paraphrases is a problem of great importance for Natural Language Processing (NLP), especially paraphrasing questions with the same intention.

In addition, it has been found that for the study of similarities some characteristics are not taken into account, which means that the identification yields fewer results.

In this thesis, we address the problem of automatic identification of paraphrases in the *Quora Question Pair* (QQP) data set, paying special attention to the shape of the data through exploratory data analysis (EDA) and the pattern search, this is in order to obtain better results in the identification tasks, as well as compare different classifiers with different configurations.

Palabras claves— NLP, Praphrasis, EDA, PCA, Machine Learning.

Agradecimientos

Primero doy gracias a Dios y a la Vida, por permitirme llegar al final de esta parte de un gran sueño que tengo.

Agradezco al **Instituto Politécnico Nacional** (mi alma máter), al **Centro de investigación en Computación**, por brindarme un lugar de estudio y los conocimientos en mi vida. Agradezco sinceramente a **CONACYT**, por el financiamiento que me otorgo durante mis estudios de posgrado.

Doy gracias a mis **compañeros/amigos** del laboratorio **CsCog** y anexos, por brindarme su amistad y apoyo durante este tiempo que hemos compartido.

Agradezco infinitamente a mi asesor **Dr. Hiram Calvo**, por los conocimientos, guía, dedicación, apoyo brindado en todos los aspectos y la confianza durante mi investigación.

Gracias a mi Madre **Helen**, por apoyarme en mi decisión, no dejarme sola, acompañarme día a día, por buscar la manera de que pudiera salir adelante frente a todo y darme ánimos en mis peores momentos.

Pero sobre todo agradezco a **Omar García**, por ser mi compañero de vida, mi mayor cómplice, por no dejarme caer en los momentos más obscuros de mi vida, por ayudarme a perseguir mis sueños profesionales, personales y de pareja. Amor, sin ti nada de esto sería posible, además tenías razón, **juntos todo es posible**.

Dedicatoria

Sé que me escuchas ahí en el cielo, porque hablamos siempre a través de nuestros corazones.

A ti que te fuiste a la mitad de este sueño...
Noé Alcántara

Índice general

1 Introducción **2**

1.1 Planteamiento del problema 3
1.2 Objetivo general 4
1.3 Objetivos específicos 4
1.4 Novedad científica 4
1.5 Estructura de la tesis 5

2 Marco teórico **6**

2.1 Procesamiento de textos 6
2.1.1 Tokenización 9
2.1.2 Tokenizador WordPiece 10
2.2 Embeddings 10
2.2.1 Tipos de Embeddings 11
2.3 Función SoftMax 14
2.3.1 Cálculo del SoftMax 15
2.4 Redes Neuronales 16
2.4.1 Modelo de una Red Neuronal 18
2.4.2 Redes Neuronales Profundas 19
2.4.3 Redes Neuronales Recurrentes 20
2.5 Mecanismos de Atención 21
2.5.1 Transformers 24
2.5.2 BERT 28
2.5.3 Diferentes BERT 34
2.6 Parafraseo 36
2.6.1 Técnicas de Paráfrasis 36
2.6.2 Análisis Exploratorio de Datos 38
2.7 Técnicas de validación de modelos 40
2.8 Métricas de evaluación 40
2.8.1 Matriz de confusión 40

2.8.2 Exactitud *(Accuracy)* . . . 41
2.8.3 Precisión . . . 41
2.8.4 Exhaustividad (Recall) . . . 41
2.8.5 Medida F1 (F_1 *score*) . . . 42
2.9 Quora Question Pairs . . . 42

3 Estado del Arte **44**
3.1 Comparación de esta propuesta con el estado del arte . . . 46

4 Desarrollo y Solución **48**
4.1 Definición del conjunto de datos . . . 48
4.2 Análisis Exploratorio (Fase 1) . . . 49
4.3 Preprocesamiento . . . 51
4.4 Extracción de Características Manual . . . 52
4.5 Análisis Exploratorio de datos (Fase 2) . . . 53
4.6 Extracción de características automáticas . . . 57
4.7 Análisis Exploratorio (Fase 3) . . . 58
4.8 Suma de Características . . . 59
4.9 Análisis de Componentes Principales (PCA) . . . 59
4.10 Balanceo de los Datos . . . 60
4.10.1 Técnica de balanceo de datos . . . 61
4.11 Clasificación . . . 61
4.11.1 Clasificadores Tradicionales . . . 61
4.11.2 Clasificadores Modernos . . . 62

5 Resultados **68**
5.1 Clasificadores clásicos . . . 68
5.1.1 Resultados Clásicos . . . 69
5.1.2 Resultados modernos . . . 73
5.1.3 Método Propuesto . . . 74
5.2 Mejores Resultados del Análisis . . . 74
5.3 Comparación con el estado del arte . . . 76

6 Conclusiones **77**

7 Trabajo a futuro y Aportaciones **79**
7.1 Aportaciones . . . 79
7.2 Trabajo a futuro . . . 79

8 Trabajos derivados de esta tesis **81**
8.1 Publicaciones 81
8.2 Congresos 81

Bibliografía **81**

Índice de tablas

Tabla 2.1.1 Tipos de *tokenización*. 9
Tabla 2.1.2 Formas de tokenización de una palabra de acuerdo a *WordPiece* (Briggs, 2021). 10
Tabla 2.3.1 Valores de SoftMax (Wood, 2018). 15
Tabla 2.5.1 Valores de Atención (Vaswani et al., 2017). 28
Tabla 2.8.1 Matriz de Confusión. 40
Tabla 2.9.1 Ejemplo del corpus de entrenamiento de QQP (Iyer et al., 2021). 43
Tabla 2.9.2 Ejemplo del conjunto de pruebas de QQP (Iyer et al., 2021). 43

Tabla 4.2.1 Datos del conjunto de datos con pocos caracteres. 51

Tabla 5.1.1 Resultados de la métrica *Accuracy* de los clasificadores aplicados al conjunto de datos sin balancear, con **1561 características.** 69
Tabla 5.1.2 Resultados de la métrica *F1* de los clasificadores aplicados al conjunto de datos sin balancear, con **1561 características.** 69
Tabla 5.1.3 Resultados de la métrica *Accuracy* de los clasificadores aplicados al conjunto de datos sin balancear y **con PCA, reducido a 800 características.** 69
Tabla 5.1.4 Resultados de la métrica *F1* de los clasificadores aplicados al conjunto de datos sin balancear y con **PCA, reducido a 800 características.** . . . 70
Tabla 5.1.5 Resultados de la métrica *Accuracy* a clasificadores aplicados al conjunto de datos sin balancear con **25 características manuales.** 70
Tabla 5.1.6 Resultados de la métrica *F1* a clasificadores aplicados al conjunto de datos sin balancear con **25 características manuales.** 70
Tabla 5.1.7 Resultados de la métrica *Accuracy* de los clasificadores aplicados al conjunto de datos balanceado, con **1561 características.** 71
Tabla 5.1.8 Resultados de la métrica *F1* de los clasificadores aplicados al conjunto de datos balanceado, con **1561 características.** 71
Tabla 5.1.9 Resultados de la métrica *Accuracy* de los clasificadores aplicados al conjunto de datos balanceado y **con PCA, reducido a 800 características.** . 71
Tabla 5.1.10 Resultados de la métrica *F1* de los clasificadores aplicados al conjunto de datos balanceado y con **PCA, reducido a 800 características.** 71

Tabla 5.1.11 Resultados de la métrica *Accuracy* a clasificadores aplicados al conjunto de datos balanceado con **25 características manuales.** 72
Tabla 5.1.12 Resultados de la métrica *F1* a clasificadores aplicados al conjunto de datos balanceado a QQP con **25 características manuales.** 72
Tabla 5.1.13 Comparación de resultados de la métrica *Accuracy* en datos balanceados y sin balancear. 72
Tabla 5.1.14 Comparación de resultados de la métrica *F1* en datos balanceados y sin balancear. 73
Tabla 5.1.15 Resultados de la métrica *Accuracy* a clasificadores modernos. 73
Tabla 5.1.16 Resultados de la métrica *F1* a clasificadores modernos 74
Tabla 5.2.1 Mejores resultados de la métrica *Accuracy* del conjunto de datos QQP con y sin balanceo, aplicados a diferentes clasificadores 75
Tabla 5.2.2 Mejores resultados de la métrica *F1* del conjunto de datos QQP con y sin balanceo, aplicados a diferentes clasificadores 75
Tabla 5.2.3 Matriz de confusión para MADEPAC 2 75
Tabla 5.2.4 Matriz de confusión para MADEPAC 3 75
Tabla 5.3.1 Estado del arte vs. clasificadores propuestos en la métrica *Accuracy* . . . 76
Tabla 5.3.2 Estado del arte vs. clasificadores propuestos en la métrica *F1* 76

Índice de figuras

Figura 2.1.1 Componentes del PLN, (Chowdhary, 2020). 8
Figura 2.2.1 Ejemplo de *word embedding* en un vocabulario de 9 palabras (Lynn, 2018). 11
Figura 2.2.2 Mapeo tridimensional más eficiente de acuerdo al vocabulario (Lynn, 2018). 12
Figura 2.2.3 Mapeo tridimensional ya vectorizado, los *embeddings* similares tienen resultados similares (Lynn, 2018). 13
Figura 2.2.4 La arquitectura SBERT, para calcular las puntuaciones de similitud. (Reimers and Gurevych, 2019). 14
Figura 2.4.1 Esquema de conexiones de un par de neuronas biológicas (Demuth and Jesús, 2014). 17
Figura 2.4.2 Esquema del modelo artificial de una red neuronal (Demuth and Jesús, 2014). 18

Figura 2.4.3 Secuencia de un modelo neuronal profundo (ChatBot, 2019). 20
Figura 2.4.4 Base de una RN y RNN a una sola neurona (Rivera, 2018). 20
Figura 2.4.5 RNN desenrollada (Rivera, 2018). 21
Figura 2.5.1 Diagrama del *modelo mecanismos de atención* (Bahdanau et al., 2015). . 22
Figura 2.5.2 Matriz de atención de una traducción (Bahdanau et al., 2015). 24
Figura 2.5.3 Modelo de Transformer (Vaswani et al., 2017). 25
Figura 2.5.4 Partes del modelo transformer (Vaswani et al., 2017). 26
Figura 2.5.5 Producto escalar de atención (Vaswani et al., 2017). 27
Figura 2.5.6 Atención por Multicabezal (Vaswani et al., 2017). 28
Figura 2.5.7 Representación de la entrada pasando por los tokens *embeddings*, segmentación y posicionamiento de *embeddings*, (Devlin et al., 2018). . . . 30
Figura 2.5.8 *Token embeddings* obtenidas en la matriz de tamaño *vocabulario*$\times H$ (Montanes, 2021). 30
Figura 2.5.9 *Segment embedding* están todos en 0 o todos en 1, especificando si la frase pertenece a la oración 1 o a la oración 2 (Montanes, 2021). 31
Figura 2.5.10MLM enmascaramiento de palabras de BERT (Montanes, 2021). 32
Figura 2.5.11Ilustración de *Fine-tuning* de BERT en diferentes tareas (Devlin et al., 2018). 33
Figura 2.5.12*Fine-tuning para la tarea de respuesta a preguntas*(Montanes, 2021). . . 34
Figura 2.5.13*Configuraciones de BERT* (Devlin, 2020). 35
Figura 2.5.14*Extracto de resultados de GLUE para cofiguraciones más pequeñas de BERT* (Devlin, 2020). 35

Figura 4.2.1 Clases de QQP. 50
Figura 4.2.2 Número de preguntas repetidas dentro de QQP. 50
Figura 4.5.1 Gráficas del Análisis Exploratorio de Datos para Características Manuales 1 53
Figura 4.5.3 Gráficas del Análisis Exploratorio de Datos para Características Manuales 3. 54
Figura 4.5.2 Gráficas del Análisis Exploratorio de Datos para Características Manuales 2. 54
Figura 4.5.4 Gráficas del Análisis Exploratorio de Datos para Características Manuales 4. 55
Figura 4.5.5 Gráficas del Análisis Exploratorio de Datos para Características Manuales 5. 55
Figura 4.5.6 Gráficas del Análisis Exploratorio de Datos para Características Manuales 6. 56

Figura 4.5.7 Gráficas del Análisis Exploratorio de Datos para Características Manuales 7. 56
Figura 4.5.8 Gráficas del Análisis Exploratorio de Datos para Características Manuales 8. 56
Figura 4.5.9 Gráficas del Análisis Exploratorio de Datos para Características Manuales 9. 57
Figura 4.5.10 Proporción de la densidad de probabilidad del conjunto ordenado de la característica *fuzz* de la librería fuzzywuzy. 57
Figura 4.7.1 Gráficas del Análisis Exploratorio de Datos para Características Automáticas 1. 58
Figura 4.7.2 Gráficas del Análisis Exploratorio de Datos para Características Automáticas 2. 58
Figura 4.9.1 Importancia de las, características de acuerdo a PCA. 60
Figura 4.11.1 Arquitectura de la red siamesa. 63
Figura 4.11.2 Gráfica de función de error de la red siamesa. 64
Figura 4.11.3 Gráfica de métrica *accuracy* de la red siamesa. 64
Figura 4.11.4 Tensores por par de preguntas con separador. 65
Figura 4.11.5 Arquitectura del codificador de Small BERT. 66
Figura 4.11.6 Arquitectura del clasificador de BERT. 67

Capítulo 1

Introducción

Desde el inicio de la vida humana, ha existido la necesidad de tener representaciones escritas de nuestra lengua hablada. Debemos recordar que la evolución del lenguaje escrito no ha sido tan rápido como el lenguaje oral.

Las primeras representaciones gráficas fueron los pictogramas, las cuales representan palabras o frases completas y fueron labradas en piedras, desde ese momento y hasta nuestros días hemos desarrollado diferentes lenguajes con estructuras y reglas diferentes, ahora cada palabra tiene un significado y aún existen palabras que representan frases completas, pero todas ellas nos han ayudado a comunicarnos.

La escritura como representación gráfica es un punto muy importante dentro de nuestra cognición, ya que proporciona dentro de nuestra mente la estructura y organización para generarla, lo que da como resultado que la generación de un texto no es una tarea lingüística sencilla de realizar.

Al escribir encontramos una inspiración de manera directa e indirecta en diferentes autores, lo que hace que parafrasear se vuelva supremamente importante, pero, esta tarea es aún más compleja, ya que supone un entendimiento del contexto, lo que da como resultado un análisis, razonamiento lógico de manera crítica y aplicación de conocimientos previos en el lenguaje.

Bajo este contexto, la identificación de esas similitudes se convirtió en una tarea que debe ser transferida para que sea realizada de manera más sencilla y óptima. La identificación de paráfrasis es la tarea en la que se debe identificar un par de frases, oraciones, preguntas y determinar si

ésta es o no parafraseo.

En esta tesis se presenta la identificación de paráfrasis, desde un método propuesto denominado ***Método Exploratorio Profundo antes de la Clasificación*** **(MADEPAC)**, el cual ayudara para encontrar patrones no definidos para el conjunto de datos, identificar las mejores características además de crear nuevas.

1.1. Planteamiento del problema

La identificación de paráfrasis o en inglés *Paraphrase identification (PI)* se trata de detectar diferentes expresiones lingüísticas con la misma intención o similitud. Esto puede ser a diferentes niveles textuales (nivel documento, nivel párrafo, nivel frase, nivel palabra o combinación entre palabras) (Bhagat and Hovy, 2013). El parafraseo requiere un proceso de comprensión sintáctica y semántica, lo que hace que la tarea que se convierta en un tema de interés dentro del procesamiento de lenguaje natural (PLN) (Flores, 2014a). Ya que el parafraseo automático de un texto puede ser utilizado para múltiples aplicaciones, como la búsqueda de respuestas, el resumen automático, el análisis de plagio y la traducción automática, es importante conocer un ejemplo:

Ejemplo de paráfrasis:

- **Original.** "La finalidad del arte es dar cuerpo a la esencia secreta de las cosas, no el copiar su apariencia", decía Aristóteles.

- **Paráfrasis.** Según Aristóteles, el arte tiene la misión de encarnar la esencia oculta de la realidad, en vez de simplemente copiar su apariencia.

Hoy en día al realizar una búsqueda por internet realizamos una pregunta, pero debemos estar conscientes de que otras personas realizan estas preguntas, pero de otra manera, teniendo esto en cuenta la identificación de en preguntas similares.

La identificación en pares de oraciones es un problema central en la comprensión del lenguaje natural, pero cuando hablamos de pares de oraciones, solo pensamos en párrafos y no en preguntas.

Las preguntas son un tipo de paráfrasis poco explorada y es lo que nos encontramos a diario, por ejemplo, las búsquedas diarias en Internet que ya han realizado otros usuarios, el conjunto de datos Quora Question Pairs (QQP) nos proporciona este caso de estudio, con una gran variedad de preguntas parafraseadas.

1.2. Objetivo general

Detectar paráfrasis de manera automática en pares de preguntas a través del análisis exploratorio de datos y el análisis de componentes principales, con el fin de optimizar la clasificación en diferentes modelos.

1.3. Objetivos específicos

Los objetivos específicos que contribuirán a desarrollar el *objetivo general* del trabajo son los siguientes:

- Reconocer los principales patrones del conjunto de datos.
- Extraer de manera manual y automática las características de estos patrones.
- Comparar arquitecturas modernas sin un análisis profundo vs tradicionales con un análisis profundo.
- Determinar las mejores características para la clasificación de estos patrones.
- Determinar la mejor clasificación con diferentes algoritmos, combinaciones y escenarios basados en los patrones encontrados y las mejores características.

1.4. Novedad científica

Los desarrollos actuales se han centrado en aplicar diferentes algoritmos, novedosos y modernos, pero han dejado de lado el análisis de los datos; ¿Cuáles son los patrones que siguen a las preguntas? ¿Cuántas repeticiones existen en mi conjunto ¿Qué reglas gramaticales son las más utilizadas? Estas son solo algunas preguntas que serán respondidas a través del análisis previo antes de la clasificación y diferentes técnicas, tales como el EDA y PCA, las cuales harán que la aplicación de algoritmos obtenga mejores resultados que los aplicados sin análisis.

1.5. Estructura de la tesis

La tesis presentada se encuentra organizada de la siguiente manera.

1. **Capítulo 1:** Se encuentran los antecedentes de la investigación, objetivo general, específicos y la estructura del trabajo.
2. **Capítulo 2:** Se encuentran los conocimientos de investigación necesarios para realizar este trabajo.
3. **Capítulo 3:** Incluye un análisis de los trabajos relacionados con la generación de textos y parafraseo.
4. **Capítulo 4:** Proceso para el desarrollo y solución de este proyecto.
5. **Capítulo 5:** Resultados de este trabajo.
6. **Capítulo 6:** Conclusiones
7. **Capítulo 7:** Trabajo a futuro y las aportaciones de este trabajo
8. **Capítulo 8:** Publicaciones y congresos donde fue presentado este trabajo.

Capítulo 2

Marco teórico

2.1. Procesamiento de textos

El lenguaje es la capacidad en la que el ser humano logra expresar pensamientos, emociones y sentimientos de manera verbal y escrita. Los lenguajes a través del tiempo han sufrido modificaciones y se han añadido nuevas palabras a cada uno, con todo esto, los lenguajes contienen ambigüedades.

Los lenguajes contienen de manera finita oraciones construidas a partir de los diferentes alfabetos y los términos sintácticos. “Dado que el conjunto del alfabeto es finito, así como la longitud de las oraciones, el conjunto de oraciones (en un lenguaje) también es finito. Por ejemplo, si el conjunto del alfabeto es de tamaño dos, y la longitud de las frases es diez, solo puede haber 1024 número máximo de frases posibles” (Chowdhary, 2020, p. 604).

Cuando el estudio es sobre un lenguaje infinito, se pueden utilizar gramáticas generadoras para analizar esta estructura. En este punto, las teorías del lenguaje derivadas del nacimiento de la computación, contienen especificaciones de las que se pueden extraer gramáticas específicas. Los científicos han tratado de replicar e insertar esta interacción dentro de las máquinas, tratado de abstraer las características del lenguaje.

Con todo lo anterior nace el Procesamiento de Lenguaje Natural, abreviado PLN o *NLP* por sus siglas en inglés *(Natural Language Processing)*, el cual es un conjunto de mecanismos de análisis y representación de la lingüística, generalmente plasmados de manera textual, extraídos de diferentes fuentes de información. Todo esto nace motivados por las teorías del lenguaje.

La descripción del PLN podrá sonar muy simple, pero en análisis automático de textos, al mismo nivel que lo podemos realizar, los humanos requiere un entendimiento profundo del lenguaje, a través de las bondades de la lengua en la que se está trabajando. El principal objetivo es que las máquinas, puedan tener este nivel (Chowdhary, 2020), esto a través de las estructuras de carácter sintáctico, semántico, las reglas y principios de combinación para la construcción de las palabras.

Existen diferentes líneas de trabajo dentro del PLN: La recuperación de textos, análisis de textos y generación de textos, esto a nivel de palabras, frases o párrafos.

El PLN requiere métodos de tipo simbólicos de alto nivel, siendo los siguientes (Chowdhary, 2020):

1. Compresión de características léxicas, semánticas y episódicas.
2. Manipulación de estructuras recursivas.
3. Módulos de procesamiento y aprendizaje.
4. Identificación de construcciones lingüísticas básicas.
5. Representación abstracta.

A través de la figura 2.1.1 se pueden visualizar los componentes del PLN, los cuales son los pasos para determinar la estructura y significado para el análisis y generación del lenguaje, todo esto a través de la teoría.

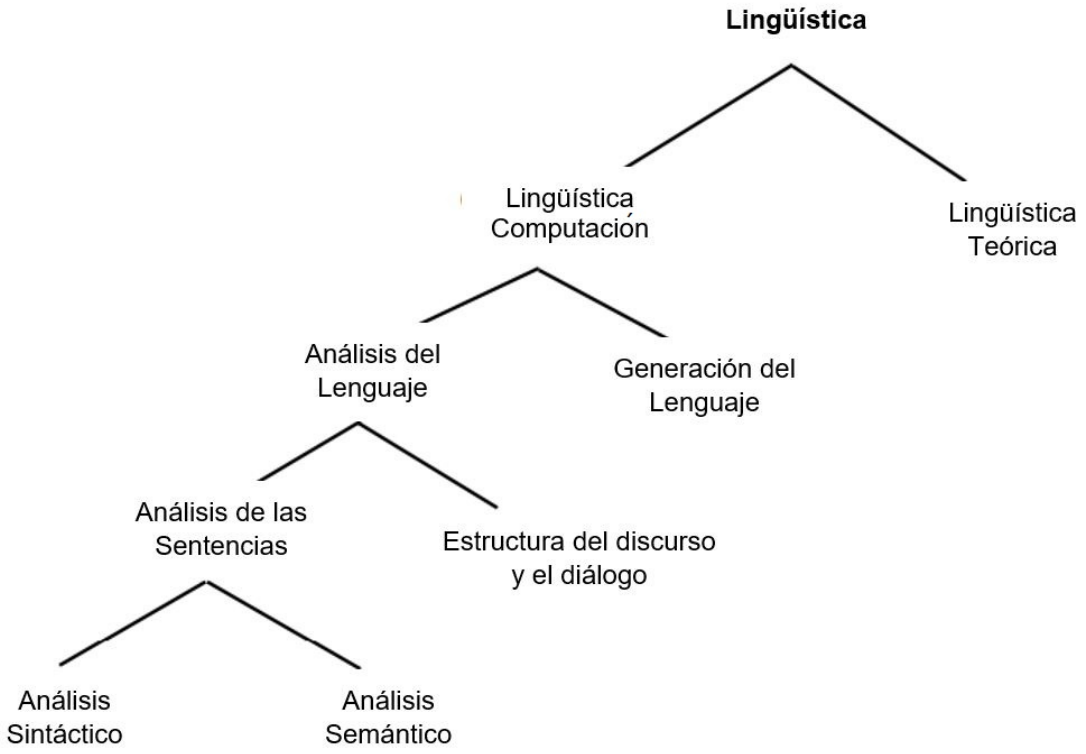


Figura 2.1.1: Componentes del PLN, (Chowdhary, 2020).

"La meta de este campo es que las computadoras puedan desarrollar tareas útiles que involucran el lenguaje humano, tareas como habilitar la comunicación humano-máquina, mejorar la comunicación humano-humano o hacer simplemente procesamiento de texto y del habla" (Calvo, 2013, p. 604)

Para la tarea de procesamiento de textos, se necesita un conjunto de datos en formato electrónico de tipo textual, a este le llamaremos *corpus*. Este corpus necesitará una segmentación de textos, un procedimiento para identificar las palabras y en algunos casos diferentes sentencias, es decir, dónde comienzan y dónde terminan. Como resultado, este proceso produce *tokens* y al procedimiento de segmentación se le llama *tokenización*.

Derivado de las ambigüedades de los lenguajes, se necesita un proceso de normalización del texto, es decir, una transformación del texto a una forma canónica. Durante este proceso se pueden incluir correcciones ortográficas, acentuación, uso de signos de puntuación, uso de mayúsculas, abreviaciones y acrónimos.

2.1.1. Tokenización

El PLN está basado en gran medida en el sentido de las palabras y lo que transmiten, lo que hace imprescindible preprocesar la información antes de trabajar con ella.

La *tokenización* es el proceso de separar largas oraciones de texto en "trozos" más pequeños, esta separación se realiza de acuerdo a espacios o signos de puntuación, según se requiera. A continuación se describirán los diferentes tipos de *tokenización*:

1. **Tokenización por espacios en blanco:** Esta es la *tokenización* más usual, se trata de la división del texto cada vez que se encuentre un espacio en blanco.
2. **Tokenización por espacios de puntuación:** Este método realiza una mezcla de la *tokenización* basada en espacios en blanco y con puntuaciones, es decir, cada vez que se encuentre un signo de puntuación o espacio en blanco realizará un *token*.
3. ***Tokenización* por *TreebankWordTokenizer:*** Este es un proceso de *tokenización* utilizado expresiones regulares, el cual supone que el texto está dividido en oraciones. Este tipo de *tokenización* incorpora reglas de lingüística.
4. ***Tokenización* por *TweetTokenizer*:** Este método está especialmente pensado para *tweets* de Twitter, los cuales tienen una estructura especial, este puede eliminar el código *HTML*, eliminar caracteres especiales como identificadores de Twitter y en algunos casos, normalizar la longitud de los textos con la eliminación de palabras repetidas.
5. ***Tokenización* por MWETokenizer:** Este es un método basado en reglas, una vez que el texto haya sido *tokenizado* por algún método, algunos *tokens* se pueden reagrupar en expresiones de varias palabras.

Para comprender mejor esto, la tabla 2.1.1 muestra las *tokenizaciones* en la frase: *¡Es verdad, Srta. María Martínez!*

Tabla 2.1.1: Tipos de *tokenización*.

Tipo de Tokenización	Tokens
Basada en Espacios en Blanco	['¡Es', 'verdad,', 'Srta.', 'Maria', 'Martinez!']
Basada en Puntuación	['¡', 'Es', 'verdad', ',' , 'Srta', '.', 'Maria', 'Martinez', '!']
Basada en TreebankWordTokenizer	['¡', 'Es', 'verdad', ',' , 'Srta.', 'Maria', 'Martinez', '!']
Basada en TweetTokenizer	['¡', 'Es', 'verdad', ',' , 'Srta', '.', 'Maria', 'Martinez', '!']
Basada en MWETokenizer	['¡', 'Es', 'verdad', ',' , 'Srta', '.', 'Maria_Martinez', '!']

2.1.2. Tokenizador WordPiece

El *tokenizador* de wordpiece tiene dos formas de trabajo:

1. **Formas completas:** Una palabra se convierte en un token.
2. **Piezas de palabra:** Una palabra puede ser dividida en múltiples tokens.

En la tabla 2.1.2 se visualizan las formas de trabajo, estas pueden ser mezcladas, es decir, no se debe usar solo un tipo.

Tabla 2.1.2: Formas de tokenización de una palabra de acuerdo a *WordPiece* (Briggs, 2021).

Word	Tokens(s)
surf	['surf']
surfing	['surf', '##ing']
surfboarding	['surf', '##board', '#ing#']
surfboard	['surf', '##board']
snowboard	['snow', '##board']
snowboarding	['snow', '##board', '##ing']
snow	['snow']
snowing	['snow', '##ing']

Como se puede visualizar en la tabla 2.1.2, al dividir las palabras se pueden identificar que algunas comparten significado, tal es el caso de la palabra *snowboard* y *surfboard*, las cuales comparten el significado de la palabra *#board* (Briggs, 2021).

Este tipo de *tokenizador* permitirá identificar con mayor facilidad las palabras relacionadas, esto permitiría reducir, ya que comparten los mismos *tokens* (Briggs, 2021).

2.2. Embeddings

Embeddings o *word embeddings* (en español, integración (conceptual) de palabras), se trata de un conjunto de técnicas y modelos de PLN, donde las frases y/o palabras son representadas de manera numérica. Esta técnica es utilizada para mejorar el rendimiento y dar un conjunto de valores reales a las palabras.

Los *embeddings* pueden ser vectores de N-dimensiones, que intentarán tener de manera más clara el significado y contexto de cada palabra. En estos sistemas, cualquier conjunto de números puede ser un vector de palabra válido (Lynn, 2018). Existen algunas características para que los *embeddings* sean útiles:

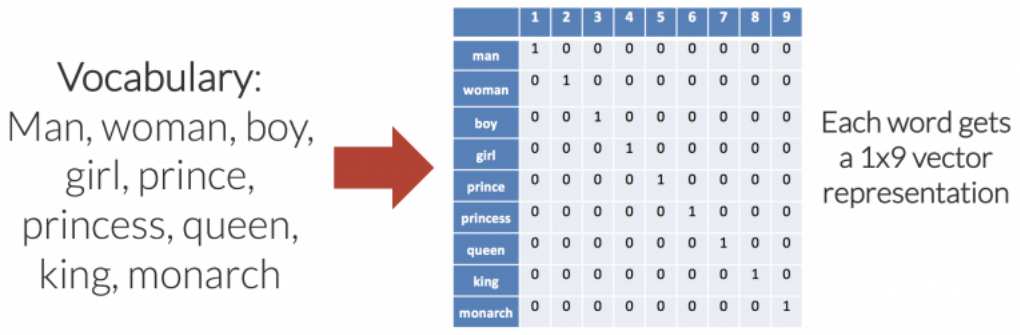
1. Cada palabra tiene un *embedding* único.
2. Cada uno puede ser multidimensional, que pueden ir de un rango de 50 a 500 de longitud.
3. Para cada palabra, el significado debe ser claro.
4. Si las palabras son similares, su *embedding* también lo será.

Una de las principales propiedades de los *embeddings* es la identificación de reconocimiento de palabras similares y que son capturadas de forma natural.

2.2.1. Tipos de Embeddings

One-hot

El modelo más utilizado es el llamado *one-hot* o codificación 1 a N. En este caso la cantidad de palabras y de *embeddings* será la misma; cada palabra procesada se contendrá ceros, con un 1 que corresponde a la posición de la palabra. La figura 2.2.1, representa este tipo de *embedding* en un corpus corto de 9 palabras.



	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Figura 2.2.1: Ejemplo de *word embedding* en un vocabulario de 9 palabras (Lynn, 2018).

Aunque es un sistema excelente para la codificación, el número de dimensiones puede aumentar de manera lineal a medida que se añaden palabras, además de que la información entre palabras comunes no es posible.

Custom Encoding

Para este tipo de representación se utilizan menos números para representar de manera manual las dimensiones que tengan referencias del vocabulario y utilizando otra clasificación, para entender mejor este ejemplo, pondremos de contexto un vocabulario de 9 palabras: Man, Woman, Girl, Prince, Princess, Queen, King y Monarch (en español hombre, mujer, chica, príncipe, princesa, rey y monarca), como dimensión se utiliza *femininity*, *youth royalty* (en español, feminidad, juventud y realeza) (Lynn, 2018) .

Cada uno de los términos, le asignamos los valores válidos de acuerdo a nuestra clasificación, como resultado el *embedding* será más corto (Lynn, 2018)

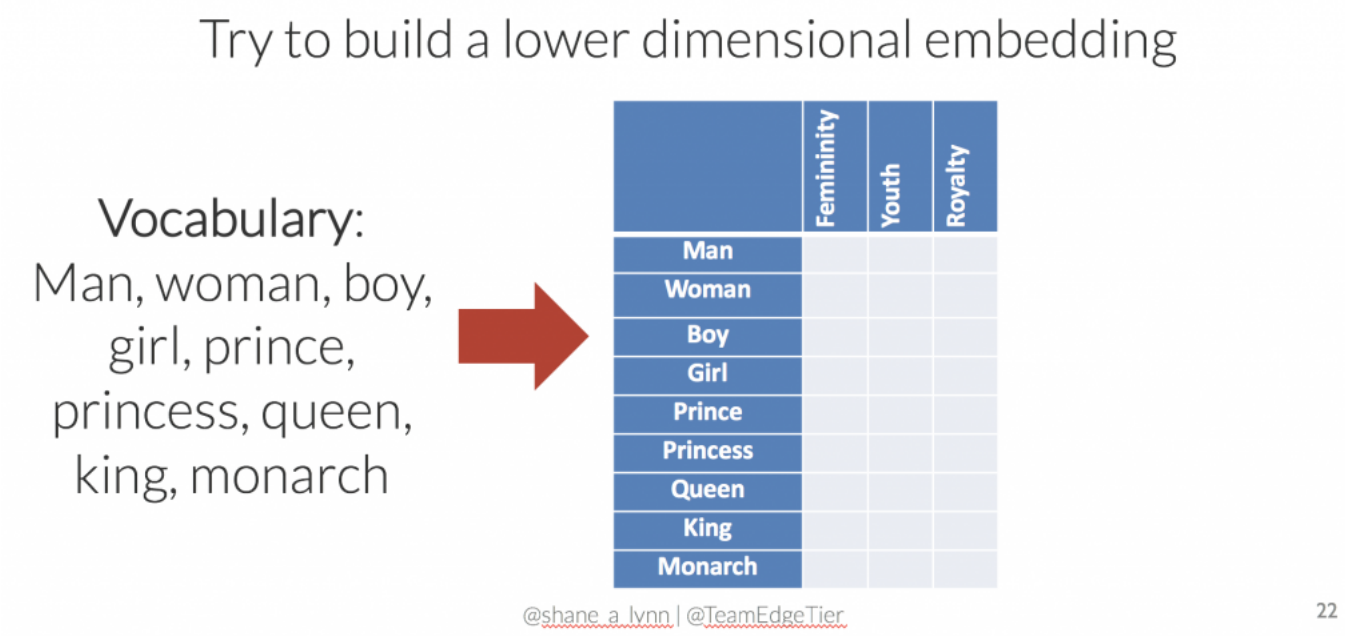


Figura 2.2.2: Mapeo tridimensional más eficiente de acuerdo al vocabulario (Lynn, 2018).

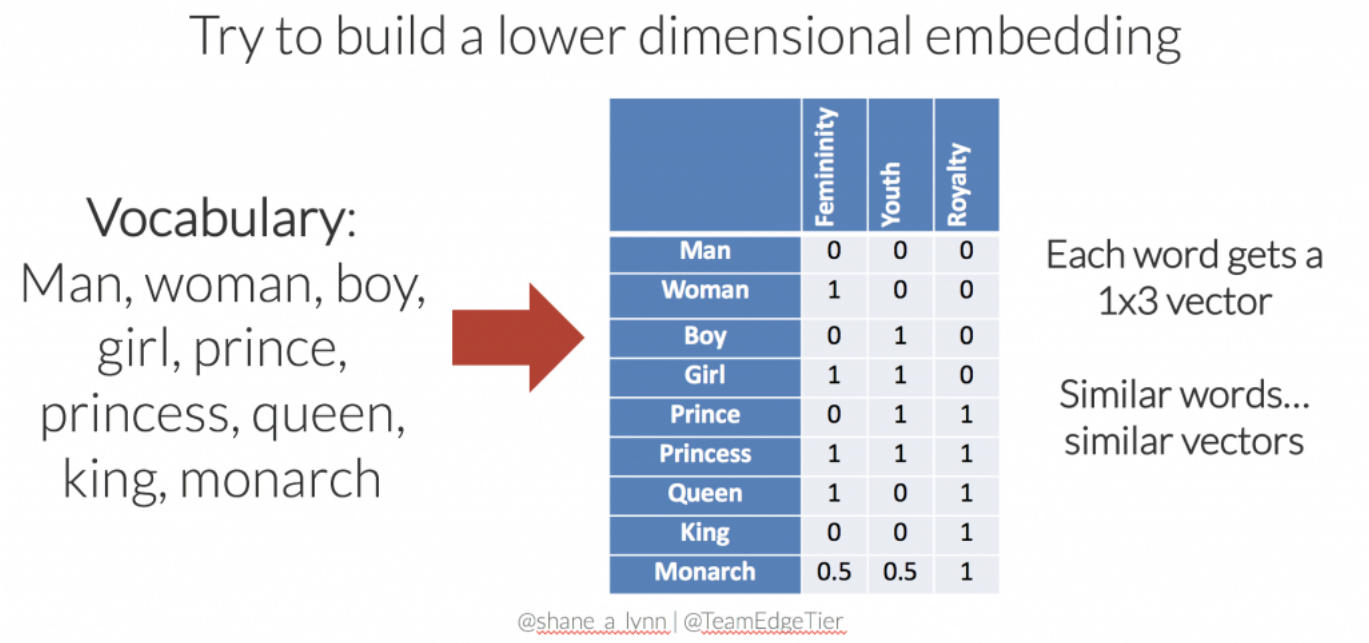


Figura 2.2.3: Mapeo tridimensional ya vectorizado, los *embeddings* similares tienen resultados similares (Lynn, 2018).

Este tipo de *embedding* tiene sus ventajas:

1. El conjunto es más eficiente, y cada palabra es capaz de representar un vector tridimensional.
2. Las palabras similares pueden tener menor distancia en sus *embeddings*.
3. La matriz reduce su dispersión, es decir, incluye menos espacios vacíos.

La figura 2.2.2 muestra la estructura del *embedding* vacío y la figura 2.2.3 muestra la estructura llena; en ella podemos ver las representaciones y sobre todo que las palabras similares incluyen una menor distancia. Este tipo de *embbedings* tienen la posibilidad de ampliar el vocabulario a más de 9 palabras, estos vectores podrán extenderse a ser más de N-dimensiones, que contendrán un significado o un contexto de la palabra más simple.

SentenceBERT

Sentence-Bert (SBERT) (Reimers and Gurevych, 2019), es una modificación de la red pre entrenada de *BERT*, que usa estructuras de red siamesas y tripletas para derivar *embeddings* de frases semánticamente significativas, que pueden ser comparadas utilizando la similitud coseno.

Mediante ese sistema, se reducen los esfuerzos para encontrar similitudes, además de tener reducciones de tiempo, manteniendo la exactitud del *embedding* de BERT (Reimers and Gurevych, 2019).

Reimers and Gurevych (2019), aseguran que esta arquitectura de red siamesa permite obtener vectores de tamaño fijo para las frases de entrada, a través de una medida de similitud como la similitud coseno o la distancia euclidiana. Lo anterior, permitirá encontrar frases semánticamente similares. Estas medidas de similitud pueden realizarse de forma extremadamente eficiente en el hardware moderno, permitiendo que *SBERT* se utilice para la búsqueda de similitud semántica, así como para agrupación de datos.

En la figura 2.2.4 se muestra la arquitectura SBERT en la inferencia, por ejemplo, para calcular las puntuaciones de similitud. Esta arquitectura también se utiliza con la función objetivo de regresión.

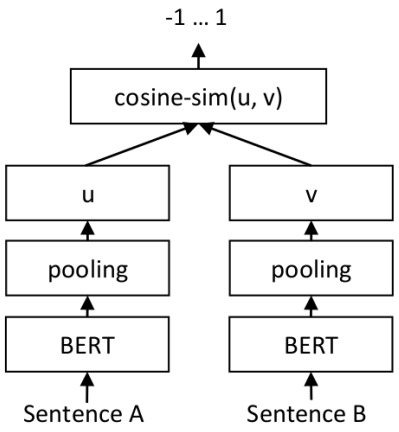


Figura 2.2.4: La arquitectura SBERT, para calcular las puntuaciones de similitud. (Reimers and Gurevych, 2019).

2.3. Función SoftMax

La función softmax es una función que convierte un vector de K valores reales en un vector de K valores reales que suman 1. Los valores de entrada pueden ser positivos, negativos, cero o mayores que uno, pero softmax los transforma en valores entre 0 y 1, para que puedan interpretarse como probabilidades. Si una de las entradas es pequeña o negativa, el *softmax* la convierte en una probabilidad pequeña, y si una entrada es grande, la convierte en una probabilidad grande, pero siempre permanecerá entre 0 y 1.

Una función *SoftMax* se trata de una conversión de un vector con K valores reales a otro que suma 1 (Wood, 2018). La entrada pueden ser valores positivos, negativos, cero o mayor que uno, no importa, ya que la función realizará la transformación en valores 0 y 1. La conversión del *softmax* está dentro de 2 rangos superiores e inferiores, es decir, si el valor es una probabilidad pequeña irá hacia 0, si es grande irá hacia 1.

Este tipo de función es utilizada en las Redes Neuronales, en donde en la penúltima capa genera valores escalados y la función ayuda a normalizar estas cantidades.

$$\sigma(\vec{z})_i = \frac{e^{zi}}{\sum_{j=1}^{K} e^{zi}}$$

Tabla 2.3.1: Valores de SoftMax (Wood, 2018).

Símbolo	**Descripción**
$\sigma(\vec{z})$	Entrada de la función, desde z_0 hasta z_k.
z_i	Los valores de z_i pueden tomar cualquier valor dentro del vector de entrada
e^{zi}	Valor exponencial que se aplicará a cada elemento del vector de entrada.
$\sigma(\vec{z})_i = \sum_{j=1}^{k} e^{zj}$	Asegura que todos los valores de salida sumarán 1 y están en el rango de 0,1.
K	Clasificador Multiclase

2.3.1. Cálculo del SoftMax

Supongamos una matriz que contiene tres valores reales, para dichos valores se requiere una distribución de probabilidad.

$$\begin{bmatrix} 8 \\ 5 \\ 0 \end{bmatrix} \tag{2.1}$$

$$e^{z1} = e^{8} = 2981.0$$
$$e^{z2} = e^{5} = 1148.4$$
$$e^{z3} = e^{0} = 1.0$$

Para este paso, se obtiene la mitad de la ecuación sumando los términos exponenciales.

$$\sum_{j=1}^{K} e^{z_j} = e^{z_1} + e^{z_2} + e^{z_3} = 2981.0 + 148.4 + 1.0 = 3130.4$$

Después, se divide por el término de normalización, obteniendo la salida *softmax* para cada elemento.

$$\vec{z_1} = \frac{2981.0}{3130.4} = 0.9523$$
$$\vec{z_2} = \frac{148.4}{3130.4} = 0.0474$$
$$\vec{z_3} = \frac{1.0}{3130.4} = 0.0003$$

Las salidas de cada función de transferencia, se interpretan como las probabilidades asociadas a cada elemento de la matriz (Demuth et al., 2014). Cada salida estará en los rangos de 0 y 1 y la suma de las mismas será = 1, como en nuestro caso la sumatoria de:

$$0.9523 + 0.0474 + 0.0003 = 1$$

2.4. Redes Neuronales

Las Redes Neuronales o NN por sus siglas en inglés *(Neural Network)*, se inspiran en sus homólogas las biológicas alojadas en nuestro cerebro. Se estima que el cerebro humano tiene aproximadamente 86 billones de neuronas y cada una de ellas interconectadas. De manera biológica una neurona la componen 3 partes:

1. **Dendritas (*Dendrites*):** Son cada una de las partes de la membrana de una célula nerviosa, se encarga de transmitir la información.
2. **Cuerpo celular (*Cell Body*):** Al recibir la información de las dendritas, pueden enviar la información al axón, este envío dependerá de la fuerza de la señal.
3. **Axón (*Axon*):** Son las encargadas de conducir la información, hacia otra neurona.

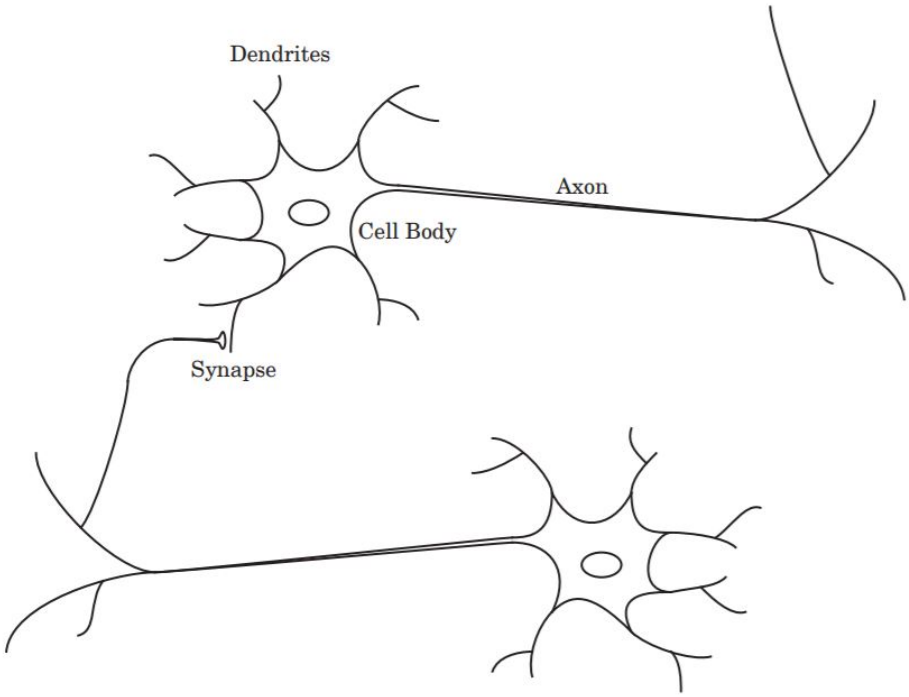


Figura 2.4.1: Esquema de conexiones de un par de neuronas biológicas (Demuth and Jesús, 2014).

En la figura 2.4.1 se puede ver de manera esquemática las conexiones entre dos neuronas, la dendrita, el cuerpo celular y el axón. Cuando existe una conexión entre el axón de una célula y la dendrita de otra se le llama ***sinapsis*** (en inglés, *Synapse*).

Las NN han tratado de construir a partir del modelo biológico bloques, donde se simulan estas conexiones de manera más simple, ya que no se han acercado a la complejidad de aprendizaje del cerebro.

A partir de esto McCulloch and Pitts (1943), a través de su artículo llamado *A logical calculus of the ideas immanent in nervous activity*, crearon un modelo de redes neuronales llamado, *lógica umbral*. Este es el primer paso hacia la investigación de las redes neuronales artificiales (RNA) y como se representan hoy en día.

2.4.1. Modelo de una Red Neuronal

Para representar este modelo de manera igualitaria con una red neuronal biológica, pondremos una red neuronal de una sola entrada, en donde las partes mencionadas de una red biológica son puestos de manera matemática y que sustituyen a las partes biológicas: entradas, pesos y bias.

Dentro de una red biológica, lo que impulsara al axón a enviar o no la señal recibida será la fuerza de la señal, dentro de una red artificial la suma de las entradas multiplicadas asociadas a sus pesos, determinarán el impulso. Este valor se procesa a través de una función de activación que dará un valor que será la salida de la neurona (Demuth and Jesús, 2014). Las partes de una RNA son:

1. **Entradas (*Inputs*):** Parte receptora de los datos o señales procedentes del entorno.
2. **Peso de las Entradas (denotados como *W*):** Cada entrada tendrá un valor o en este caso peso, que determinara la entrada.
3. **Módulo de Suma (Visualizado con el símbolo de sumatoria):** Esta parte se encarga de sumar las entradas ya multiplicadas de cada entrada, por los pesos, dando un solo valor.
4. **Bias (denotado como *b*):** Dará la pauta de impulsar la carga eléctrica.
5. **Función de Activación: (Visualizado por el símbolo matemático de función** f**)** Calcula el valor final de acuerdo a la función activadora, y decide si esa neurona se activará o no.

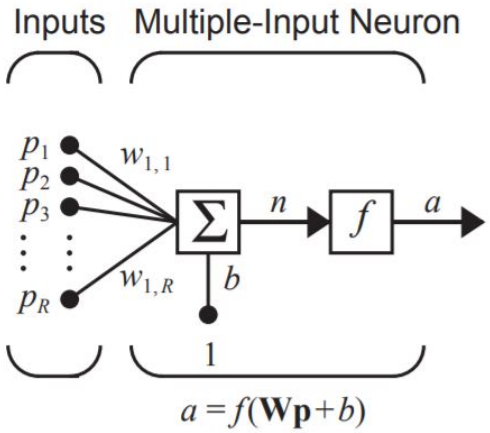


Figura 2.4.2: Esquema del modelo artificial de una red neuronal (Demuth and Jesús, 2014).

La Figura 2.4.2 ejemplifica el funcionamiento del modelo más simple de la RN, una neurona:

1. Entrada de datos numéricos
2. Multiplicación de los datos de entrada por el peso de la entrada.
3. Ingreso al módulo donde serán sumados junto con un bias.
4. El valor obtenido pasará por una función de activación, la cual dará el valor final de la neurona.
5. El valor final será denotado por A.

Matemáticamente, esta función se denota como:

$$A = \varphi(X_1 w_1 + X_2 w_2 + X_3 w_3 ... X_n w_n + b)$$

La red neuronal profunda contendrá al menos una capa, los parámetros mínimos son el número de neuronas, la función de activación y la dimensión de los datos de entrada.

2.4.2. Redes Neuronales Profundas

Redes Neuronales Profundas o *Deep Learning* en inglés (DL), son la versión mejorada de las RN clásicas. Estas son más complejas y contienen al menos tres capas:

1. **Capa de entrada:** Estas son las receptoras de los datos de entrada de la red.
2. **Capas ocultas:** Capas intermedias, para que el sistema se considere DL contendrá al menos una capa oculta.
3. **Capa de salida:** Esta capa codificará la salida, y se tendrán tantas salidas como se requiera.

En la figura 2.4.3 se puede visualizar una red neuronal de 4 capas.

La figura 2.4.3 se puede ver un ejemplo de una RN profunda, esto con capas intermedias que van desde *1...n* capas.

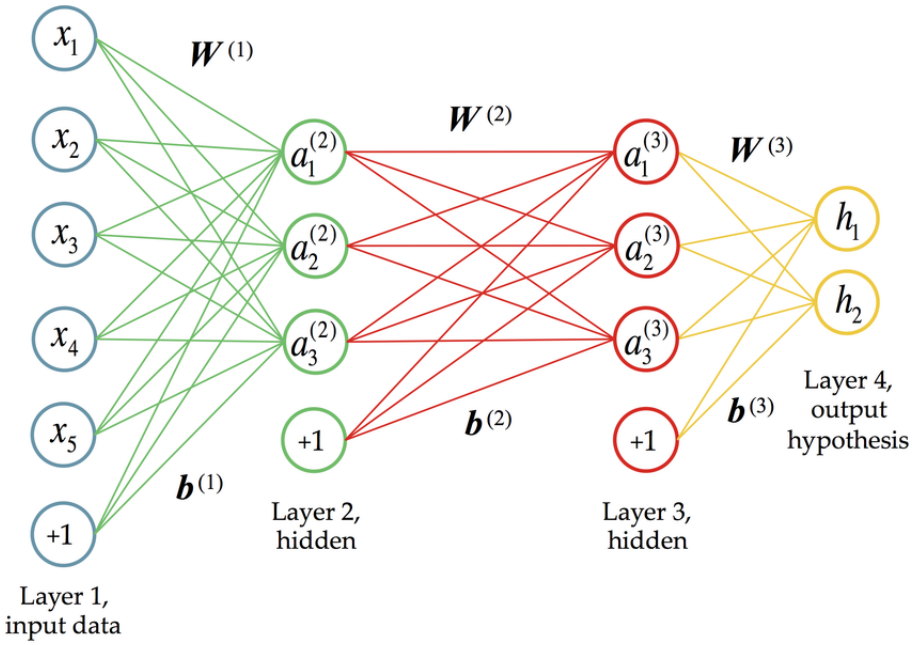


Figura 2.4.3: Secuencia de un modelo neuronal profundo (ChatBot, 2019).

2.4.3. Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes (RNN) o en inglés *Recurrent Neural Networks* (RNN por sus siglas en inglés), son un tipo de Red Neuronal que contienen la propiedad de dirigir la salida de una neurona hacia la entrada de otra, es decir, tienen la propiedad de ir de adelante y hacia atrás, de manera bidireccional recorriendo y retroalimentando las entradas (Torres, 2018).

Para visualizar este comportamiento en la forma más simple, pondremos la RNN en una sola neurona, en la figura 2.4.4 podemos visualizar el comportamiento común de una NN y el que realizan las RNN, en la cual recibe una entrada y genera una salida que para nuestros casos es enviada a sí misma.

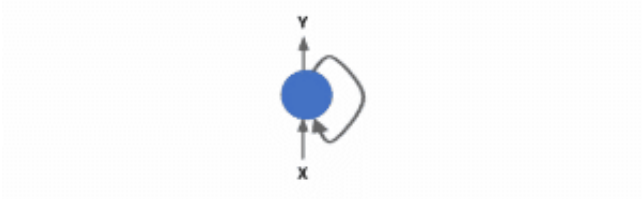


Figura 2.4.4: Base de una RN y RNN a una sola neurona (Rivera, 2018).

A la neurona que entra en recurrencia la nombraremos "*y*", recibirá la entrada de la capa anterior y al mismo tiempo su propia salida, este proceso ocurre en un solo proceso y eso al final generará una salida. Este concepto puede ser aplicado a más de una neurona, a una capa, por ejemplo. En la figura 2.4.5 se puede ver el proceso de retroalimentación: La primera entrada corresponde a la salida de la capa anterior; La segunda entrada, corresponde a la salida anterior.

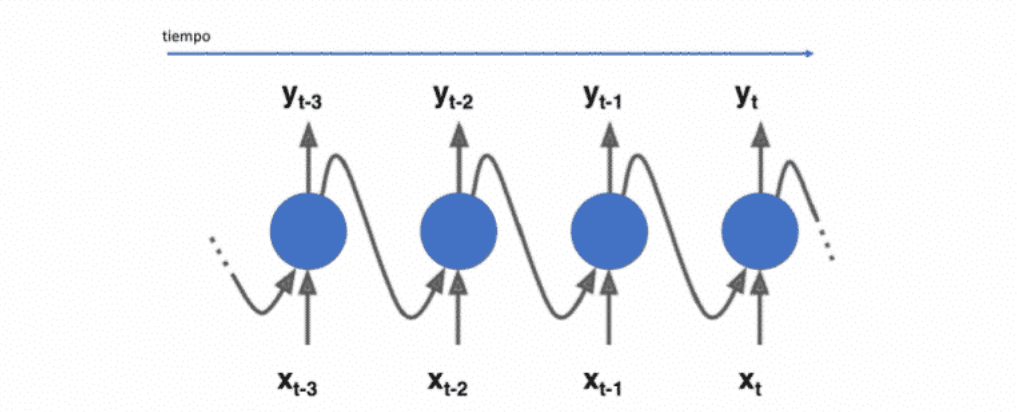


Figura 2.4.5: RNN desenrollada (Rivera, 2018).

Ahora, en cada neurona contendrá dos parámetros, uno correspondiente a la entrada de la capa anterior y otro que aplica la entrada de datos que corresponden al vector de salida anterior (Torres, 2018). La expresión matemática que corresponde a esto es: $Y_T = f(W_{Xt} + U_{t-1} + b)$.

2.5. Mecanismos de Atención

Supongamos que nos encontramos leyendo una frase como: ***"Las olas del mar elevaban majestuosamente su espuma blanca"***, esta frase es fácil de entender para nosotros, ya que podemos ver el contexto y ver las relaciones de la oración, ya que lo leemos de manera secuencial, grosso modo así lo hacen las Redes Neuronales Recurrentes abreviadas como RNR o RNN por sus siglás en inglés *(Recurrent Neural Networks)*.

Tratar de recordar la primera palabra de esta tesis sería complicado a menos que regresáramos a leerla, ese es el principal problema con las RNR, ya que al nutrir la entrada con la salida anterior durante algunas ocasiones, los pesos que tienen durante el entrenamiento, las primeras palabras respecto a las últimas es menor, es decir, va "olvidando" las primeras palabras y provocará que la red no encuentre relaciones interesantes. Para ver de manera más clara esto, podemos regresar

al ejemplo de la frase *"Las olas del mar elevaban majestuosamente su espuma blanca"*, sería difícil para la red encontrar la relación entre la frase, *las olas* y la palabra *su*, que hace referencia a la pertenencia de nuestro sujeto, que en este caso es **las olas**. Para esto, necesitamos poner ***atención***, de manera literal.

El objetivo es buscar una solución a las faltas de relaciones de las diferentes palabras que se visualizan distanciadas, pero que tienen una relación muy cercana, es decir, el objetivo es encontrar la relación de todas contra todas. La solución a esto se llaman *Mecanismos de Atención* (Bahdanau et al., 2015).

Las primeras redes en utilizar este modelo fueron las llamadas LSTM por sus siglas en inglés *(Long short-term memory)* de manera bidireccional, usadas para generar secuencias de comentarios $(h_1, h_2, h_3...h_{x^T})$, esto para cada entrada de la sentencia (Bahdanau et al., 2015). Todos los vectores $h_1, h_2...h_{x^T}$, utilizados, son una concatenación de manera bidireccional de las capas ocultas del codificador, esto está dado por: $h_j = [\overrightarrow{h}_{j^T}; \overleftarrow{h}_{j^T}]^T$

Esta representación bidireccional se representa de manera gráfica en el diagrama **??**

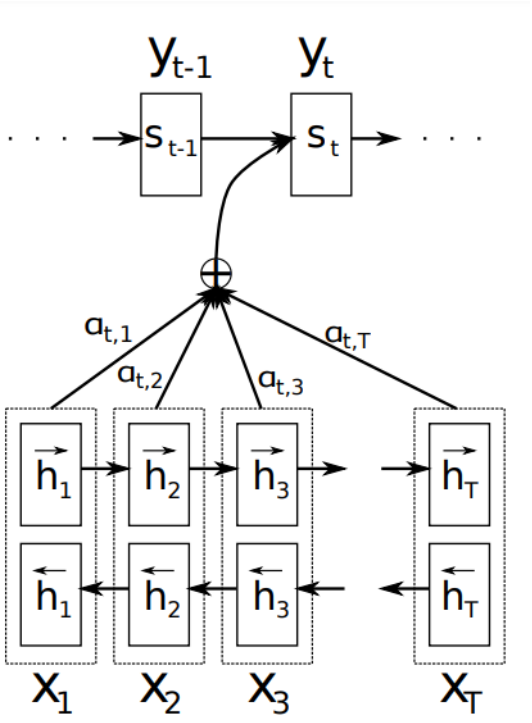


Figura 2.5.1: Diagrama del *modelo mecanismos de atención* (Bahdanau et al., 2015).

En términos simples, todos los vectores desde *h1 a hTx* se representan por Tx, mediante el número de palabras de la oración de entrada. Bahdanau et al. (2015) pone especial enfasis en los *embeddings* de todas la oración para crear el vector de contexto, esto a través de la suma ponderada de los estados ocultos (Bahdanau et al., 2015). En este punto el cálculo del vector (c_i) para la palabra (y_i) es generada a través de la suma ponderada de cada notación, esto expresado mediante:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Los pesos de α_{ij} están realizados mediante una función softmax dada por:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{j=1}^{T_x} exp(e_{ik})}$$

En este caso e_{ij} es el puntaje de salida, la cual intenta alinear j e i, y se describe por la función:

$$e_{ij} = a(s_{i-1}, h_j)$$

Todo este proceso se realiza para calcular a qué parte de la frase se le está poniendo mayor "atención", qué tan fuerte es la relación de una palabra a otra. Esto puede ser graficado y podemos visualizarlo mediante una matriz de atención. La figura 2.5.2 muestran a qué parte de la información prestan atención las entradas, es decir, las relaciones. En este ejemplo de traducciones, podemos ver que aunque las palabras que son similares no están tan cercanas, cuando se presenta una fuerte relación el color será más intenso.

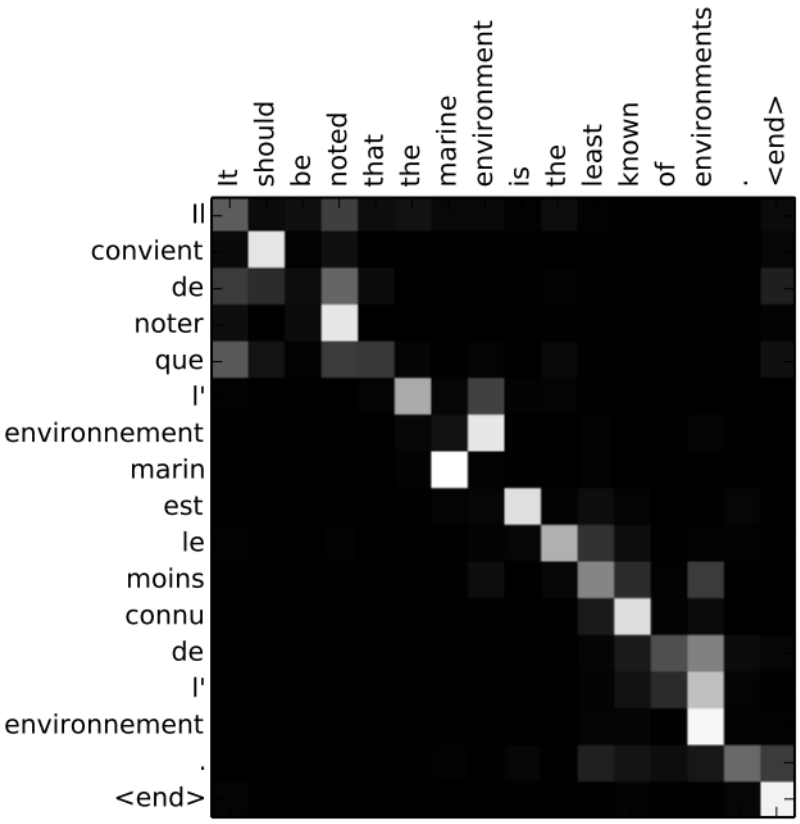


Figura 2.5.2: Matriz de atención de una traducción (Bahdanau et al., 2015).

2.5.1. Transformers

Aunque la palabra *transformer* puede sonarnos a un robot o transportarnos a una película de ciencia ficción, la realidad es que se trata de un modelo de DL realizado por Vaswani et al. (2017) a través de su artículo "Attention Is All You Need", nos propone un nuevo modelo de codificador y decodificador (en inglés encoder y decoder), basado en el principio de los mecanismos de atención. La estructura principal de este modelo se visualiza con la figura 2.5.3.

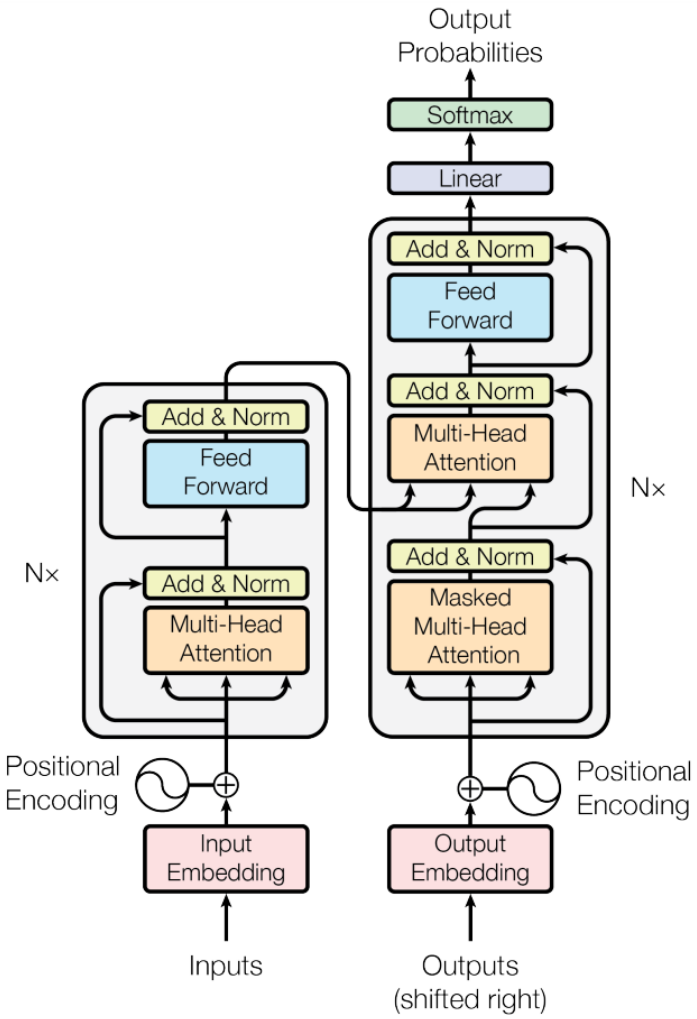


Figura 2.5.3: Modelo de Transformer (Vaswani et al., 2017).

Embedding

Las palabras ingresadas dentro del sistema deben tener una representación vectorial, es decir, una transformación numérica de las palabras y que incluyan un sentido semántico para que la ubicación dentro del espacio vectorial sea completa. Esta parte está representada en la parte verde de la imagen 2.5.4.

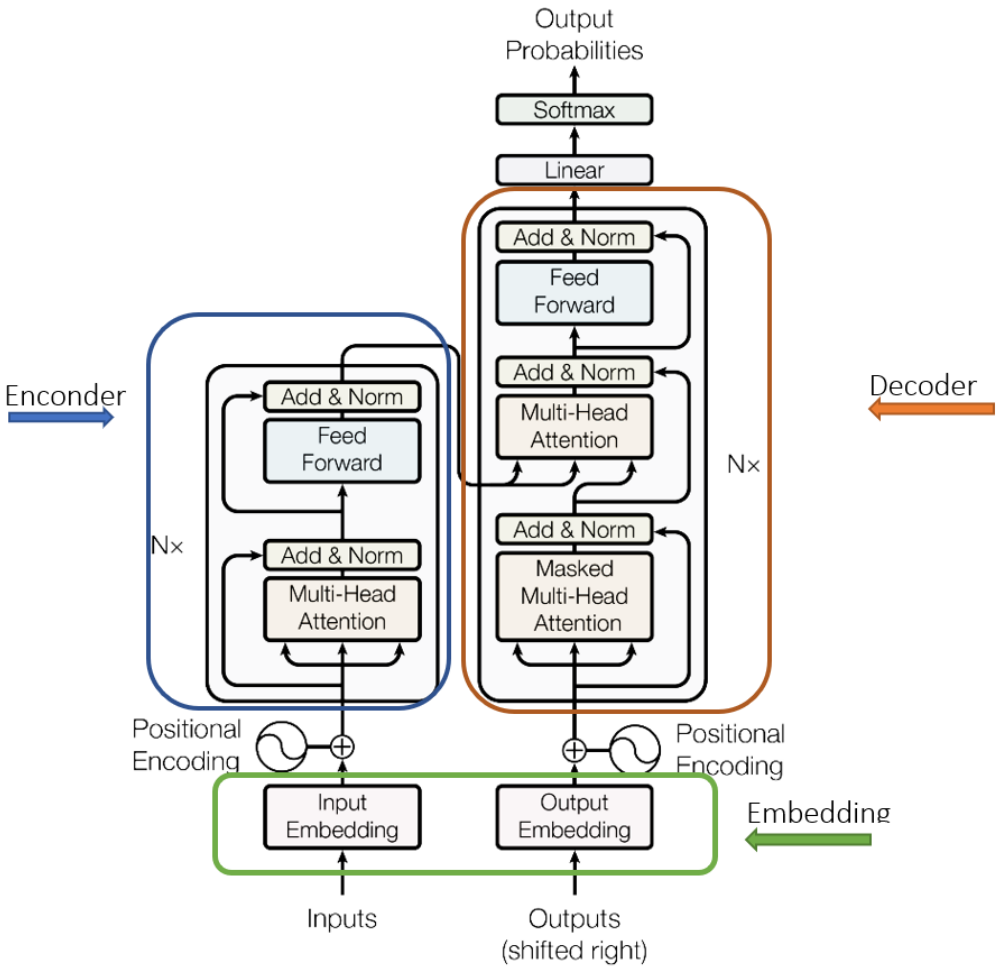


Figura 2.5.4: Partes del modelo transformer (Vaswani et al., 2017).

Encoder y Decoder

Para comprender mejor esto vamos a visualizarlo de manera gráfica, en la figura 2.5.4, el modelo de *Transformer* se compone de un ***Encoder*** (ubicado de color azul dentro de la imagen 2.5.4) y un ***Decoder*** (ubicado de color naranja dentro de la imagen 2.5.4):

Encoder: es el encargado de analizar la secuencia de entrada, está compuesto por 6 capas idénticas, cada una incluye dos subcapas: La primera es un mecanismo de auto atención de tipo multicabezal; La segunda, una alimentación directa en función de la posición. Se emplea una conexión residual entre cada dos subcapas, seguida de una capa de normalización. Esta será la salida de cada subcapa (en inglés *sublayer*) y será principalmente dada por la capa de normalización o *Layernorm* en inglés sea: $Layernorm = Layernorm(x + Sublayer(x))$, donde la $Sublayer(x)$ es la función implementada por la subcapa de sí misma (Vaswani et al., 2017). Todo esto se ve

reducido a capas de tipo *embeddings*, que producen salidas de $d_{model} = 512$

Decoder: Generará la secuencia de salida, enlazado directamente a la entrada, se compone de 6 capas idénticas, cada una incluye tres subcapas. Las primeras dos capas realizan la misma función que las encontradas en el *encoder*, pero la tercera se tratará de realizar mecanismo de auto atención de tipo multicabezal hacia la salida del *encoder*. Similar al *encoder* se emplea una capa de normalización. Se añade una modificación de auto atención a en las subcapas, para prevenir una atención a palabras en posiciones menores (Vaswani et al., 2017).

Atención

Tal como se menciona en *"Attention is all you need"* en español, *Atención es todo lo que necesitas* (Vaswani et al., 2017), la función de atención se describe como un mapa de vector key y vector query de tamaño d_k y valores de dimensión d_v , el cual el vector key probará todas sus "llaves" en cada "candado" y calculará la compatibilidad con cada palabra con ayuda del producto escalar, $\sqrt{d_k}$ y aplicando una función softmax. Entre mayor sea el resultado, mayor compatibilidad tendrán las palabras (de manera contextual), con esto podremos visualizar a qué parte de la frase presta atención y dará como resultado un vector de atención. Este proceso se puede ver en la figura 2.5.5.

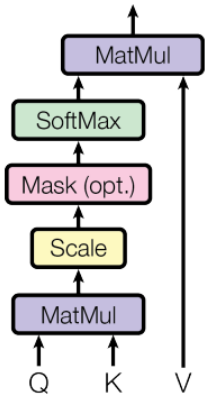


Figura 2.5.5: Producto escalar de atención (Vaswani et al., 2017).

Las llaves (*keys*) y valores son agrupados de manera conjunta en matrices K y V, con la función:

$$\textit{Attention}(Q, K, V) = \textit{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Tabla 2.5.1: Valores de Atención (Vaswani et al., 2017).

Símbolo	Descripción
Q	Petición (*Query*) del vector de la palabra.
K	Las llaves de todas las otras palabras de la secuencia.
V	Valor vectorial de la palabra que se está provocando

Atención por multicabezal:

La cantidad de cabezas o head del mecanismo será $h = 8$, esto permitirá que cada cabeza trabaje en aspectos diferentes y concatenar los resultados al final, esto hará que la propia palabra no sea la dominante del contexto si no todas. Esto lo podemos ver en la figura 2.5.6 y se define con la siguiente ecuación:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W_O$$

Donde:

$$head_i = AttentionQW_i^Q, KW_i^K, VW_i^V$$

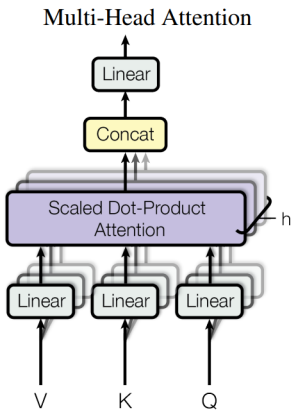


Figura 2.5.6: Atención por Multicabezal (Vaswani et al., 2017).

2.5.2. BERT

BERT son las siglas en inglés de Bidirectional Encoder Representations from transformers y fue desarrollado en octubre del 2018 en el artículo *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (Devlin et al., 2018).

Las búsquedas de Google, *grosso modo*, funcionaban sobre una tokenización de palabras, es decir, al buscar *"comprar juguetes de perros baratos"*, google analizaba estos términos y hacía búsqueda sobre esos índices. Ahora con BERT **el contexto es la clave.**

El nacimiento de BERT causó un gran impacto dentro de la comunidad científica, ya que la mejora de la comprensión de lenguajes e implanta un entrenamiento bidireccional de *transformer*. Con este cambio, los modelos de análisis de secuencia de texto de izquierda a derecha o derecha a izquierda quedarán a un lado. Devlin et al. (2018) demuestra que las frases analizadas de manera bidireccional tiene mejores resultados en el contexto y el flujo del lenguaje.

Como se mencionó, el contexto dentro de una oración son los elementos lingüísticos que dan sentido a la oración o a cada palabra, esto con el fin de determinar el sentido correcto de las palabras. Con todo esto nos vuelve a demostrar, que la atención es la clave para todo (Vaswani et al., 2017) y que puede ser usada para múltiples tareas como la clasificación, pregunta–respuesta, NER, etc.

¿Cómo trabaja?

El mecanismo de atención para el aprendizaje contextual y las relaciones entre palabras en una frase. El modelo incluye dos mecanismos de separación, un *encoder* que lee los textos de entrada y un *decoder* que se encarga de hacer las predicciones de la tarea, pero BERT solo utiliza el *encoder*.

La sección del *encoder* lee completamente la secuencia de entrada, lo que hace que aprenda su contexto completo (es por eso que se dice que el modelo es bidireccional). Se realiza un *embedding* de las palabras en la frase para después ser procesada por la red neuronal. La salida obtenida será un vector denominado como H, en la que cada vector corresponde a un token de entrada. BERT utiliza dos estrategias de entrenamiento:

Entrenamiento de las entradas

La entrada necesaria para BERT debe ser un par de sentencias (A y B), en el ejemplo visualizado en la figura 2.5.7 tenemos dos frases: A, *my dog is cute*; B, *he likes play*. La frase A y B son preprocesadas usando el tokenizador *wordpiece* (Ver sección 2.1.2), para convertir las frases en tokens. Para que el vector sea delimitado y el vocabulario no tome varias formas, el *tokenizador*

añade *"##ing"* al final de la oración. En la figura 2.5.7, la parte rosa representa esta primera parte.

Después, para que el modelo pueda diferenciar entre una frase y otra, al principio de la primera frase se inserta un token llamado [CLS] y en medio de ellas un token [SEP] al final de cada frase. Este proceso se puede visualizar en la figura 2.5.7.

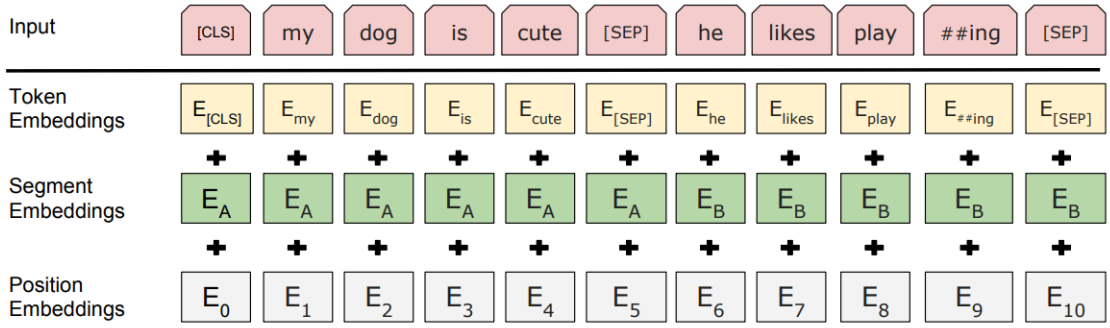


Figura 2.5.7: Representación de la entrada pasando por los tokens *embeddings*, segmentación y posicionamiento de *embeddings*, (Devlin et al., 2018).

La sección de *token embeddings* dentro de la figura 2.5.7, obtendrá una matriz de tamaño 30000 x 768(H), este sera el tamaño del vocabulario después del proceso *wordpiece* (Montanes, 2021). La figura 2.5.8 muestra este procedimiento.

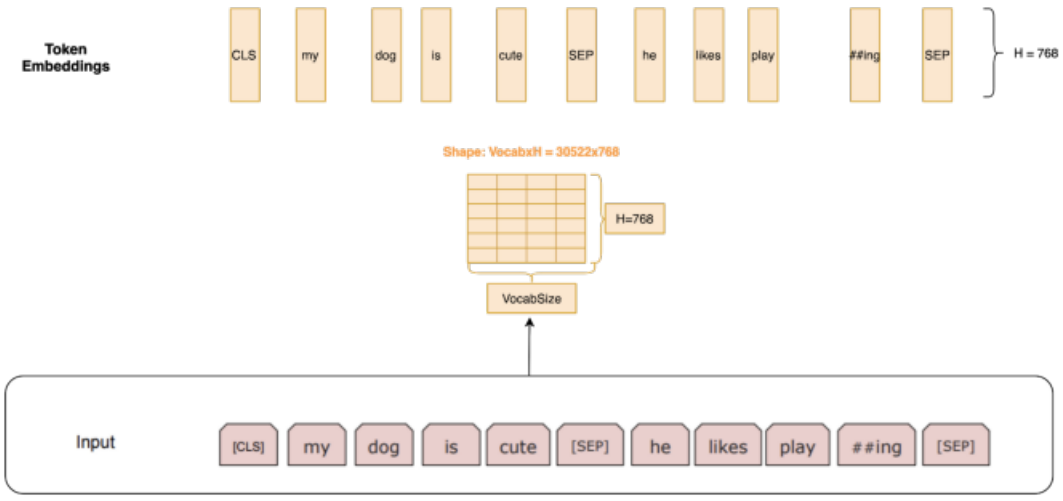


Figura 2.5.8: *Token embeddings* obtenidas en la matriz de tamaño *vocabulario*$\times H$ (Montanes, 2021).

La sección de *segment embedding* dentro de la figura 2.5.7, para una tarea de pregunta respuesta, se debe especificar la parte proveniente de la frase (Montanes, 2021), estos serán todos los vectores 0 de longitud H, si el *token* es de la frase 1 o es de la oración 2 (Montanes, 2021). La figura 2.5.9 representa esta sección.

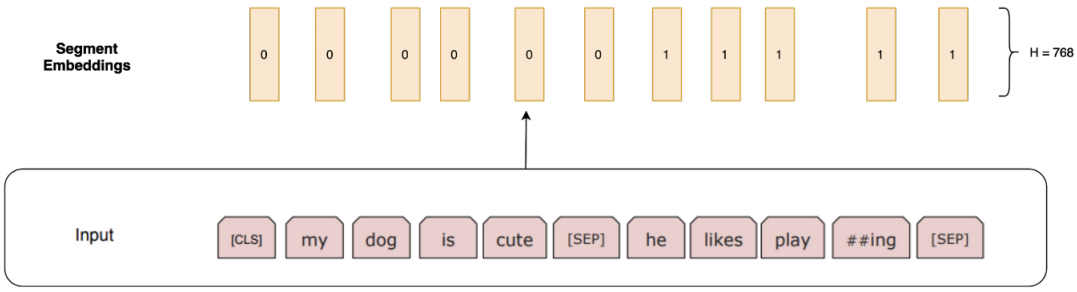


Figura 2.5.9: *Segment embedding* están todos en 0 o todos en 1, especificando si la frase pertenece a la oración 1 o a la oración 2 (Montanes, 2021).

La sección de *position embeddings* dentro de la figura 2.5.7, estos *embeddings* se usan para dar la posición de las palabras en la secuencia total (Montanes, 2021). Esto a través de una matriz con 768 columnas, la primera fila de esta matriz es el *embedding* del token [CLS], la segunda fila es la palabra "my", la tercera fila la palabra "dog" y así, de manera sucesiva.

Al final, la entrada que se dará a BERT será Token Embeddings + Segment Embeddings + Position Embeddings (Montanes, 2021) .

Modelo de lenguaje enmascarado

Modelo de lenguaje en Modelo de lenguaje enmascarado (MLM por sus siglas en inglés), es el proceso más interesante dentro de BERT, elige el 15 % de los tokens al azar. Si el i-ésimo *token* es elegido, se reemplaza con el token [MASK], el 80 % de las veces un token aleatorio, el 10 % el i-ésimo *token* sin cambios (Montanes, 2021).

"Entonces, si se tiene una secuencia de longitud 500, se enmascaran 75 *tokens* (15 % de 500), y de esos 75 *tokens*, 7 *tokens* (10 % de 75) serían reemplazados por palabras aleatorias y 7 *tokens* (10 % de 75) se utilizarán tal cual" (Montanes, 2021). Es decir, se reemplazan algunas palabras enmascaradas con palabras aleatorias y algunas palabras con reales.

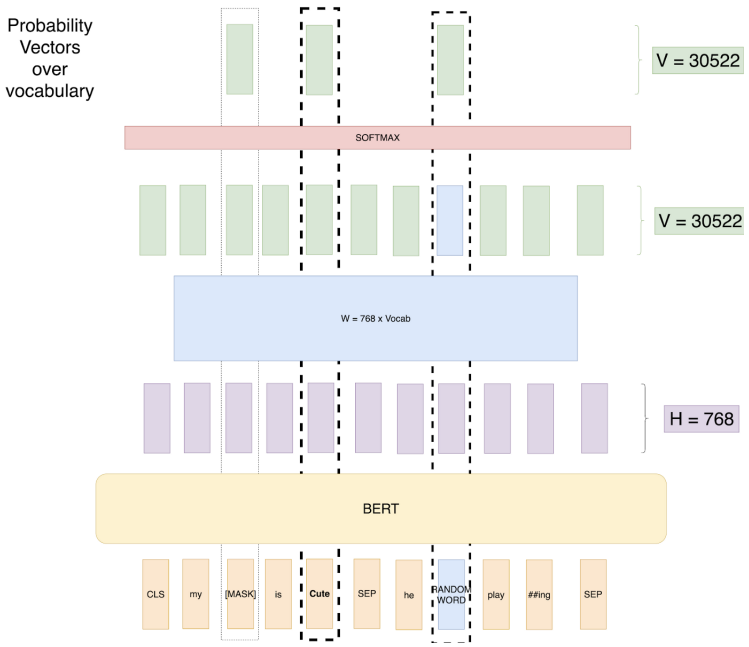


Figura 2.5.10: MLM enmascaramiento de palabras de BERT (Montanes, 2021).

Fine Tuning

Ahora, se puede elegir la tarea que se quiere realizar con BERT. Dentro del artículo original, Devlin et al. (2018) nos muestra diferentes tareas que puede realizar 2.5.11: Clasificación de pares de preguntas, respuesta a preguntas, tareas de etiquetado de una sola frase y para nuestros fines decir si una frase es parecida a otra a través de las técnicas de parafraseo.

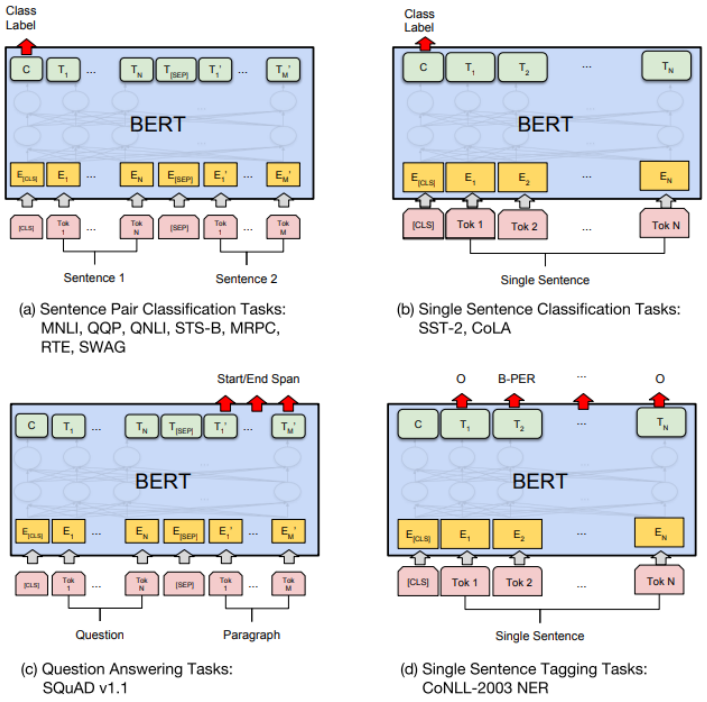


Figura 2.5.11: Ilustración de *Fine-tuning* de BERT en diferentes tareas (Devlin et al., 2018).

Dentro del *fine-tuning*, gran parte de los hiperparámetros permanecerán igual que en el entrenamiento y dentro del artículo guiará cuáles requieren ajuste. En la figura 2.5.12 se puede visualizar el *fine tuning* que la tarea Q&A (respuesta a preguntas), a través de este se definen dos vectores (S y E) con la forma de 1×768. Después, se realiza un producto escalar de estos vectores con los de salida de la segunda oración de BERT (Montanes, 2021). Posteriormente, se aplica la función *softmax* para obtener las probabilidades. Matemáticamente, el objetivo es la sumatoria de estas probabilidades en la posición inicial y final.

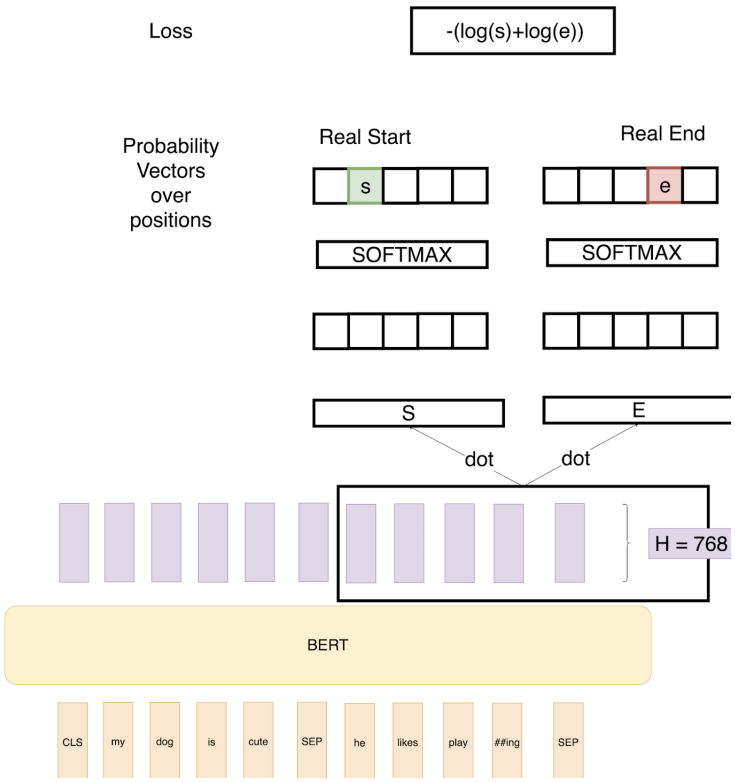


Figura 2.5.12: *Fine-tuning para la tarea de respuesta a preguntas*(Montanes, 2021).

2.5.3. Diferentes BERT

BERT posee un amplio espectro de configuraciones. Devlin (2020) nos proporciona diferentes configuraciones, las cuales poseen las características de BERT base y entrenados bajo el mismo régimen que el original, pero que pueden ser utilizadas en distintas tareas e infrestructuras. La figura 2.5.13 muestra las distintas configuraciones de BERT.

	H=128	H=256	H=512	H=768
L=2	**2/128 (BERT-Tiny)**	2/256	2/512	2/768
L=4	4/128	**4/256 (BERT-Mini)**	**4/512 (BERT-Small)**	4/768
L=6	6/128	6/256	6/512	6/768
L=8	8/128	8/256	**8/512 (BERT-Medium)**	8/768
L=10	10/128	10/256	10/512	10/768
L=12	12/128	12/256	12/512	**12/768 (BERT-Base)**

Figura 2.5.13: *Configuraciones de BERT* (Devlin, 2020).

SmallBERT 4/512

SmallBERT tiene 4 capas ocultas y un *embedding* de 512. Además, fue probado para GLUE (General Language Understanding Evaluation), obteniendo buenos resultados incluso en la tarea de QQP, con menos poder de procesamiento que sus homólogos. La figura 2.5.14 muestra los resultados para esta evaluación.

Model	Score	CoLA	SST-2	MRPC	STS-B	QQP
BERT-Tiny	64.2	0.0	83.2	81.1/71.1	74.3/73.6	62.2/83.4
BERT-Mini	65.8	0.0	85.9	81.1/71.8	75.4/73.3	66.4/86.2
BERT-Small	71.2	27.8	89.7	83.4/76.2	78.8/77.0	68.1/87.0
BERT-Medium	73.5	38.0	89.6	86.6/81.6	80.4/78.4	69.6/87.9

Figura 2.5.14: *Extracto de resultados de GLUE para cofiguraciones más pequeñas de BERT* (Devlin, 2020).

2.6. Parafraseo

La paráfrasis es un mecanismo lingüístico para la expresión de una idea de forma diferente, pero con el mismo significado del contenido, teniendo como objetivo conservar la idea de lo que se transmite originalmente. El parafraseo requiere un proceso de comprensión sintáctico y semántico, lo que hace que se convierta en un tema de interés para el PLN, ya que el parafraseo automático de un texto puede ser utilizado para múltiples aplicaciones, como la búsqueda de respuestas, el resumen automático, el análisis de plagio y la traducción automática (Flores, 2014b).

Flores (2014b) nos proporciona algunos ejemplos de paráfrasis:

- **Original.** "La finalidad del arte es dar cuerpo a la esencia secreta de las cosas, no el copiar su apariencia" decía Aristóteles.
- **Paráfrasis.** Según Aristóteles, el arte tiene la misión de encarnar la esencia oculta de la realidad, en vez de simplemente copiar su apariencia.

2.6.1. Técnicas de Paráfrasis

Técnica 1: Cambiar la función gramatical de algunas palabras.

Original: El profesor de medicina John Swanson plantea que los cambios globales influyen en la propagación de las enfermedades.

Paráfrasis: De acuerdo a John Swanson, un profesor de medicina, los cambios alrededor del globo causan que las enfermedades se propaguen (James, 2004).

Técnica 2: Uso de sinónimos.

Original: Un portavoz del gobierno norteamericano declaró que la crisis del SIDA...

Paráfrasis: Un portavoz del gobierno de los Estados Unidos anunció que el SIDA...

Técnica 3: Expresar números y porcentajes de formas diferente.

Original: Los grupos minoritarios de los Estados Unidos han sido golpeados con más fuerza por la epidemia. Los afroamericanos, que constituyen el 13 por ciento....

Paráfrasis: La epidemia del SIDA ha afectado principalmente a las minorías en los Estados Unidos, menos del 15

Técnica 4: Cambiar el orden de las palabras (e.g., cambiar de voz activa a pasiva y mover los modificadores a diferentes posiciones).

Original: Angier (2001) informó que la malaria mata a más de un millón de personas anualmente...

Paráfrasis: Cada año, más de un millón de personas muere a causa de la malaria (Angier, 2001)...

Técnica 5: Usar diferentes estructuras para las definiciones.

Original: La enfermedad de Lyme es una enfermedad inflamatoria causada por una bacteria que transmiten las garrapatas (pequeños arácnidos chupa sangre...)

Paráfrasis: La enfermedad de Lyme —una enfermedad que causa hinchazón y enrojecimiento— es ocasionada por una bacteria que transmite un pequeño arácnido conocido como garrapata...

Técnica 6: Usar diferentes indicadores de atribución.

Original: "Esta enfermedad pudo haber llego a nuestras fincas de diferentes formas", aseguró el veterinario Mark Walters en su reciente libro *Las seis plagas modernas*.

Paráfrasis: De acuerdo con Mark Walters (como se cita en Peterson, 2004), un veterinario que es autor del libro *Las seis plagas modernas*, la enfermedad pudo haber llegado de diferentes maneras a las fincas del país.

Técnica 7: Cambiar la estructura de la oración, y usar diferentes conectores.

Original: Aunque sólo cerca de una décima parte de la población mundial vive allí, el África subsahariana continúa siendo la región más golpeada...

Paráfrasis: Aproximadamente 10 % de la población mundial vive en el África subsahariana. Sin embargo, esta área del mundo tiene el porcentaje más alto de enfermedades relacionadas con el SIDA...

Técnica 8: No cambiar los términos clave ni los nombres propios.

Original: En el noreste de los Estados Unidos, las personas construyen sus hogares cerca de los bosques, donde las garrapatas portadoras de la enfermedad de Lyme se pegan a los ciervos. Además, en África, los cazadores traen de vuelta la carne de los animales que los científicos creen que pueden transmitir el Ébola, una enfermedad por lo general fatal que causa hemorragias masivas en sus víctimas.

Paráfrasis: En los Estados Unidos, las áreas residenciales se construyen cerca de las áreas boscosas en el noreste. Estas áreas son también los hogares de las garrapatas que portan la enfermedad de Lyme. Además, de acuerdo con los científicos, los cazadores en África matan los animales que pueden ser portadores de virus Ébola (un virus generalmente fatal que causa hemorragias masivas).

2.6.2. Análisis Exploratorio de Datos

No es fácil ver una pila de libros y decir cuál es el mejor, o mirar un conjunto de datos en una tabla de excel, nos cuesta entender de qué se trata, reconocer posibles patrones o reconocer distribuciones que podría ayudar.

Exploratory Data Analisys (EDA) es una herramienta útil para este tipo de análisis, ya que implica una exploración profunda de la información, y el objetivo es explorar y encontrar diferentes patrones en los datos para encontrar los modelos más adecuados (Camizuli and Carranza, 2018).

Los estudios EDA nacen de la estadística, desarrollada en 1977 por John Tukey, quien explicó que el objetivo de esta es "observar" los datos para ver cómo se ven Beyer (1981), para visualizar y extraer la información. en lo que es menos evidente a nuestros ojos. El núcleo de este enfoque proviene de las estadísticas descriptivas, respaldadas por herramientas gráficas para una mejor visualización.

EDA también nos permite visualizar la estructura de los datos e identificar diferentes patrones y valores atípicos (errores, peculiaridades o anomalías); Además, se buscan claves y/o pistas que puedan conducir a la identificación de patrones, ya que la exploración de las variables se realiza de forma diferente, primero una a una, luego dos a dos y luego sobre conjuntos más grandes, un proceso iterativo.

Hay algunas preguntas para entender esto:

- ¿Cuántos registros hay?
- ¿Existen distribuciones binarias?
- ¿Se puede reducir el conjunto de datos?
- ¿Es esta una tarea supervisada?
- ¿Cuál es la forma de los datos?
- ¿Está sesgada la distribución?
- ¿Los datos tienen patrones específicos?
- ¿Hay valores atípicos o puntos inusuales?
- ¿Los datos tienen datos temporales?

¿Qué obtenemos con la EDA?

Es una pregunta más que válida para este tipo de estudios, para datos crudos o procesados es posible obtener:

- Una aproximación a los datos de primera mano
- Saber si los datos son suficientes o no
- Identificar patrones de datos iniciales
- Si la variable en estudio es medible o no
- Distribuciones de tipo probabilístico

- Valores más frecuentes e infrecuentes
- Patrones usuales e inusuales

Para los datos ya procesados, se pueden obtener exactamente las mismas características, y se pueden crear características adicionales a partir de las ya observadas, por ejemplo, las distancias entre cada variable a razón de una similitud.

2.7. Técnicas de validación de modelos

Al compilar nuestro modelo existirá siempre una función de perdida y se debe verificar que el enfoque haya sido correcto.

2.8. Métricas de evaluación

Para determinar que los sistemas cumplen con las medidas de rendimiento y calidad aceptable, los modelos deben ser evaluados mediante métricas. Antes de realizar una evaluación, se debe considerar que los datos se encuentren equilibrados.

2.8.1. Matriz de confusión

Dentro de las tareas de aprendizaje máquina, una matriz de confusión o conocida también como matriz de errores (Stehman, 1997),permite visualizar el rendimiento de la tarea que se esta realizando. Cada columna de la matriz representara el número de predicciones hechas y permitirá ver el número y tipo de aciertos y errores en el modelo (Barrios, 2021). La matriz de confusión esta representada por la tabla 2.8.1

Tabla 2.8.1: Matriz de Confusión.

	Positive (PP)	Negative (PN)
Positive (P)	True Positive (TP)	False Negative (FN)
Negative (N)	False Positive (FP)	True Negative (TN)

2.8.2. Exactitud *(Accuracy)*

El *accuracy* es la medida más usada para determinar si un modelo es el mejor, en función de los datos de entrada del modelo. Este es declarado por:

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

Esta puede ser simplificada por:

$$Accuracy = \frac{TruePositive}{SizePositive}$$

Donde *TruePositives* hará referencia a los resultados positivos y *size of corpus* el número de textos totales del corpus.

2.8.3. Precisión

La precisión hace referencia a la proporción de los textos que realmente fueron clasificados correctamente de los textos que el sistema clasifico como positivos, es decir, mide qué tantos realmente positivos hay dentro de los positivos encontrados por el sistema. El cálculo de la precisión se expresa de la siguiente manera :

$$Precision = \frac{TruePositives}{(TruePositives + FalsePositives)}$$

2.8.4. Exhaustividad (Recall)

Dado un conjunto de resultados

La exhaustividad o *recall*, es el valor en forma de porcentaje que indica cuantos resultados fueron encontrados de forma correcta. Este calculo se realiza mediante la división de los datos positivos correctos entre la sumatoria de los positivos correctos más falsos negativos. El cálculo de la exhaustividad se expresa de la siguiente manera:

$$Recall = \frac{TruePositives}{(TruePositives + FalseNegative)}$$

2.8.5. Medida F1 (F_1 *score*)

La medida F_1, o F_1 *score* es una combinación de las métricas *Recall* y Precisión. Esta métrica es generalmente utilizada en problemas en los que el conjunto de datos se encuentra desbalanceado.

El cálculo del F_1 *score* se expresa de la siguiente manera:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{(Precision + Recall)}$$

2.9. Quora Question Pairs

Quora Question Pairs o QQP es un conjunto de datos compuesto por pares de preguntas. Dentro del conjunto de datos se pueden encontrar preguntas similares y otras que no lo son.

QQP consiste en más de 400,000 pares de preguntas separados en datos de entrenamiento y prueba. Dentro de los datos de entrenamiento, los pares de preguntas que son paráfrasis del otro son etiquetados de manera binaria con un 1, y si no lo son están etiquetados con un 0.

De acuerdo a Iyer et al. (2021), la inspiración para la creación de este conjunto de datos por Quora, es por la similitud de preguntas de las personas que visitan su sitio, además de múltiples preguntas con la misma intención, causan perdida de tiempo, además de no encontrar la mejor respuesta a la pregunta. Quora evalúa de manera canónica las preguntas porque ellas proveen mejores experiencias para los escritores y los que buscan.

El conjunto de datos proporciona la oportunidad de entrenar y probar modelos. La parte de entrenamiento en un formato .csv por 6 columnas: **ID**, identifica el par de preguntas; **qid1 y qid2**, contiene el identificador de cada una de las preguntas y va ligado con las columnas *question1* y *question2*; **question1 y question2**, contiene cada una de las preguntas; **is_ duplicate**,representa si la frase esta considerada como paráfrasis o no, esto a través de 1 o 0, si los textos en *question1* y *question2* tienen la misma intención se clasificará con 1, en caso contrario 0. En la tabla 5.1.10 se muestra un fragmento del conjunto de datos.

Tabla 2.9.1: Ejemplo del corpus de entrenamiento de QQP (Iyer et al., 2021).

id	qid1	qid2	question1	question2	is_duplicate[1]
0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
5	11	12	Why do rockets look white?	Why are rockets and boosters painted white?	1
1	3	4	How does a long distance (relationship work?	How are long distance relationships maintained?	1

La parte de pruebas o *test*, encuentra en un formato .csv por 3 columnas: **ID**, identifica el par de preguntas; **qid1 y qid2**, contiene el identificador de cada una de las preguntas y va ligado con las columnas *question1* y *question2*; **question1 y question2**, contiene cada una de las preguntas parafraseadas o no. En la tabla 2.9.2 se muestra un fragmento del conjunto de datos de pruebas.

Tabla 2.9.2: Ejemplo del conjunto de pruebas de QQP (Iyer et al., 2021).

test_id	question1	question2
0	How does the Surface Pro himself 4 compare with iPad Pro?	Why did Microsoft choose core m3 and not core i3 home Surface Pro 4?
1	Should I have a hair transplant at age 24? How much would it cost?	How much cost does hair transplant require?

Este conjunto de datos ha sido complementado desde su creación hasta el día de hoy, se han incluido más ejemplos negativos, ya que el conjunto de datos original incluía más combinaciones positivas y aunque hay algunos pares de preguntas similares, no son realmente sistemáticamente. Algunas preguntas contienen cierta cantidad de ruido y se han eliminado preguntas con detalles, es decir, preguntas extremadamente largas.

[1]QQP integra la columna *is_duplicate*, la cual no representa que sea un duplicado literal, sino la existencia o no de paráfrasis en los pares de pregunta de cada fila (0 para no parafraseados, 1 para parafraseados).

Capítulo 3

Estado del Arte

Entailment as Few-Shot Learner (EFL)

(Wang et al., 2021) Se centran en Entailment as Few-Shot Learner (EFL), que puede convertir pequeños Modelos del Lenguaje (ML) o Language Model (LM), para mejorar el aprendizaje en unos pocos activadores. La idea clave es reformular la tarea de Procesamiento de Lenguaje Natural (PNL), en una tarea de vinculación y refinarla con 8 ejemplos. Demuestran que el método mejora en un 12 % el aprendizaje profundo.

Charformer: Fast Character Transformers via Gradient-based Subword Tokenization

(Tay et al., 2021) Proponen un nuevo modelo de sesgo inductivo, que aprende una tokenización de subpalabras con el gradiente suave (GBST). El enfoque principal es el camino de los modelos sin token de alto rendimiento.

Transformer Likes Residual Attention

(He et al., 2020) Este documento se centra en RealFormer, una técnica simple y genérica para crear redes de transformadores en la capa de atención residual, que puede modificar transformadores como BERT con un amplio espectro de tareas.

FNet: Mixing Tokens with Fourier Transforms

(Lee-Thorp et al., 2021) A través de las capas de atención, realiza mezclas de tokens de entrada, las capas de avance demuestran ser competentes en el modelado de relaciones semánticas. las

capas demuestran ser competentes en el modelado semántico relaciones, esto a través de una transformada de Fourier estándar codificada a través de la transformada estándar.

data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language

(Baevski et al., 2022) Este modelo propone data2vec, un marco que utiliza el mismo método de aprendizaje para el habla. Esto ayuda a predecir objetivos específicos, en este caso palabras.

StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding

(Wang et al., 2019) Proponen una extensión de BERT a un nuevo modelo: Struct BERT, que propone estructuras lingüísticas en pre-entrenamiento, con dos tareas auxiliares para aprovechar al máximo el orden. En pre-entrenamiento, con dos tareas auxiliares para aprovecha al máximo el pedido.

Quora Question Pairs.

(Chen et al., 2018) La detección de preguntas duplicadas, utilizan una red neuronal, el modelo es el Manhattan LSTM (Long-Short Term Memory), a través de la fusión de dos redes neuronales se realiza la clasificación final, para finalmente evaluar el desempeño a través de la pérdida logarítmica entre los valores predichos y la realidad.

Identification of Duplication in Questions Posed on Knowledge Sharing Platform Quora using Machine Learning Techniques

(Rishickesh et al., 2019) A través de la extracción de funciones de contexto, como funciones de bolsa de palabras, Count-Vectorizer y TFIDF-Vectorize, puede realizar predicciones de similitud con métodos de modelado de aprendizaje automático como XGBoost y TFIDF.

Siamese Neural Networks with Random Forest for detecting duplicate question pairs

(Godbole et al., 2018) En este artículo utilizaron una adaptación siamesa de unidades recurrentes cerradas (GRU), en modo bidireccional, esto con una combinación de un Random Forest

Classifier, para el conjunto QQP.

What Do Questions Exactly Ask? MFAE: Duplicate Question Identification with Multi-Fusion Asking Emphasis

(Zhang et al., 2020) Utiliza un mecanismo transformador llamado BERT, esto ayudará a obtener las incrustaciones de las palabras de forma dinámica. Entonces provoca un énfasis ascendente de atención y autoatención. Cuando una palabra interactúa más con otra, más importante se vuelve. Finalmente, se utiliza una combinación de ocho vías para la identificación de preguntas.

Transfer Fine-Tuning: A BERT Case Study

(Arase and Tsujii, 2019) Proponen inyectar relaciones de paráfrasis de frase en BERT para generar representaciones adecuadas para la evaluación de equivalencia semántica en lugar de aumentar el tamaño del modelo. Los experimentos en tareas estándar de comprensión del lenguaje natural confirman que el método mejora efectivamente el modelo BERT base mientras mantiene el tamaño. El modelo generado exhibe un rendimiento superior en comparación con un modelo BERT más grande en tareas de evaluación de equivalencia semántica.

3.1. Comparación de esta propuesta con el estado del arte

El estudio de paráfrasis, es una tarea que se ha sido de gran interés aplicado al conjunto de datos QQP.

El estado del arte se ha centrado en el estudio de métodos modernos, haciendo que en algunos puntos de sus métodos, los algoritmos sean forzadas sin tener el conocimiento de los patrones o el comportamiento del conjunto de datos.

Nuestra propuesta se centrará en la obtención de los mejores patrones, tomando en cuenta la naturaleza de los datos, todo esto mediante el análisis exploratorio de datos profundo antes de la clasificación, el cual combinará los métodos clásicos, métodos modernos y diferentes combinaciones de características, pretendiendo ser más efectiva en cuanto a los resultados obtenidos, además de demostrar que se pueden obtener resultados similares con ambos métodos.

La comparación inicial se realizará con los resultados de esta tesis, para posteriormente ser comparados con el estado del arte.

Capítulo 4

Desarrollo y Solución

4.1. Definición del conjunto de datos

Características

Para esperar los mejores resultados en el entrenamiento, se debe realizar un análisis inicial de los datos "puros", es decir, sin procesar.

El conjunto de QQP está dividido desde su creación en dos partes; entrenamiento y pruebas.

Número de datos:

- **Entrenamiento**
    1. Cantidad de datos: 404,290 pares de preguntas
    2. Clases de los datos:
        a) **0:** Pares de preguntas que no son iguales.
        b) **1:** Pares de preguntas iguales
    3. Columnas
        a) **id:** Identificador de la fila
        b) **qid1:** Identificador de la pregunta 1.
        c) **qid2:** Identificador de la pregunta 2
        d) **question1:** Pregunta 1.

e) **question2:** Pregunta 2.

f) **is_duplicate:** Clase

- **Prueba**

1. Cantidad de datos: 2,345,796
2. Clases de los datos:

   a) **0:** Pares de preguntas que no son iguales.

   b) **1:** Pares de preguntas iguales
3. Características

   a) **id:** Identificador de la fila 2

   b) **question1:** Pregunta 1.

   c) **question2:** Pregunta 2.

Para este proyecto, únicamente trabajaremos con el conjunto de datos denominado **Entrenamiento**, ya que la cantidad de datos definida para QQP en la parte de **Pruebas** es superior a lo que puede ser soportado por nuestra infraestructura.

Derivado de lo anterior, nuestro conjunto de datos de Entrenamineto, será divido como:

Entrenamiento: 70 % de los datos.

Práctica: 30 % de los datos.

4.2. Análisis Exploratorio (Fase 1)

Se debe determinar si los datos encontrados dentro de la documentación son correctos.

Porcentaje de las clases: El **36.92 % (149,263)** de los pares de preguntas son paráfrasis (representadas con 1), mientras que el **63.08 % (255,027)** no lo son (representadas con 0). En la gráfica 4.2.1, se visualiza esa distribución de las clases.

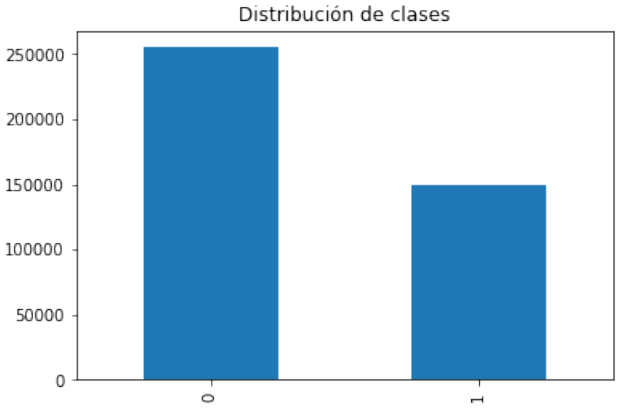


Figura 4.2.1: Clases de QQP.

Preguntas vs repetición: Los conjuntos de entrenamiento suelen tener pequeñas trampas para realizarlos, el conjunto de QQP no es la excepción.

Preguntas únicas: De un total de **808,580** solo **537,929** son únicas. **1** pregunta es repetida **157** veces (Pregunta: *How do I improve my English speaking?*), y ese es el número máximo de repetición. La gráfica 4.2.2, muestra esta distribución.

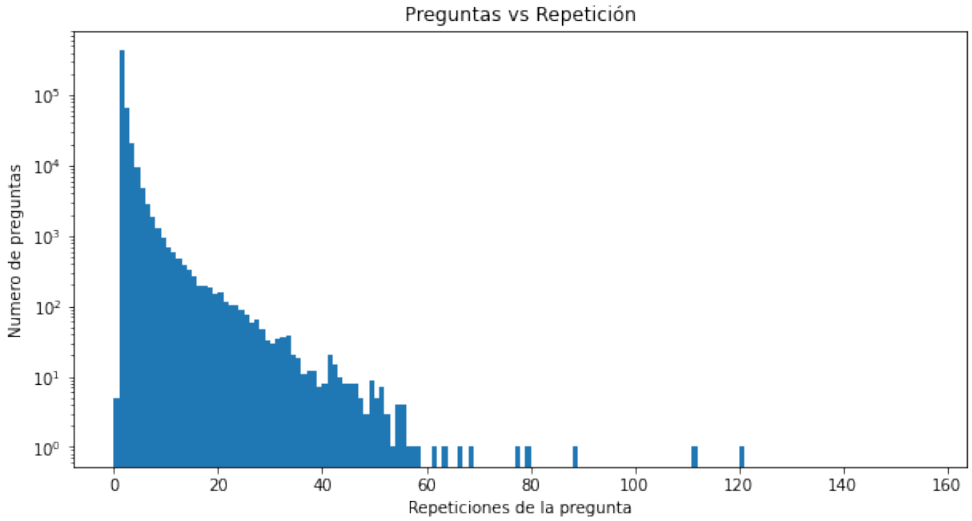


Figura 4.2.2: Número de preguntas repetidas dentro de QQP.

Datos incompletos: No todos los datos tienen las 6 columnas, por lo cual se deben eliminar aquellos datos que no cumplan con esto.

Preguntas con menos caracteres: Existen otro tipo de errores, como que no sea una pregunta, o solo un símbolo. Se encontró que en el par de pregunta con el, id 3306, una de ellas solo es un signo de punto, y la pregunta con el, id 47056, solo es un signo de pregunta cerrada. La tabla 4.2.1 muestra las este tipo de preguntas dentro del conjunto.

Tabla 4.2.1: Datos del conjunto de datos con pocos caracteres.

id	qid1	qid2	question1	question2	is_duplicate
3306	6553	6554	.	Why is Cornell's endowment the lowest in the Ivy League?	0
47056	84067	84068	Is there anywhere in the world ? offering pain management for peripheral neuropathy as opioids haved been banned in US	?	0

Por lo tanto, todos aquellos datos que contienen este tipo de errores serán eliminados de nuestro conjunto de datos. El conjunto final tendrá un tamaño de **404,287** pares de preguntas con un **porcentaje de las clases:** El **36.92 % (149,262)** de los pares de preguntas son duplicadas (representadas con 1), mientras que el **63.08 % (255,025)** no lo son (representadas con 0).

4.3. Preprocesamiento

Limpieza y Homogeneización de los datos

Para obtener los mejores resultados, se debe realizar un procesamiento donde los datos que son de igual significado, pero se expresan de manera diferente, puedan ser iguales:

- **Minúsculas:** Todas las letras se pasarán a minúsculas.
- **Remplazo**
  1. Cantidades numéricas por símbolos
  2. Contracciones
  3. Símbolos matemáticos por descripciones
  4. Divisas por descripciones
- **Hipervínculos:** Todos aquellos www, https, http, etc.
- **Lematización de los datos:** Una única representación a todas las variantes de una palabra.

4.4. Extracción de Características Manual

Como parte de la propuesta de este trabajo, derivado del análisis minucioso de la información presentada en el conjunto de preguntas QQP, se han creado diferentes características, a saber:

1. **Número de palabras de la pregunta 1:** Cuenta las palabras de la pregunta 1.
2. **Número de palabras de la pregunta 2:** Cuenta las palabras de la pregunta 2.
3. **Número de caracteres de la pregunta 1:** Cuenta los caracteres de la pregunta 1.
4. **Número de caracteres de la pregunta 2:** Cuenta los caracteres de la pregunta 2.
5. **Número total de palabras:** La suma entre la 1 y la 2
6. **Diferencia absoluta entre palabras:** Cuantas palabras tiene la pregunta 1 y la pregunta 2.
7. **Primera palabra igual:** Determina si la primera palabra de la pregunta 1 y la pregunta 2, son iguales
8. **Última pregunta igual:** Determina si la última palabra de la pregunta 1 y pregunta 2, es igual.
9. **Total de palabras únicas:** Cuenta el total de palabras únicas de la pregunta 1 y 2.
10. **Total de palabras únicas sin palabras de parada:** Cuenta el total de palabras únicas sin palabras de parada de la pregunta 1 y 2.
11. **Relación de número de palabras únicas totales:** Determina la relación que existe entre el número de palabras totales y las palabras únicas totales.
12. **Palabras comunes:** Cuenta el total de palabras comunes entre la pregunta 1 y la pregunta 2.
13. **Proporción de palabras comunes:** Igual al número de palabras comunes dividido por el número mínimo de palabras entre la pregunta 1 y la pregunta 2.
14. **Proporción de palabras comunes máximas:** La relación del número de palabras comunes dividido por el número máximo de palabras entre las preguntas.
15. **Proporción de palabras comunes mínimas:** La relación del número de palabras comunes dividido por el número mínimo de palabras entre las preguntas.
16. **Palabras comunes sin palabras de parada:** Cuenta las palabras, excluyendo las de parada.

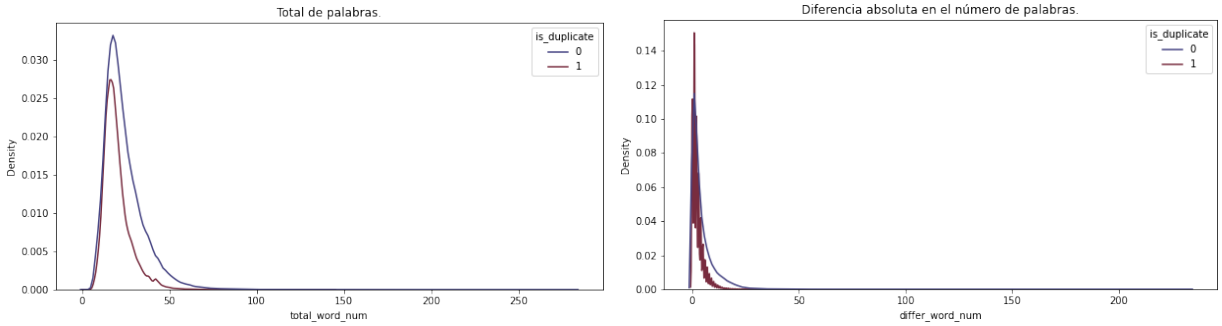
17. **Proporción de palabras comunes máximas sin palabras de parada:** La relación del número de palabras comunes sin palabras de parada dividido por el número máximo de palabras entre las preguntas.
18. **Proporción de palabras comunes mínimas sin palabras de parada:** La relación del número de palabras comunes sin palabras de parada dividido por el número mínimo de palabras entre las preguntas.
19. **Proporción de palabras comunes sin palabras de parada:** Es igual al número de palabras comunes sin palabras de parada, dividid por el número máximo de palabras entre la pregunta 1 y la pregunta 2, sin palabras de parada.
20. **Características *Fuzzywuzzy*:** Se trata de características de la librería *fuzzywuzzy*, con las cuales se extraen las de proporción: parcial, conjunto de tokens y comparación de preguntas (3 características).
21. **Similitudes coseno**: La cual mide la similitud existente entre dos vectores en un espacio (2 características).

4.5. Análisis Exploratorio de datos (Fase 2)

Se condujeron 3 tipos de gráficas de acuerdo con la característica analizada:

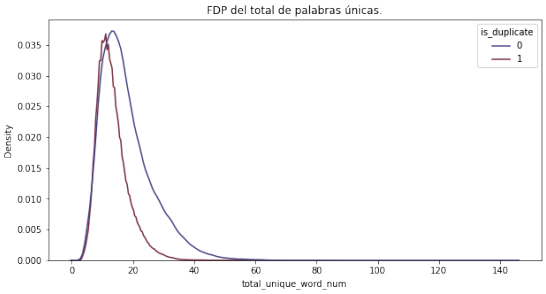
- Función de densidad de probabilidad
- Conteo de observaciones

Las gráficas 4.5.1, 4.5.2, 4.5.3, 4.5.4, 4.5.5, 4.5.6, 4.5.7, 4.5.8 y 4.5.9 y 4.5.10, son el resultado del análisis exploratorio de datos, a partir de las características extraídas manualmente.

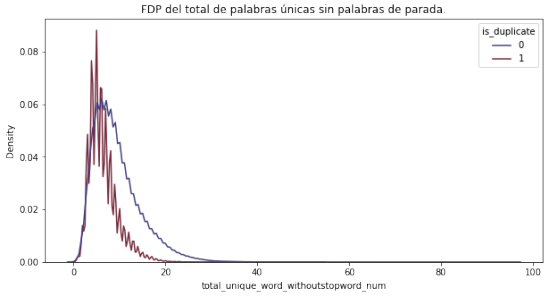


((a)) Número Total de Palabras en las preguntas ((b)) Diferencia Absoluta en el número de palabras

Figura 4.5.1: Gráficas del Análisis Exploratorio de Datos para Características Manuales 1

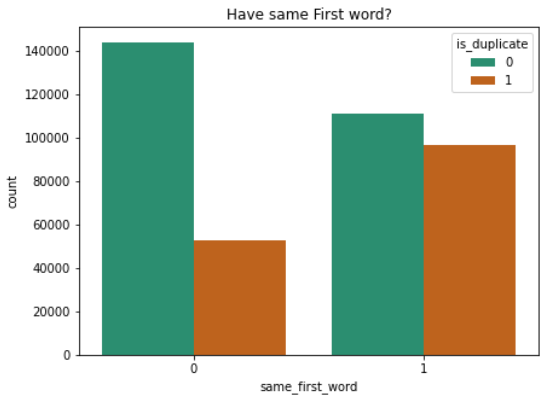


((a)) [Función de densidad de probabilidad del número total de palabras únicas en las preguntas duplicadas y sin duplicar.

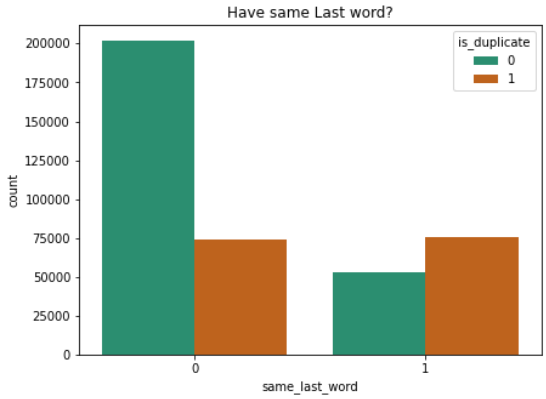


((b)) Función de densidad de probabilidad del número total de palabras únicas sin palabras de parada, en las preguntas duplicadas y sin duplicar.

Figura 4.5.3: Gráficas del Análisis Exploratorio de Datos para Características Manuales 3.

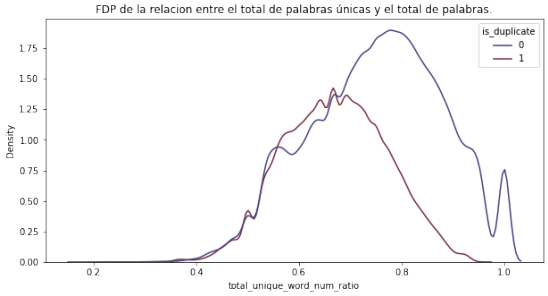


((a)) Similitud de la primera palabra entre la pregunta 1 y la pregunta 2

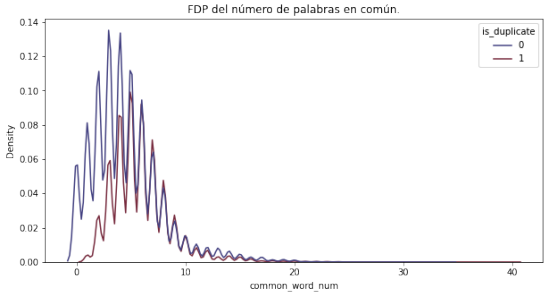


((b)) Similitud de la última palabra igual entre la pregunta 1 y la pregunta 2.

Figura 4.5.2: Gráficas del Análisis Exploratorio de Datos para Características Manuales 2.

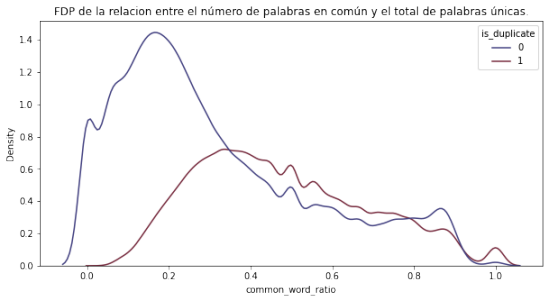


((a)) Función de densidad de probabilidad sobre la proporción existente entre el total de palabras unicas y el total de palabras generales, entre las preguntas duplicadas y sin duplicar.

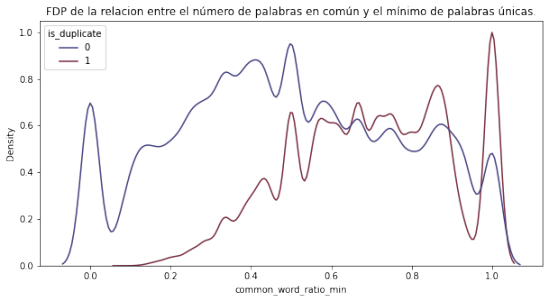


((b)) Función de densidad de probabilidad de palabras comunes entre la pregunta 1 y pregunta 2, clasificadas como duplicadas y sin duplicar.

Figura 4.5.4: Gráficas del Análisis Exploratorio de Datos para Características Manuales 4.

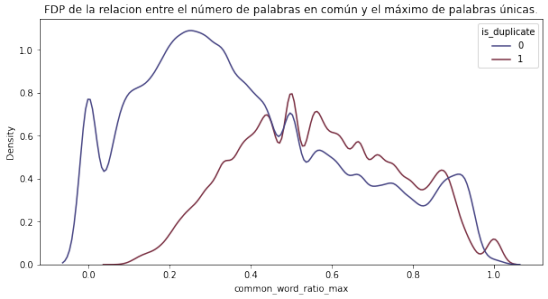


((a)) Función de densidad de probabilidad sobre la proporción existente entre el total de palabras en común y el total de palabras únicas, entre las preguntas duplicadas y sin duplicar.

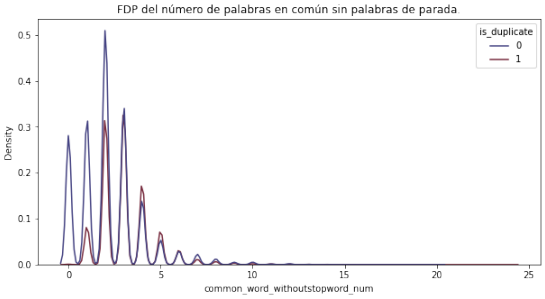


((b)) Función de densidad de probabilidad de la proporción que existe entre el número de palabras en común y el mínimo de palabras únicas, duplicadas y sin duplicar.

Figura 4.5.5: Gráficas del Análisis Exploratorio de Datos para Características Manuales 5.

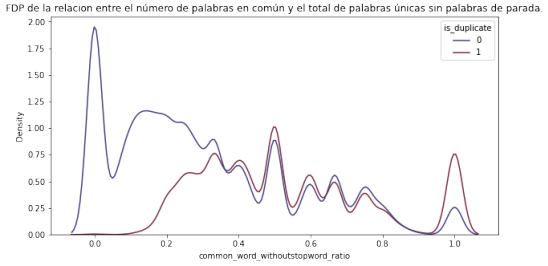


((a)) Función de densidad de probabilidad de la proporción que existe entre el número de palabras en común y el máximo de palabras únicas, duplicadas y sin duplicar.

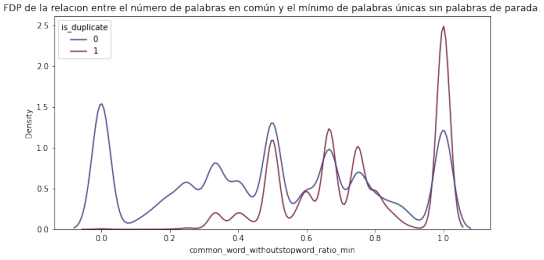


((b)) Función de densidad de probabilidad del número de palabras comunes sin palabras de parada

Figura 4.5.6: Gráficas del Análisis Exploratorio de Datos para Características Manuales 6.

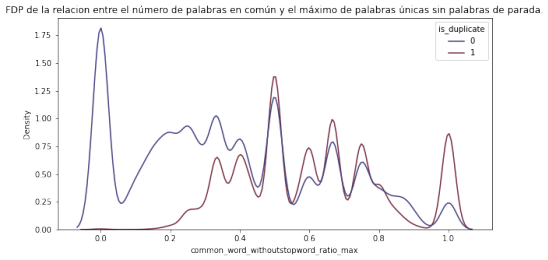


((a)) Proporción de No. de palabras comunes sobre el total de palabras únicas sin palabras de parada.

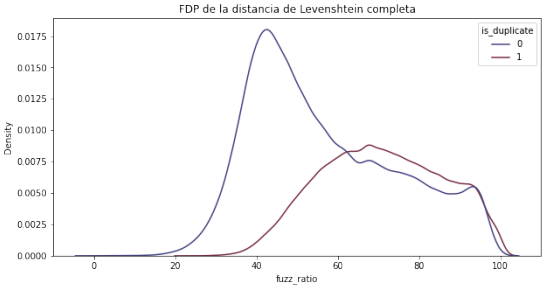


((b)) Relación entre las palabras comunes y las palabras únicas mínimas sin palabras de parada.

Figura 4.5.7: Gráficas del Análisis Exploratorio de Datos para Características Manuales 7.

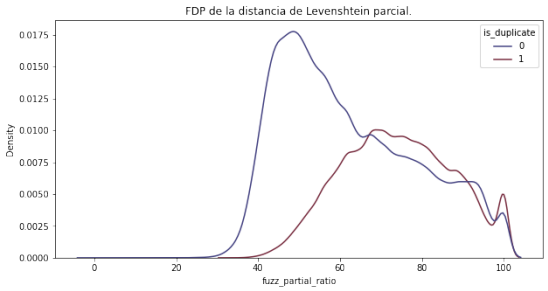


((a)) Relación entre el número de palabras comunes y el número máximo de palabras únicas sin palabras de parada..

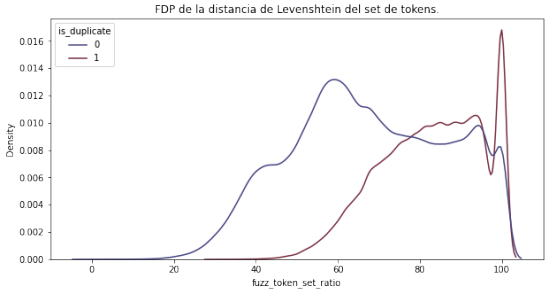


((b)) Proporción de la densidad de probabilidad de la proporción de la característica *fuzz* de la librería fuzzywuzy.

Figura 4.5.8: Gráficas del Análisis Exploratorio de Datos para Características Manuales 8.



((a)) Proporción de la densidad de probabilidad de la proporción parcial de la característica fuzz de la librería fuzzywuzys.



((b)) Proporción de la densidad de probabilidad de la proporción del conjunto de tokens de la característica fuzz de la librería fuzzywuzy

Figura 4.5.9: Gráficas del Análisis Exploratorio de Datos para Características Manuales 9.

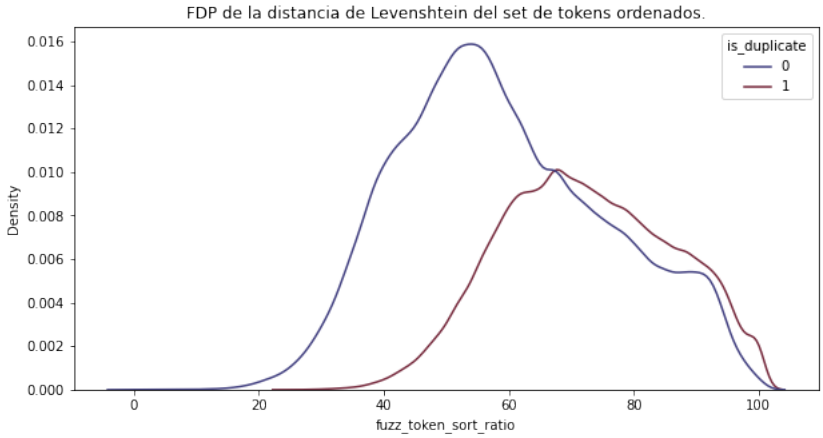


Figura 4.5.10: Proporción de la densidad de probabilidad del conjunto ordenado de la característica *fuzz* de la librería fuzzywuzy.

4.6. Extracción de características automáticas

Los *embeddings* son representaciones de palabra en forma de vectores, con generalmente entre 50 y 300 dimensiones. Estas representaciones resultan más precisas, pero no cuentan con un contexto establecido. SentenceBERT (SBERT) (Reimers and Gurevych, 2019) representa semánticamente cada oración. La principal característica de este embedding es que entrega un vector de 768 dimensiones por palabra.

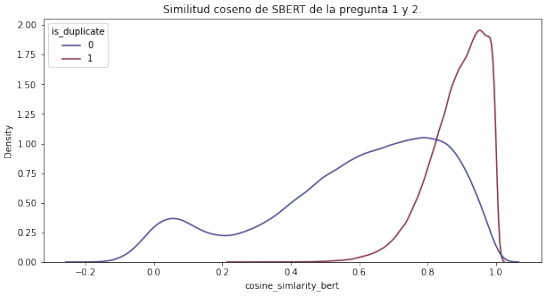
Dado lo anterior, se realizó la extracción de las características por cada pregunta, obteniendo un *embedding* 1536 características por cada fila (786 por cada pregunta).

Posteriormente, se calculó la distancia euclidiana y la similitud coseno que existe entre los resultados de la extracción de características automática de cada par de preguntas.

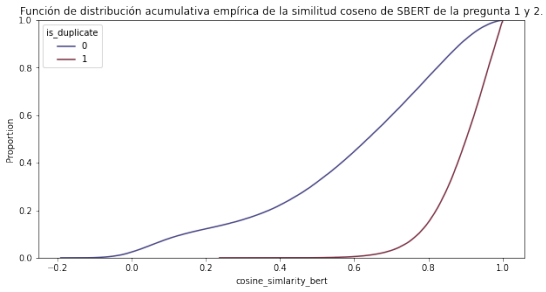
Por último, se añaden estas dos nuevas características al total de características manuales, dando un total de 25.

4.7. Análisis Exploratorio (Fase 3)

La tercera fase del análisis exploratorio de datos, nos conduce a analizar las dos nuevas características extraídas de manera automática por SBERT (Distancia Euclidiana y Similitud Coseno). Se hace el apartado de estas características, ya que miden la distancia entre cada par de palabras.

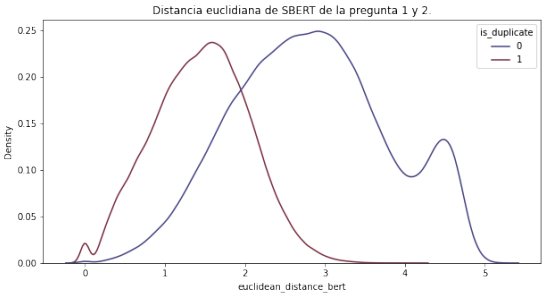


((a)) Similitud coseno entre la pregunta 1 y 2, extraída de manera automática

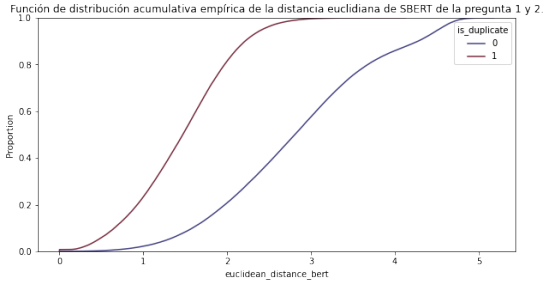


((b)) Función de distribución acumulativa empírica de la similitud coseno entre la pregunta 1 y 2, extraída de manera automática.

Figura 4.7.1: Gráficas del Análisis Exploratorio de Datos para Características Automáticas 1.



((a)) Distancia Euclidiana entre la pregunta 1 y 2, extraídas de manera automática.



((b)) Función de distribución acumulativa empírica de la similitud coseno entre la pregunta 1 y 2, extraída de manera automática

Figura 4.7.2: Gráficas del Análisis Exploratorio de Datos para Características Automáticas 2.

4.8. Suma de Características

Antes de realizar el PCA se realiza una transformación llamada *Min-Max Scaler*, la cual se trata de una transformación para convertir los valores de las características a valores entre 0 y 1.

Posteriormente, las características resultantes de la extracción automática de características al conjunto de características manuales (25) para dar un total de **1561 características.**

Orden de Características:
$1561 = 25$ características manuales + 1536 automáticas

4.9. Análisis de Componentes Principales (PCA)

Se podría pensar que el preprocesamiento estaría completo con todo el análisis anterior, pero para aumentar el valor de la propuesta se incluyeron 2 pasos más.

Uno de estos pasos es el PCA, que nos permitirá reconocer cuáles son las características que más impactan a los datos y cuáles son las que cubren el mayor porcentaje de la variabilidad de los datos.

Para realizar este análisis, es de vital importancia graficar la proporción de varianza explicada que mostrará de manera explícita como es que mientras más características se posean, más cerca se está de cubrir el 100 % de la variabilidad de los datos.

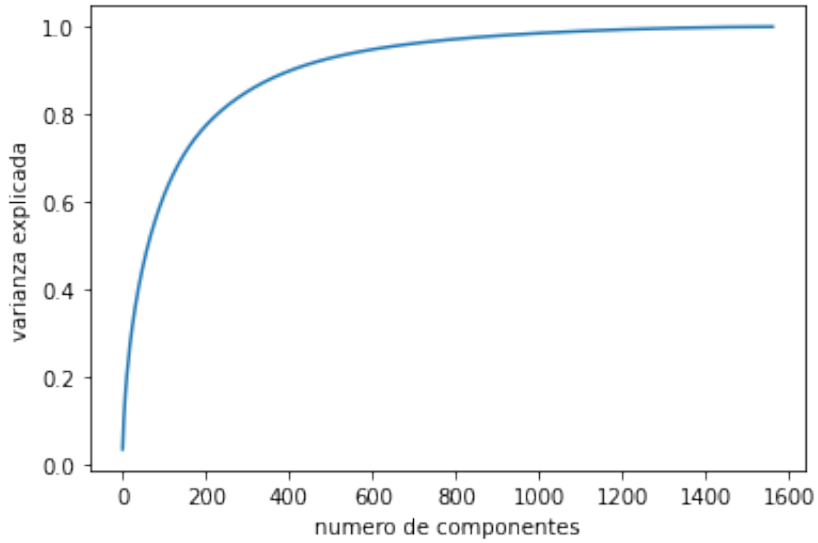


Figura 4.9.1: Importancia de las, características de acuerdo a PCA.

En la gráfica 4.9.1 podemos observar que las primeras 800 características cubren el 97 % de la variabilidad de los datos, por lo que las características subsecuentes no impactan significativamente para la tarea, por lo que se pueden eliminar, optimizando así el uso de memoria RAM y tiempo de procesamiento.

4.10. Balanceo de los Datos

Es importante confirmar que el conjunto de datos se encuentre realmente desbalanceado, para eso utilizaremos la fórmula de *imbalance ratio*:

$$Desbalanceado = \frac{NoDatos_{Mayoritaria}}{NoDatos_{Minoritaria}}$$

Sustituimos:

$$Desbalanceado = \frac{149,262}{255,025} = 0.58$$

El límite para definir que un conjunto de datos es balanceado es de 0.2, el resultado de nuestra probabilidad es *0.5* lo que indica que es un conjunto desbalanceado.

4.10.1. Técnica de balanceo de datos

Para este conjunto de datos se decidió utilizar la técnica de **Modificación del Conjunto de Datos**, pero con la técnica de reducción de datos, ya que con eso balanceamos y prevenimos el *overfitting*.

Lo que haremos es utilizar un algoritmo para reducir la clase mayoritaria, usando un algoritmo que elegirá de manera aleatoria información para eliminar basado en la proporción de datos y los parámetros asignados.

El conjunto balanceado tendrá un tamaño de **298,524** pares de preguntas con un **porcentaje de las clases:** El **50 % (149,262)** de los pares de preguntas son duplicadas (representadas con 1), mientras que el **50 % (149,262)** no lo son (representadas con 0).

4.11. Clasificación

Se utilizaron clasificadores tradicionales y modernos para comparar el alcance que tuvo este preprocesamiento y profundo análisis de los datos, así como tener un panorama mucho más amplio en cuanto a cómo impacta a las métricas de evaluación de los clasificadores.

4.11.1. Clasificadores Tradicionales

Random forest

Para elegir los mejores parámetros del modelo se utilizó una técnica que prueba diferentes combinaciones de parámetros, hasta encontrar el más óptimo de las opciones que se proporcionaron como argumento, dando como resultado los siguientes valores:

- max_depth = 50
- min_samples_split = 5
- n_estimators = 200

Con estos valores se procedió a entrenar el modelo, tomando como conjunto de entrenamiento el 70 % de los datos y reservando el 30 % para validación.

SVC

Para este modelo, se realizó un pre entrenamiento para optimizar el tiempo y el coste computacional que se requiere para este modelo. Posteriormente, se entrenó tomando como base el pre entrenamiento tomando a la función de costo llamada *hinge*.

XGBOOST

Se utilizaron dos configuraciones de parámetros que se diferencian por la cantidad de parámetros que se dejaron por defecto dentro del modelo.

Para este modelo, se utilizó el mismo método para buscar la mejor combinación de algunos parámetros, dando como resultado los siguientes valores, los cuales introduciremos como argumento para este primer modelo:

- learning_rate = 0.28
- max_depth = 4

Una vez introducidos estos valores, también se especificó que se trata de un problema con binario, un resultado donde se entregue cada 20 épocas el valor que emite la función de pérdida, así como la cantidad de épocas que se utilizaran.

Para la segunda configuración del modelo se tomaron los mismos valores que el primer modelo, con la diferencia que para este también se modificaron los parámetros alfa, lambda y la cantidad de submuestras por árbol.

4.11.2. Clasificadores Modernos

Los clasificadores modernos que se utilizaron fueron una Red Neuronal Recurrente (RNN) y el algoritmo Small BERT, cabe destacar que estos modelos solo se usaron como referencia, ya que poseen su propia arquitectura, *embeddings* y limpieza de datos.

El propósito es comparar arquitecturas modernas sin un análisis profundo de los datos contra arquitecturas tradicionales y medir qué tanto impactan los métodos propuestos en el preprocesamiento de los datos a las métricas de evaluación.

RNN

La arquitectura propuesta para esta solución comparte en buena parte técnica y métodos utilizados en los modelos tradicionales, ya que también se cuenta con:

- Una limpieza en los datos para eliminar contracciones y símbolos innecesarios
- La eliminación de palabras de parada
- Conversión a minúsculas

Posteriormente, se declara el *embedding* que se utilizara como extractor de características, en este caso se utilizó un extractor de 300 dimensiones basado en **GoogleNews**.

La arquitectura de red que se utilizó fue una RNN Manhattan LSTM donde se utiliza un arreglo de dos vectores como entrada de la red y a partir de las salidas de red siamesa, se calcula la distancia Manhattan de los vectores resultantes.

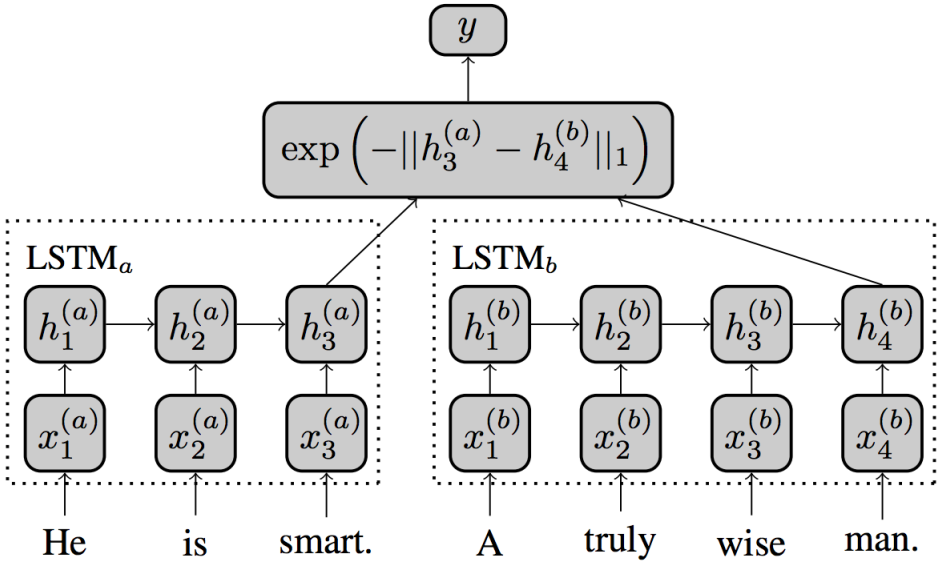


Figura 4.11.1: Arquitectura de la red siamesa.

En la figura 4.11.1 se ilustra la arquitectura de interna de la red siamesa que se utilizó.

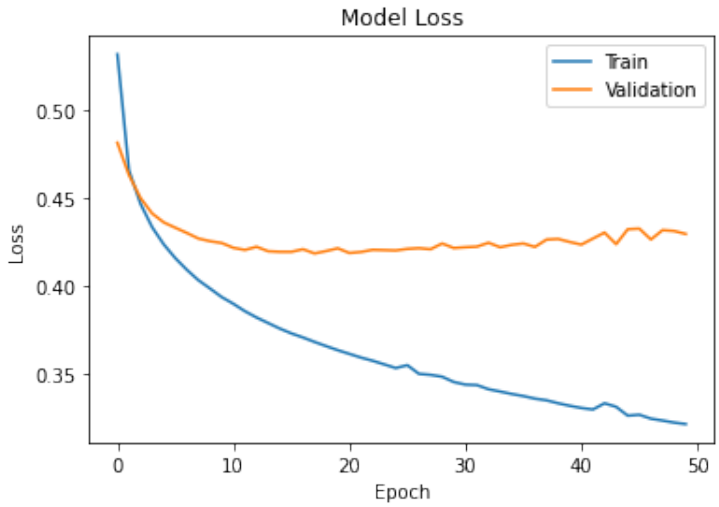


Figura 4.11.2: Gráfica de función de error de la red siamesa.

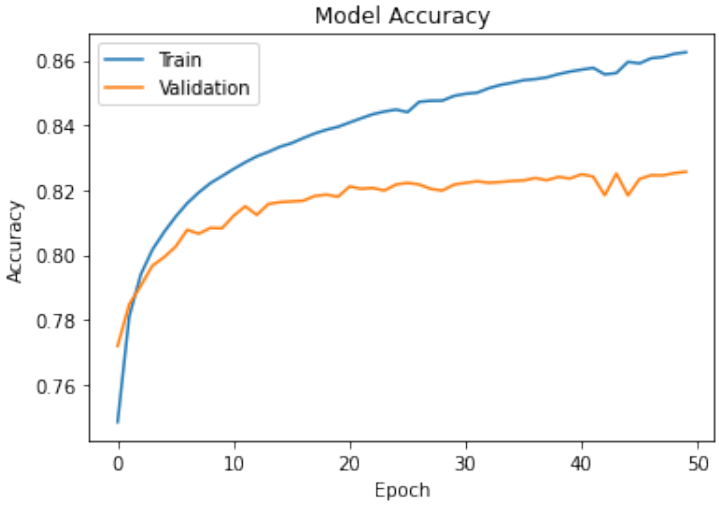


Figura 4.11.3: Gráfica de métrica *accuracy* de la red siamesa.

Una vez finalizado el entrenamiento se puede observar en la figura 4.11.2 y 4.11.3, como el valor que refleja la función de error va decayendo conforme van avanzando las épocas y como el valor de la métrica *accuracy* va aumentando.

Small BERT

La configuración de BERT que se utilizó fue SmallBert que tiene la siguiente configuración:

- Capas de transformación ocultas 4

- Tamaño del *embedding* 512

Esta arquitectura fue la adecuada tomando en cuenta las limitaciones de la infraestructura.

Este modelo ha seguido una adecuación distinta a los clasificadores clásicos, ya que al ser un paradigma con otro enfoque, muchos de los métodos utilizados no son compatibles o no son necesarios.

Como preprocesamiento se utilizó el *tokenizador* que provee la librería oficial de BERT. Una vez *tokenizadas* las preguntas, se debe concatenar cada pregunta con un separador SEP con el fin de saber donde es que termina una pregunta e inicia la siguiente, posteriormente, se convierte la lista entera de *tokens* y separadores a identificadores que pueda comprender el modelo y por último se convierte a *tensor* la lista.

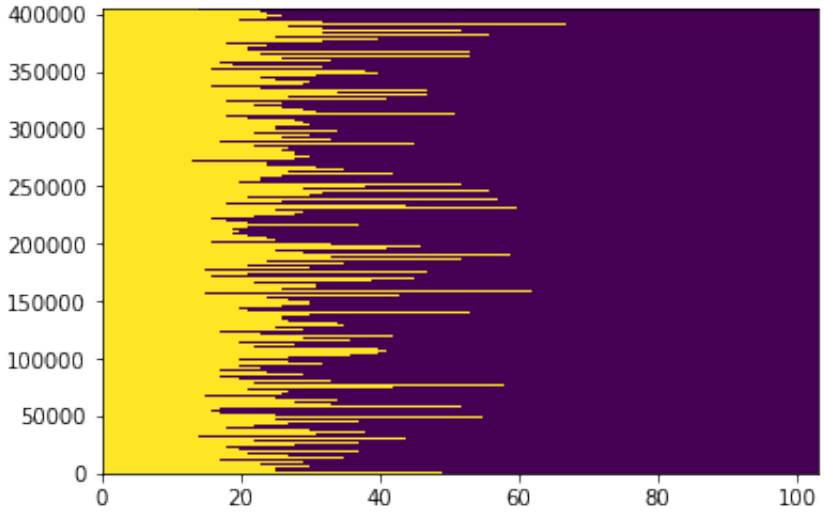


Figura 4.11.4: Tensores por par de preguntas con separador.

La gráfica 4.11.4 muestra la forma de los tensores finales para cada par de preguntas.

Como parte de las entradas a la red, se debe declarar dos tensores de enmascaramiento del mismo tamaño que el tensor principal.

El siguiente paso en el flujo es definir los parámetros generales del modelo donde se especifican los siguientes aspectos:

- hidden_size: 512
- hidden_act: gelu

- initializer_range: 0.02
- vocab_size: 30522
- hidden_dropout_prob: 0.1
- num_attention_heads: 8
- type_vocab_size: 2
- max_position_embeddings: 512
- num_hidden_layers: 4
- intermediate_size: 2048
- attention_probs_dropout_prob: 0.1

Donde los parámetros destacables son el número de capas ocultas, la función de activación y el número de cabezas de atención.

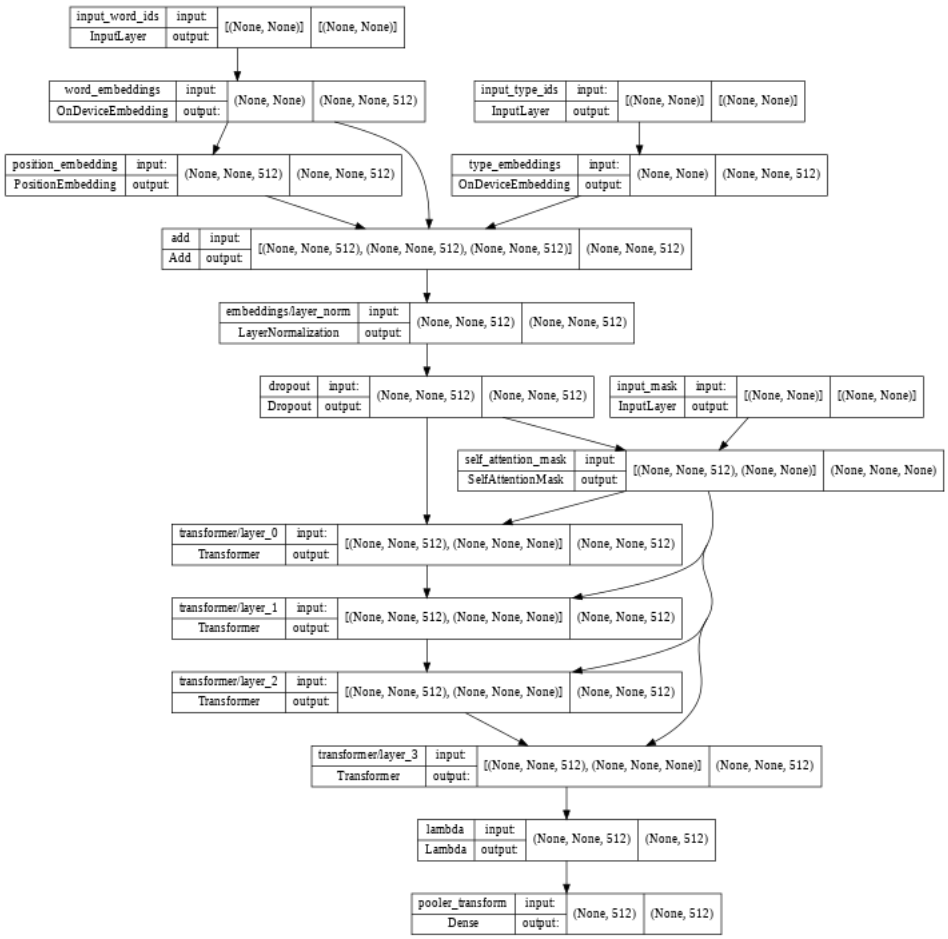


Figura 4.11.5: Arquitectura del codificador de Small BERT.

La figura 4.11.5 muestra la arquitectura del codificador de Small BERT.

Posteriormente, se cargó un *checkpoint* estable de la red de codificación de Small BERT para a partir de ese punto realizar un entrenamiento mucho más rápido y preciso.

Por último, se implementó el clasificador que provee por defecto la librería oficial de BERT, así como algunos parámetros donde destaca *warmup_steps* que permite un incremento gradual en el ritmo de aprendizaje, con el fin de que el entrenamiento se realice de forma más precisa.

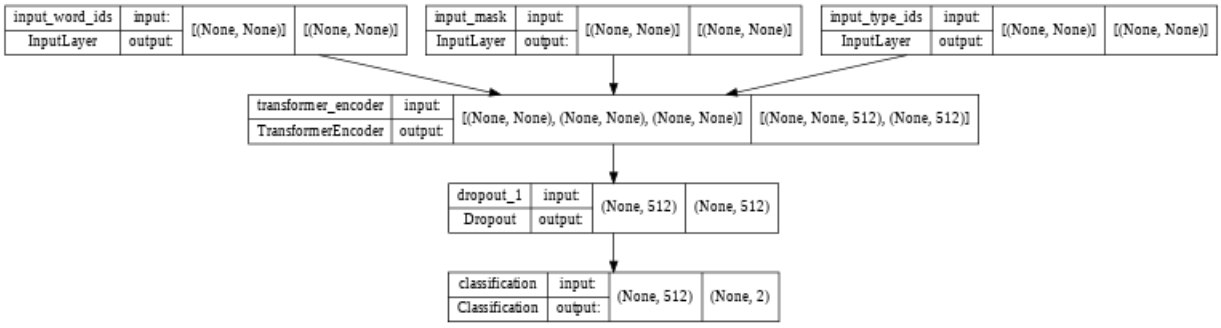


Figura 4.11.6: Arquitectura del clasificador de BERT.

La figura 4.11.6 muestra el flujo completo por el que pasan los datos desde la entrada hasta pasar por el clasificador.

Capítulo 5

Resultados

Los resultados obtenidos están dados de la siguiente manera:

1. Clasificadores Clásicos: Divididos en 2 bloques (conjunto de datos balanceado y sin balancear) de 3 experimentos cada uno.
2. Clasificadores Modernos: Únicamente en 1 bloque, con el conjunto de datos sin balancear.

5.1. Clasificadores clásicos

Conjunto Balanceado

1. Sin PCA completas (Características totales: 1536 + 25 = 1561)
2. Con PCA (Características totales: 800)
3. Sólo características automáticas (1536)
4. Sólo características manuales (Características totales: 25)

Conjunto de Datos Sin Balancear

1. Sin PCA completas (Características totales: 1536 + 25 = 1561)
2. Con PCA (Características totales: 800)
3. Sólo características automáticas (1536)
4. Solo características manuales (Características totales: 25)

Clasificadores:

- Random Forest
- XGBoost (2 configuraciones)
- SVC (Support vector Classifier)

5.1.1. Resultados Clásicos

Resultados del Conjunto de Datos Sin Balancear

A través de esta sección, se plasman los resultados de los diferentes clasificadores con las métricas *Accuracy* y *F1* al conjunto de datos QQP sin balancear y balanceado.

Tabla 5.1.1: Resultados de la métrica *Accuracy* de los clasificadores aplicados al conjunto de datos sin balancear, con **1561 características.**

id	Clasificador	Resultados
1	Random Forest	87.52 %
2	XGBoost 1	89.99 %
3	XGBoost 2	90.04 %
4	SVC(Support Vector Clasifier)	85.00 %

Tabla 5.1.2: Resultados de la métrica *F1* de los clasificadores aplicados al conjunto de datos sin balancear, con **1561 características.**

id	Clasificador	Resultados
1	Random Forest	83.28 %
2	XGBoost 1	90.29 %
3	XGBoost 2	90.11 %
4	SVC(Support Vector Clasifier)	81.00 %

Tabla 5.1.3: Resultados de la métrica *Accuracy* de los clasificadores aplicados al conjunto de datos sin balancear y **con PCA, reducido a 800 características.**

id	Clasificador	Resultados
1	Random Forest	87.76 %
2	XGBoost 1	88.42 %
3	XGBoost 2	89.18 %
4	SVC(Support Vector Clasifier)	85.00 %

Tabla 5.1.4: Resultados de la métrica *F1* de los clasificadores aplicados al conjunto de datos sin balancear y con **PCA, reducido a 800 características.**

id	Clasificador	Resultados
1	Random Forest	83.77 %
2	XGBoost 1	84.53 %
3	XGBoost 2	85.55 %
4	SVC(Support Vector Clasifier)	83.50 %

Tabla 5.1.5: Resultados de la métrica *Accuracy* a clasificadores aplicados al conjunto de datos sin balancear con **25 características manuales.**

id	Clasificador	Resultados
1	Random Forest	84.38 %
2	XGBoost 1	84.41 %
3	XGBoost 2	84.40 %
4	SVC(Support Vector Clasifier)	82.48 %

Tabla 5.1.6: Resultados de la métrica *F1* a clasificadores aplicados al conjunto de datos sin balancear con **25 características manuales.**

id	Clasificador	Resultados
1	Random Forest	79.54 %
2	XGBoost 1	79.59 %
3	XGBoost 2	79.55 %
4	SVC(Support Vector Clasifier)	81.50 %

Resultados del Conjunto de Datos Balanceado

A través de esta sección, se plasman los resultados de los diferentes clasificadores con las métricas *Accuracy* y *F1* al conjunto de datos QQP balanceado.

Tabla 5.1.7: Resultados de la métrica *Accuracy* de los clasificadores aplicados al conjunto de datos balanceado, con **1561 características.**

id	Clasificador	Resultados
1	Random Forest	86.74 %
2	XGBoost 1	87.67 %
3	XGBoost 2	87.68 %
4	SVC(Support Vector Clasifier)	83.00 %

Tabla 5.1.8: Resultados de la métrica *F1* de los clasificadores aplicados al conjunto de datos balanceado, con **1561 características.**

id	Clasificador	Resultados
1	Random Forest	87.56 %
2	XGBoost 1	88.08 %
3	XGBoost 2	88.08 %
4	SVC(Support Vector Clasifier)	83.00 %

Tabla 5.1.9: Resultados de la métrica *Accuracy* de los clasificadores aplicados al conjunto de datos balanceado y **con PCA, reducido a 800 características.**

id	Clasificador	Resultados
1	Random Forest	86.95 %
2	XGBoost 1	88.70 %
3	XGBoost 2	88.79 %
4	SVC(Support Vector Clasifier)	83.00 %

Tabla 5.1.10: Resultados de la métrica *F1* de los clasificadores aplicados al conjunto de datos balanceado y con **PCA, reducido a 800 características.**

id	Clasificador	Resultados
1	Random Forest	87.71 %
2	XGBoost 1	89.06 %
3	XGBoost 2	89.12 %
4	SVC(Support Vector Clasifier)	82.50 %

Tabla 5.1.11: Resultados de la métrica *Accuracy* a clasificadores aplicados al conjunto de datos balanceado con **25 características manuales.**

id	Clasificador	Resultados
1	Random Forest	84.78 %
2	XGBoost 1	84.86 %
3	XGBoost 2	84.86 %
4	SVC(Support Vector Clasifier)	83.00 %

Tabla 5.1.12: Resultados de la métrica *F1* a clasificadores aplicados al conjunto de datos balanceado a QQP con **25 características manuales.**

id	Clasificador	Resultados
1	Random Forest	85.50 %
2	XGBoost 1	85.55 %
3	XGBoost 2	85.55 %
4	SVC(Support Vector Clasifier)	82.50 %

Para tener una comparación general, las tablas 5.1.13 y 5.1.14 muestran una compración general para las metricas *Accuracy* y *F1*.

Tabla 5.1.13: Comparación de resultados de la métrica *Accuracy* en datos balanceados y sin balancear.

No Balanceados					
		Características			
	Modelos	*1561*	*1536*	*800*	*25*
1	*Random Forest*	87.52 %	82.12 %	87.76 %	84.38 %
2	*2 XGBoost 1*	89.99 %	85.41 %	88.42 %	83.41 %
3	*XGBoost 2*	90.04 %	85.40 %	89.18 %	84.40 %
4	*SVC(Support Vector Clasifier)*	85.00 %	74.00 %	85.00 %	82.48 %
Balanceados					
		Características			
	Modelos	*1561*	*1536*	*800*	*25*
1	*Random Forest*	86.74 %	89.76 %	86.95 %	84.78 %
2	*2 XGBoost 1*	87.67 %	85,35 %	88.70 %	84.86 %
3	*XGBoost 2*	87.68 %	85.51 %	88.79 %	84.86 %
4	*SVC(Support Vector Clasifier)*	83.00 %	70.00 %	83.00 %	83.00 %

Tabla 5.1.14: Comparación de resultados de la métrica *F1* en datos balanceados y sin balancear.

No Balanceados					
		Características			
	Modelos	*1561*	*1536*	*800*	*25*
1	*Random Forest*	83.28 %	69.88 %	83.77 %	79.54 %
2	*2 XGBoost 1*	90.29 %	80.48 %	84.53 %	79.59 %
3	*XGBoost 2*	90.11 %	80.45 %	85.55 %	79.55 %
4	*SVC(Support Vector Clasifier)*	81.00 %	69.50 %	83.50 %	81.50 %
Balanceados					
		Características			
	Modelos	*1561*	*1536*	*800*	*25*
1	*Random Forest*	87.56 %	78.89 %	87.71 %	85.50 %
2	*2 XGBoost 1*	88.08 %	85.85 %	89.06 %	85.55 %
3	*XGBoost 2*	88.08 %	85.99 %	89.12 %	85.55 %
4	*SVC(Support Vector Clasifier)*	83.00 %	70.00 %	82.50 %	81.50 %

5.1.2. Resultados modernos

Clasificadores:

1. SmallBERT: Con un *embedding* de tamaño 512.
2. LSTM: Con un **embedding** de tamaño 300.

Resultados Modernos

A través de esta sección se plasman los resultados para los clasificadores modernos. En la gráfica 5.1.15 se muestra los resultados de la métrica *Accuracy* y en la tabla 5.1.16 de la métrica *F1*

Tabla 5.1.15: Resultados de la métrica *Accuracy* a clasificadores modernos.

id	**Clasificador**	**Resultados**
1	SmallBERT	90.40 %
2	LSTM	82.57 %

Tabla 5.1.16: Resultados de la métrica *F1* a clasificadores modernos

id	Clasificador	Resultados
1	SmallBERT	90.45 %
2	LSTM	75.23 %

5.1.3. Método Propuesto

A partir del análisis anterior, se propone definir la familia de métodos llamada MADEPAC: Método de Análisis Profundo antes de la Clasificación.

MADEPAC proporciona un estudio completo de la naturaleza de los datos para la tarea de identificación de paráfrasis (estructura, clases, repeticiones, patrones, etc.), con el fin de obtener mejores resultados en clasificadores clásicos y poder aplicar de una mejor manera los modernos.

5.2. Mejores Resultados del Análisis

En la tabla 5.2.1 se muestran los mejores 5 resultados para la métrica Accuracy y en la tabla 5.2.2 para la métrica F1, para el conjunto de experimentos.

Los clasificadores clásicos han sido sometidos al método denominado MADEPAC (Método de Análisis de Datos Exploratorios Profundo antes de la Clasificación). A partir de este momento, veremos el resultado de los clasificadores junto con la etiqueta *MADEPAC* y aquellos que modelos que no contengan esta clasificación *SMADEPAC*

1. Random Forest / MADEPAC
2. XGBoost 1 / MADEPAC
3. XGBoost 2 / MADEPAC
4. SVC (Support Vector Classifier) / MADEPAC

Aquellos modelos modernos serán nombrados como:

1. Small BERT/ SMADEPAC
2. LSTM/ SMADEMAC

Tabla 5.2.1: Mejores resultados de la métrica *Accuracy* del conjunto de datos QQP con y sin balanceo, aplicados a diferentes clasificadores

id	Clasificador	Resultados	Características	MADEPAC
1	Small BERT	90.40 %	Completo	NO
2	XGBoost 2	90.04 %	1561 / Sin balancear	SI
3	XGBoost 1	89.99 %	1561 / Sin balancear	SI
4	XGBoost 2	88.79 %	800 / Balanceado	SI
5	Random Forest	89.76 %	1536 / Balanceado	SI

Tabla 5.2.2: Mejores resultados de la métrica *F1* del conjunto de datos QQP con y sin balanceo, aplicados a diferentes clasificadores

.

id	Clasificador	Resultados	Configuración	MADEPAC
1	Small Bert	91.45 %	Completo	NO
2	XGBoost 1	90.29 %	1561 / Sin balancear	SI
3	XGBoost 2	90.11 %	1561 / Sin balancear	SI
4	XGBoost 2	89.12 %	800 / Balanceado	SI
5	XGBoost 1	89.06 %	800 / Balanceado	SI

Las tablas 5.2.4 y 5.2.4 muestran los resultados de la matriz de confusión de MADEPAC 2 Y 3.

Tabla 5.2.3: Matriz de confusión para MADEPAC 2

	Positivo	Negativo
Positivos	44301	6934
Negativos	3279	47496

Tabla 5.2.4: Matriz de confusión para MADEPAC 3

	Positivo	Negativo
Positivos	43923	6949
Negativos	3506	47632

5.3. Comparación con el estado del arte

En la tabla 5.3.2 de nuestros modelos con el estado del arte para la métrica *accuracy* y en la tabla 5.3.1 para la métrica *F1* en la tabla

Tabla 5.3.1: Estado del arte vs. clasificadores propuestos en la métrica *Accuracy*

.

id	Clasificador	Resultados
1	data2vec[1]	92.4 %
2	Charformer-tall[2]	91.4 %
3	RealFormer[3]	91.34 %
4	SructBERT Roberta[4]	90.7 %
5	**SmallBERT/ SMADEPAC**	**90.40 %**
6	**XGBOOST2/ MADEPAC**	**90.04 %**

Tabla 5.3.2: Estado del arte vs. clasificadores propuestos en la métrica *F1*

.

id	Clasificador	Resultados
1	**SmallBERT/ SMADEPAC**	**90.45 %**
2	**XGBOOST 1/ MADEPAC**	**90.29 %**
3	EFL[5]	89.2 %
4	Charformer-tall[2]	88.5 %
5	RealFormet[3]	88.28 %
6	FNet-Large[6]	85 %
7	SructBERT Roberta[4]	74.4 %

Notas

[1](Baevski et al., 2022)

[2](Tay et al., 2021)

[3](He et al., 2020)

[4](Wang et al., 2019)

[5](Wang et al., 2021)

[6](Lee-Thorp et al., 2021)

Capítulo 6

Conclusiones

1. El método planteado, denominado MADEPAC, permite encontrar patrones no proporcionados por el análisis inicial.

2. A través del cálculo de proporción de varianza se puede descubrir el porcentaje de importancia que posee un número determinado de características. Con este cálculo se puede saber el número mínimo de características que se necesita para cubrir por lo menos el 97 % de la variabilidad del conjunto de datos. Con esto y a través del uso del PCA, se toman en cuenta las características manuales y automáticas (en ese orden), dando como resultado que solo se necesitan 800 características de las 1561 características iniciales, lo cual permite reducir el número de características en un 48.75 %, es decir un 51.2 % de las características originales.

3. Los resultados pueden ser aproximadamente comparados con respecto al estado del arte, por lo cual no es posible una comparación directa.

4. Derivado del ingreso de las características al PCA (manuales + automáticas), se comprobó que las características automáticas son las que más impacto tienen, esto se puede observar en la figura 4.9.1.

5. Aunado a los resultados de PCA, a través de las tablas 5.1.13 para la métrica *Accuracy* y 5.1.14 para la métrica F1, se demuestra que las características manuales propuestas en este trabajo, alcanzan resultados cercanos y en algunos modelos superiores, teniendo como comparación únicamente las características extraídas de manera automática.

6. Utilizando la técnica de reducción de la clase mayoritaria, se obtuvieron mejores resultados para los clasificadores de *XGBOOST 1 Y 2* (87.67 % y 87.68 %), comparados con SVC y *Random Forest* (85.00 % y 87.52 %) que utilizaron el conjunto con desbalance.

7. Al realizar un conocimiento de los datos previo a la clasificación, los modelos utilizan menos recursos al conocer el objetivo que se quiere lograr.

8. Gracias a MADEPAC las configuraciones de XGBOOST alcanzan resultados muy cercanos a los transformers.

Capítulo 7

Trabajo a futuro y Aportaciones

7.1. Aportaciones

- Unión y crecimiento en el número de características, estas basadas en el Análisis Exploratorio de Datos a través de un extractor automatizado (SBERT) y características manuales (Véase Capítulo 5).
- Se propone un nuevo conjunto de características que han permitido mejorar la clasificación. (Véase Capítulo 5).
- Selección de características ideales en el número de características para mejores resultados en los algoritmos seleccionados, esto al reducir el número de características ingresadas a los mismos a un 48 %(Para el caso de las 1556 a 800 características) y en un 98 % (Para el caso de 1556 a 25 características) (Véase Capítulo 5).
- Múltiples experimentos basados en diferentes configuraciones del método (conjunto de datos balanceado, sin balancear, diferente número de características y clasificadores) (Véase Capítulo 6).

7.2. Trabajo a futuro

Aunque cierto número de trabajos se han publicado en torno a la identificación de paráfrasis, es importante resaltar el uso de herramientas de PLN para el reconocimiento de patrones de alto nivel y la mejora de los algoritmos clásicos que pueden igualar aquellos métodos como los de *transformers*.

Como trabajo a futuro se puede mencionar en primer lugar una generalización en el planteamiento de nuestro procedimiento denominado **MADEPAC (Método de Análisis de Datos Exploratorios Profundo antes de la Clasificación)**, esto con el fin de ser aplicado a diferentes conjuntos de datos con más de dos clases. Esto lograría que nuestra aportación se convirtiera en un método previo a la clasificación, tomando gran relevancia el estudio y conocimiento de los datos y poniéndolo al mismo nivel de importancia de la clasificación final, y garantizar mejores resultados que sin la utilización del mismo.

Para mejorar los resultados en el PCA se piensa aumentar la extracción de características manual, basadas en las medidas de distancias (Jaccard, Hamming, subcadena común más larga (*Longest Common Substring, LCS*, etc) además de explorar aquellas que no están pensadas para estas tareas.

Por otra parte, como trabajo a futuro se planea ingresar que las características extraídas manualmente y las totales se proceden a través de nuestro propio *encoder* para ser convertidas en un *embedding*, esto para poder ser ingresadas a mecanismos de transformer y mejorar los resultados.

Por último, el presente trabajo representa una mejora para la tarea de PLN en cuestión, ya que basados en este conocimiento de patrones de paráfrasis, es decir, similitudes de ideas, se planea realizar la tarea, contraría: Generación de textos basados en la similitud de ideas, a través del conocimiento de patrones de estilo de escritura de un autor, en los niveles textuales más profundos, analizando las preferencias de palabras, de sintaxis y aquellos cambios que se dan cuando se realizan las traducciones entre idiomas.

Capítulo 8

Trabajos derivados de esta tesis

8.1. Publicaciones

Alcantara, T., Calvo, H. (2022). **Exploratory Data Analysis for the Automatic Detection of Question Paraphrasing in Collaborative Environments.**

In: Pichardo Lagunas, O., Martínez-Miranda, J., Martínez Seis, B. (eds) Advances in Computational Intelligence. Lecture Notes in Computer Science, vol 13613. Springer, Cham.

8.2. Congresos

21st Mexican International Conference on Artificial Intelligence. (MICAI 2022)
Presentación Oral: **Exploratory Data Analysis for the Automatic Detection of Question Paraphrasing in Collaborative Environments**
24 al 29 de Octubre, "Salón de Congresos", Tecnológico de Monterrey (ITESM), Monterrey, Mexico. Organizado por la Sociedad Mexicana de Inteligencia Artificial (SMIA).

Bibliografía

Arase, Y. and Tsujii, J. (2019). Transfer fine-tuning: A BERT case study. pages 5393–5404. 46

Baevski, A., Hsu, W., Xu, Q., Babu, A., Gu, J., and Auli, M. (2022). data2vec: A general framework for self-supervised learning in speech, vision and language. *CoRR*, abs/2202.03555. 45, 76

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473. 13, 22, 23, 24

Barrios, J. (2021). La matriz de confusión y sus métricas – inteligencia artificial –. 40

Beyer, H. (1981). Tukey, john w.: Exploratory data analysis. addison-wesley publishing company reading, mass. — menlo park, cal., london, amsterdam, don mills, ontario, sydney 1977, XVI, 688 s. *Biometrical Journal*, 23(4):413–414. 39

Bhagat, R. and Hovy, E. (2013). Squibs: What is a paraphrase? *Computational Linguistics*, 39(3):463–472. 3

Briggs, J. (2021). *How to build a wordpiece tokenizer for bert*. Towards Data Science. 11, 10

Calvo, H. (2013). *Procesamiento práctico de Lenguaje Natural*. Sociedad Mexicana de Inteligencia Artificial, A.C., México. 8

Camizuli, E. and Carranza, E. J. (2018). Exploratory data analysis (EDA). pages 1–7. 38

ChatBot, P. (2019). Cómo construir una red neuronal profunda desde cero usando julia - planeta chatbot. `https://planetachatbot.com/como-construir-red-neuronal-profunda-usando-julia/`. 13, 20

Chen, Z., Zhang, H., Zhang, X., and Zhao, L. (2018). Quora question pairs. *Universityof Waterloo*. 45

Chopra, A., Agrawal, S., and Ghosh, S. (2020). Applying transfer learning for improving domain-specific search experience using query to question similarity.

Chowdhary, K. (2020). *Fundamentals of Artificial Intelligence*. Springer. 12, 6, 7, 8

Demuth, H. and Jesús, B. D. (September, 2014). *Neural Network Design*, chapter 2. Martin Hagan. 12, 17, 18

Demuth, H. B., Beale, M. H., De Jess, O., and Hagan, M. T. (2014). *Neural Network Design*. Martin Hagan, Stillwater, OK, USA, 2nd edition. 16

Deng, L. and Liu, Y. (2018). *Deep Learning in Natural Language Processing*. Springer Publishing Company, Incorporated, 1st edition.

Devlin, J. (2020). Github - google-research/bert: Tensorflow code and pre-trained models for bert. `https://github.com/google-research/bert`. 13, 34, 35

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805. 13, 28, 29, 30, 32, 33

Flores, R. (2014a). Citas y referencias. recomendaciones y aspectos básicos del estilo APA, biblioteca de la universidad de lima. *Principios para citar parafrasear y resumir; Cómo evitar el plagio accidental.* 3

Flores, R. (2014b). Citas y referencias. recomendaciones y aspectos básicos del estilo APA, biblioteca de la universidad de lima. *Principios para citar parafrasear y resumir; Cómo evitar el plagio accidental.* 36

Godbole, A., Dalmia, A., and Sahu, S. K. (2018). Siamese neural networks with random forest for detecting duplicate question pairs. *CoRR*, abs/1801.07288. 45

He, R., Ravula, A., Kanagal, B., and Ainslie, J. (2020). Realformer: Transformer likes residual attention. *CoRR*, abs/2012.11747. 44, 76

Iyer, S., Dandekar, N., and Csernai, K. (2021). First quora dataset release: Question pairs. 11, 42, 43

Lee-Thorp, J., Ainslie, J., Eckstein, I., and Ontañón, S. (2021). Fnet: Mixing tokens with fourier transforms. *CoRR*, abs/2105.03824. 44, 76

Lynn, S. (2018). Intro to word embeddings and vectors for text analysiss. 12, 11, 13

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133. 17

Montanes, J. (2021). Bert transformers — how do they work? 13, 30, 31, 32, 33, 34

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. 12, 13, 14, 57

Rishickesh, R., A.Shahina, R. K., and Khan, A. N. (2019). Identification of duplication in questions posed on knowledge sharing platform quora using machine learning techniques. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*. 45

Rivera, M. (2018). Introducción a redes neuronales recurrentes (rnn). 13, 20, 21

Sanjay, C. (2021). The quora question pair similarity problem. `https://towardsdatascience.com/the-quora-question-pair-similarity-problem-3598477af172`. (Accessed on 08/25/2022).

Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77–89. 40

Tay, Y., Tran, V. Q., Ruder, S., Gupta, J. P., Chung, H. W., Bahri, D., Qin, Z., Baumgartner, S., Yu, C., and Metzler, D. (2021). Charformer: Fast character transformers via gradient-based subword tokenization. *CoRR*, abs/2106.12672. 44, 76

Torres, J. (Junio, 2018). *Deep Learning, Introducción práctica con Keras*, chapter 7. Lulu Press, Inc. 20, 21

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762. 11, 13, 24, 25, 26, 27, 28, 29

Wang, S., Fang, H., Khabsa, M., Mao, H., and Ma, H. (2021). Entailment as few-shot learner. *CoRR*, abs/2104.14690. 44, 76

Wang, W., Bi, B., Yan, M., Wu, C., Bao, Z., Peng, L., and Si, L. (2019). Structbert: Incorporating language structures into pre-training for deep language understanding. *CoRR*, abs/1908.04577. 45, 76

Wang, Z., Hamza, W., and Florian, R. (2017). Bilateral multi-perspective matching for natural language sentences. pages 4144–4150.

Wood, T. (2018). Definición de la función softmax — ia profunda. 11, 15

Zhang, R., Zhou, Q., Wu, B., Li, W., and Mo, T. (2020). What do questions exactly ask? mfae: Duplicate question identification with multi-fusion asking emphasis. pages 226–234. 46